ARI ELISEI VILELA

SISTEMA DE COLETA DE DADOS DE UMA MICROBALANÇA DE QUARTZO

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras, na disciplina de Projeto Orientado, como parte das exigências do curso de Bacharelado em Ciência da Computação, para obtenção do título de Bacharel em Ciência da Computação.

Orientador

Prof. Wilian Soares Lacerda

LAVRAS
MINAS GERAIS – BRASIL
2001

ARI ELISEI VILELA

SISTEMA DE COLETA DE DADOS DE UMA MICROBALANÇA DE QUARTZO

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras, na disciplina de Projeto Orientado, como parte das exigências do curso de Bacharelado em Ciência da Computação, para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 25 de março de 2002

Prof. Mauro dos Santos Carvalho
(Co-orientador)

Prof. Antonio Maria Pereira de Resende
(Co-orientador)

Prof. Wilian Soares Lacerda
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL

Aos meus pais Antonio Ary e Ana pelo apoio, oportunidade e confiança, a minha irmã Ariana, a meu irmão Guto, a meus avós José, Julia e Assunta (in memorian) e minha namorada Andreia.

DEDICO

AGRADECIMENTO

Agradeço primeiramente a Deus por me dar força e saúde para alcançar este objetivo.

Ao professor e orientador Wilian pela atenção, paciência, ensinamentos e ajuda prestada no desenvolvimento deste trabalho e ainda por acreditar que tudo daria certo.

Ao professor Mauro pela ajuda, incentivo e por permitir a utilização da Microbalança de Quartzo.

Ao professor Antonio Maria pela grande colaboração prestada na escrita desta obra.

Ao alemão Sasha Maric pela grande colaboração prestada. Ao aluno de graduação em Agronomia Lucas e a todos os professores do DCC que de certa maneira participaram do desenvolvimento deste projeto com os ensinamentos passados no decorrer do curso.

Agradeço também aos colegas Guilherme e Marcelo que muito me ajudaram no decorrer do curso. Aos grandes amigos Flavio, Paulo Leandro, Eduardo (Nelore), Agnaldo, Gustavo, Leandro, Rondinelli, Rodrigo Felício, Rodrigo Marinho, a toda a galera da sala pelos bons momentos vividos, ao grande amigo Sérgio que hoje não se encontra mais em nosso meio e aos amigos de república VC, Daniel, Valdir e Tácio.

RESUMO

Esta monografia apresenta o desenvolvimento de um sistema para computador IBM-PC que coleta todos os dados medidos por uma Microbalança de Quartzo. A coleta destes dados é realizada via porta serial RS-232 de nove pinos. Os dados coletados pelo PC são analisados pelo sistema, salvos e visualizados através de sua interface gráfica. Estes dados ainda podem ser visualizados em formato gráfico, que é traçado com os dados adquiridos (freqüência de oscilação dos sensores) em função de seu tempo de coleta. Os dados são salvos em dois formatos diferentes de arquivos: o formato (*dat) que é compatível a outros aplicativos e o formato (*mbq) que é o formato do sistema, sendo que o mesmo pode ser visualizado pelo sistema tanto na interface gráfica quanto no modo gráfico. Todo este sistema foi desenvolvido utilizando a ferramenta de desenvolvimento Delphi 5.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

Com as inovações tecnológicas advindas da Terceira Revolução Industrial ou Revolução Digital, novos equipamentos e sistemas foram desenvolvidos.

Devido a esta “Revolução” e a chamada Era da Informação, foi desenvolvido um equipamento utilizado para detecção e padronização de aromas ou padrões olfativos, chamado de “nariz eletrônico”, introduzidos no mercado no início da década de 90 [13].

Os narizes eletrônicos são aparelhos constituídos por conjuntos de sensores químicos com especificidades parciais a diferentes classes de compostos químicos, associados a modelos estatísticos de análise que permitem identificar e separar substâncias odorantes (como o café) em complexos sistemas de odores [12] [13] [14].

Os aparelhos mais comercializados possuem sensores de quartzo em sua constituição, sendo denominados de Microbalanças de Quartzo. Recebem esse nome, em conseqüência do seu funcionamento ser similar a de uma balança, onde as respostas são obtidas proporcionalmente devido à mudança de massa do conjunto sensitivo quando da absorção na superfície sensitiva [12].

Como a Microbalança não possui interface gráfica, é necessário que se desenvolva um sistema para computadores IBM-PC, capaz de coletar todos os dados por ela medidos.

O novo sistema terá como objetivo coletar todos os dados medidos pelos doze sensores piezoelétricos da Microbalança.

Os dados a serem medidos pela Microbalança serão de uma amostra de café. Esses dados serão coletados pelo sistema e visualizados em sua interface gráfica. O modo de visualização pode ser através de gráfico, traçado com os dados adquiridos (freqüência de oscilação dos sensores) em função do seu tempo de coleta ou em uma tabela temporária onde eles ficam armazenados na interface principal.

Todos os dados coletados são salvos independentes ou não da vontade do usuário. Utiliza-se de dois formatos diferentes de arquivos para salvar os dados: (*dat) e (*mbq). O formato (*dat) é compatível a outros aplicativos podendo assim ser utilizado para outras funções. O formato (*mbq) é o formato do sistema utilizado para visualizar os dados salvos no próprio sistema.

O Sistema de Coleta de Dados de uma Microbalança de Quartzo é implementado utilizando a ferramenta de desenvolvimento Delphi 5, sendo um programa de código aberto para futuras atualizações e melhorias.

Nos próximos capítulos serão apresentados:

- Revisão de literatura;
- Proposta de Desenvolvimento;
- Metodologia de Desenvolvimento do Sistema;
- Resultados e Discussões;
- Conclusão;
- Proposta de Continuidade.

2. REVISÃO DE LITERATURA

Neste capitulo será feita uma revisão de tudo que for relacionado ao desenvolvimento do sistema como:

- Sistemas de Numeração Hexadecimal;
- Interface de Comunicação Serial;
- Interface Serial RS-232;
- UML (Linguagem de Modelagem Unificada);
- Componentes de Comunicação Serial do Delphi 5;
- Microbalança de Quartzo ou “Narizes Eletrônicos”.

2.1 Sistema de Numeração Hexadecimal

O sistema de numeração hexadecimal é largamente utilizado em trabalhos com microprocessadores, devido ao fato deles serem muito mais curtos que os números binários [2] [10].

Dessa maneira eles são fáceis de escrever e lembrar, podendo até convertê-los mentalmente em binários sempre que for necessário.

O sistema numérico hexadecimal tem uma base igual a dezesseis, que significa, que uma variedade de dezesseis dígitos é disponível para cada casa hexadecimal de um número.

Os dezesseis números hexadecimais são: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

A maneira mais fácil de aprender a contar em números hexadecimais é usando um odômetro hexadecimal. Este dispositivo hipotético é similar ao odômetro de um carro, exceto que cada rodinha indicadora tem dezesseis dígitos, numerados de 0 a F. Quando um rodinha comuta F para 0, ela envia um vai-um para a rodinha mais alta seguinte [10], como ilustra a Figura 1:

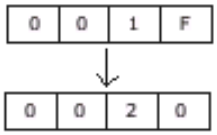

Figura 1: Contagem de números hexadecimais.

Sendo o número 001F o próximo número a ser computado será 0020; porque a última coluna está em F, sendo assim ela irá para 0 e enviará um vai-um para a coluna seguinte (a sua esquerda) que está em 1 e irá para 2.

A Tabela 1 faz uma comparação entre os números decimais, binários e hexadecimais:

Tabela 1 – Comparando decimal, binário e hexadecimal.

Decimal	Binário	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

2.1.1 Conversão de Hexadecimal para Binário

Para converter um número hexadecimal em binário basta converter cada dígito hexadecimal em seu equivalente de quatro bits em binário, usando o código dado na Tabela 1 [2] [10].

Como exemplo tomemos o número C5E2:

- C $\rightarrow$ 1100
- 5 $\rightarrow$ 0101
- E $\rightarrow$ 1110
- 2 $\rightarrow$ 0010

Assim formamos o número 1100 0101 1110 0010 em binário.

2.1.2 Conversão de Binário para Hexadecimal

Para converter na direção oposta, de binário para hexadecimal é só utilizar o código da Tabela 1 [2] [10].

Como exemplo tomemos o número 1110 1000 1101 0110:

- 1110 $\rightarrow$ E
- 1000 $\rightarrow$ 8
- 1101 $\rightarrow$ D
- 0110 $\rightarrow$ 6

Assim formamos o número E8D6 em hexadecimal.

2.2 Interface de Comunicação Serial

O nome de comunicação serial vem do fato que, é transferido um bit de cada vez do transmissor ao receptor [6].

Os dados na forma serial são aqueles que trafegam em uma única linha ou fio. Os bits de dados são transmitidos, a uma taxa de transmissão que é constante e em fase com um clock de referência [3] [6] [17].

2.2.1 Taxa de Transferência em Bauds

A freqüência de clock de um dado serial é normalmente chamada de taxa de transferência em baud. Uma taxa de transferência em baud típica para um teletipo é de 110 bits por segundo (110 bps). A esta taxa é possível enviar dez números binários de 11 bits em um segundo [3]. No caso da Microbalança de Quartzo a taxa de transferência dos dados está em torno de 9600 bps [11].

2.2.2 Método de Transmissão Serial

A forma de transmissão serial mais usada, e a que é usada no caso da Microbalança de Quartzo, é a transmissão serial assíncrona. Nela os dados chegam organizados em pequenas partes, com algum intervalo de tempo entre eles [3] [6] [11].

A Figura 2 ilustra o formato típico de um dado de sete bits que será transmitido pelo método assíncrono [3]:

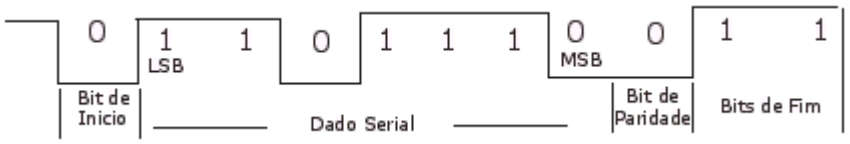

Figura 2: Formato serial assíncrono.

Cada palavra a ser transmitida por este método é iniciada por um bit de início ou “start bit”, identificado na Figura 2 pela borda de descida. É ele que inicia a operação de carregamento do registrador com a palavra ou dado serial que será transmitida [3] [6] [17].

A palavra ou dado pode variar de cinco a oito bits. Na Figura 2 é apresentado uma palavra de sete bits. Na Microbalança de Quartzo o número de bits a ser utilizado é de tamanho oito.

Os próximos sete bits chamados de dado serial na Figura 2 são normalmente apresentados na forma de dados no código ASCII (ANEXO A) para as letras do alfabeto. A transmissão é iniciada do bit menos significativo (LSB – Figura 2) ao bit mais significativo (MSB –Figura 2).

Completados os bits de informação que é o dado serial, podemos encontrar presente ou não o bit de paridade. A Microbalança de Quartzo não faz a utilização deste bit.

Finalmente para o encerramento da transmissão assíncrona temos os bits de fim ou "stop bit" que podem ser um, um e meio ou dois bits. Para a Microbalança de Quartzo será utilizado somente um único "stop bit".

Nota-se que para a transmissão do dado da Figura 2 gasta-se onze pulsos de clock.

2.3 Interface Serial RS-232

A interface RS-232 é um padrão já antigo, mas também muito popular. Foi criado para um único propósito, realizar a comunicação entre dois tipos de equipamentos (DTE e DCE) [3] [6] [16] [17].

O padrão RS-232 foi adotado em 1960 pela Associação das Indústrias Eletrônicas (EIA) para padronizar a transferência serial de dados usada por modens, sendo este mesmo o padrão utilizado pela Microbalança de Quartzo [3] [11] [16] [17].

Este padrão define os níveis de tensão para os estados lógicos 1 e 0, bem como outras características necessárias para fazer com que um equipamento de um sistema de computador se comunique com outro. A sigla RS significa Padrão Recomendado (Recommended Standard), 232 representa o número de identificação do padrão, e a letra C informa que este padrão já foi revisado três vezes [3].

Muitos dispositivos interessantes, tais como terminais, osciloscópios, impressoras, unidades de fitas, interconectam-se através da porta serial RS-232C. Em qualquer aplicação prática, é necessário utilizar circuitos que convertam os níveis TTL para os exigidos pela interface e também conversões para o caminho inverso.

As linhas RS-232C fornecem um caminho unidirecional, ponto a ponto, para uma distância de até 15,24m, a uma taxa máxima de transmissão de 20 Kbit/s. A regra de conversão de TTL para RS-232C é muito simples. Nível alto TTL é convertido em -12 V e nível baixo TTL é convertido em +12 V, como mostrado na Tabela 2:

Tabela 2 – Conversão entre TTL e RS-232C.

TTL	RS-232C
+5 V	-12 V
0 V	+12 V

Observe na Tabela 2 que a tensão para o estado 1 ou +5 V é mais baixa que a tensão para o estado 0 ou 0 V. Isto faz com que o padrão RS-232C seja um sistema de lógica negativa porque a tensão no estado 1 é negativa enquanto que a tensão no estado 0 é positiva.

A Figura 3 ilustra as seqüências de bits numa transmissão serial comparando o formato da transmissão TTL com o da RS-232C. Nota-se que na Figura 3 a palavra “marca” (+5V e –12V) é utilizada para caracterizar o estado binário 1 (bit = 1) e a palavra “espaço” (0V e +12V) para caracterizar o estado binário 0 (bit = 0) [16] [17].

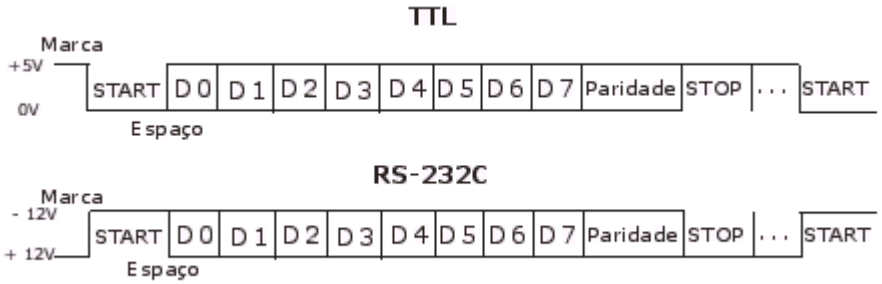

Figura 3: Seqüência de bits numa transmissão serial.

2.3.1 Descrição dos Pinos da Interface Serial RS-232

Existem dois tipos de conectores db-macho utilizados pela Interface RS-232C: o db-9 e db-25. O conector db-9 é o utilizado pela Microbalança de Quartzo [3] [6] [11] [16].

A Figura 4 ilustra o conector db-9 [1]:

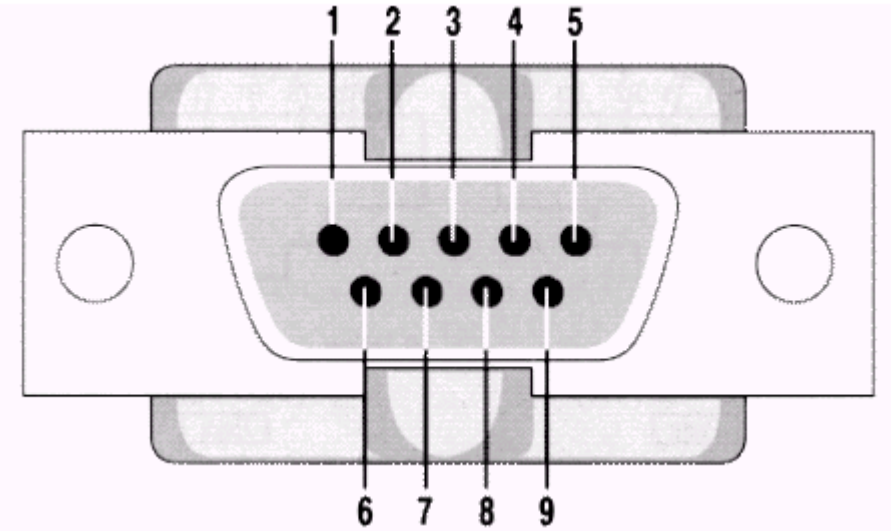

Figura 4: Conector db fêmea de nove pinos

O conjunto de nove pinos do conector db-9 possui dois tipos de sinais: sinais de entrada (recebidos pelo PC) e sinais de saída (enviados pelo PC).

São os seguintes, os sinais de entrada (para evitar confusões na tradução, serão utilizados os nomes já consagrados em inglês):

- Received Data – RD pino 2: por ele chegam os dados seriais;
- Clear to Send – CTS pino 8: por essa linha, o periférico informa que está pronto para transmitir os dados;
- Data Set Ready – DSR pino 6: por essa linha, o periférico informa que está pronto para se comunicar;
- Carrier Detect – CD pino 1: por essa linha, o periférico indica que detectou a portadora;
- Ring Indicator – RI pino 9: serve para o periférico, neste caso um modem, indicar o tom de discar da linha telefônica.

São os seguintes, os sinais de Saída (para evitar confusões na tradução, serão utilizados os nomes já consagrados em inglês):

- Transmitted Data – TD pino 3: por ele são enviados os dados seriais;
- Request to Send – RTS pino 7: informa ao periférico que está pronto para transmitir os dados;
- Data Terminal Ready – DTR pino 4: informa ao periférico que está pronto para se comunicar.

2.3.2 Estabelecendo Comunicação entre dois Computadores

Através da porta serial RS-232C é possível realizar a comunicação entre dois computadores. Para isso utiliza-se de um cabo serial chamado null-modem, conectado a dois conectores db-9 e ligados como é mostrado na Figura 5 [16]:

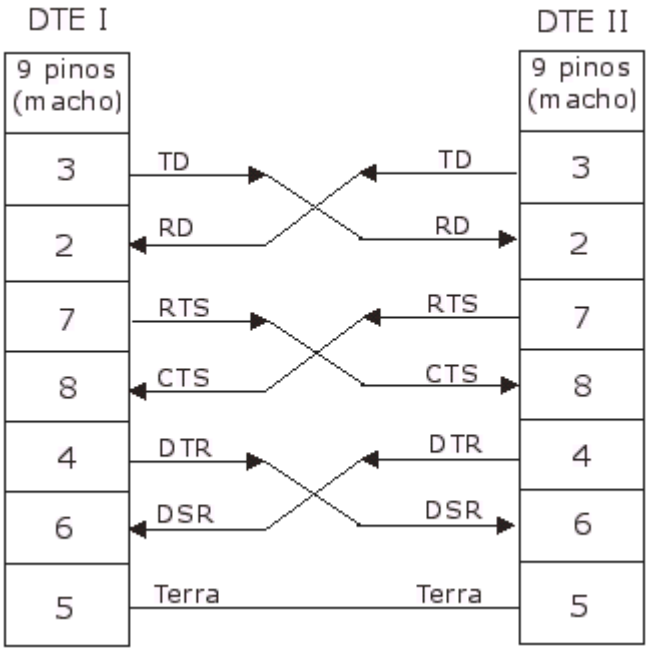

Figura 5: Cabo null-modem.

A palavra DTE (no conector db macho) na Figura 5 significa equipamento terminal de dados, ou seja, ele está nos extremos da comunicação [3] [6] [16].

Também existe o DCE (conector db fêmea) chamado de equipamento comunicador de dados, por onde passam os dados, ou seja, ele sempre estará no meio do caminho, como um modem.

A diferença entre o DTE e o DCE é que o DTE transmite pelo pino 3 e recebe pelo pino 2, como ilustra a Figura 4 e o DCE transmite pelo pino 2 e recebe pelo pino 3 [16].

Os demais pinos funcionam da seguinte maneira: o pino 4 (saída) do DTE I informa ao DTE II que já está pronto para se comunicar, e o DTE II recebe esta informação através do pino 6 (entrada).

O pino 7 (saída) do DTE I informa o momento em que ele está pronto para transmitir os dados, esta informação chega ao DTE II através do pino 8 (entrada). Completado isso o pino 3 (saída) do DTE I, transmite os dados que chegam pelo pino 2 (entrada) do DTE II.

O pino 5 é utilizado como uma ligação terra entre os terminais.

2.4 Linguagem de Modelagem Unificada (UML)

UML são as iniciais de Unified Modeling Language, que em português significam Linguagem de Modelagem Unificada. A UML é a padronização da linguagem de desenvolvimento orientado a objetos para visualização, especificação, construção e documentação de sistemas [7] [8] [9].

A UML se propõe a ser a linguagem definitiva para modelagem de sistemas orientada a objetos. Por ser unificada ela facilita que grupos de desenvolvimentos de software interpretem de uma maneira correta e sem ambigüidades modelos gerados por outros analistas ou grupos de desenvolvimento.

A UML começou a se concretizar no ano de 1994 com Grady Booch e Jim Rumbaugh através da unificação de seus métodos (Booch e OMT) com o apoio da Rational Corporation. No outono de 1995, Ivar Jacobson juntou-se a eles fundindo seu método (OOSE) ao método deles.

A idéia principal dos três era criar uma linguagem de modelagem unificada que tratasse de assuntos de escala inerentes a sistemas complexos e críticos.

Muitos reconheceram este esforço como um item estratégico, e assim muitas parceiras importantes foram acontecendo como a Microsoft, Hewlet-Packard, Oracle, IBM, Ericsson, Digital Equipament Corporation, Texas Instruments Software, MCI Systemhouse, Sterling Software, Expersoft Corporation, ICON Computing, Intellicorp, Forté, Greenbrier & Russel, i-Logix,

James Martin & Company, Jim Odell, Lockheed Martin Advanced Concepts Center, Persistence Software, POET Software Corporation, TIER Corporation, Visient, ObjectTime, Platinum Techonology, Ptech, Reich Tecnologies, Taskon, Softeam, Unisys. Essa colaboração produziu em janeiro de 1997 a versão 1.0 da UML (Unified Modeling Language) que foi batizada assim devido à unificação das linguagens de modelagem. Em setembro de 1997 surgiu a versão 1.1 não patenteada, aberta a todos e aprovada em novembro de 1997 pela OMG.

A UML pode ser usada para seis finalidades:

1. Mostrar as fronteiras de um sistema e suas funções principais utilizando atores e casos de uso;
2. Ilustrar a realização de casos de uso com diagramas de interação;
3. Representar uma estrutura estática de um sistema utilizando diagramas de classe;
4. Modelar o comportamento de objetos com diagramas de transição de estado;
5. Revelar a arquitetura de implementação física com diagramas de comportamento e de implantação;
6. Estender sua funcionalidade através de estereótipos.

No desenvolvimento do sistema da Microbalança foram utilizados os itens 1, 2 e 3 descritos acima.

2.4.1 Diagrama de Caso de Uso

Os diagramas de casos de uso têm como finalidade fornecer um modo que descreve a visão externa do sistema e suas interações com o mundo exterior [7] [8] [9].

Na modelagem de casos de uso, o sistema é visto como uma caixa-preta que fornece situações de aplicação (ou casos de uso), lembrando que neste

momento não é importante compreender como o sistema implementa os casos de uso ou como ocorre o seu funcionamento interno.

Os propósitos primários dos casos de uso são:

1. Descrever os requerimentos funcionais do sistema de maneira consensual entre usuários e desenvolvedores de sistemas;
2. Fornecer uma descrição consistente e clara sobre as responsabilidades que devem ser cumpridas pelo sistema além de formar a base para a fase de desenho;
3. Oferecer as possíveis situações do mundo real para o teste do sistema.

Um diagrama de caso de uso vem a ser um gráfico de atores, um conjunto de casos incluindo um limite de domínio, comunicação, participação e associações entre atores, assim como generalizações entre casos de uso.

Há quatro elementos básicos em um diagrama de caso de uso:

1. Ator;
2. Caso de Uso;
3. Interação;
4. Sistema.

A Figura 6 ilustra um diagrama de caso de uso [8]:

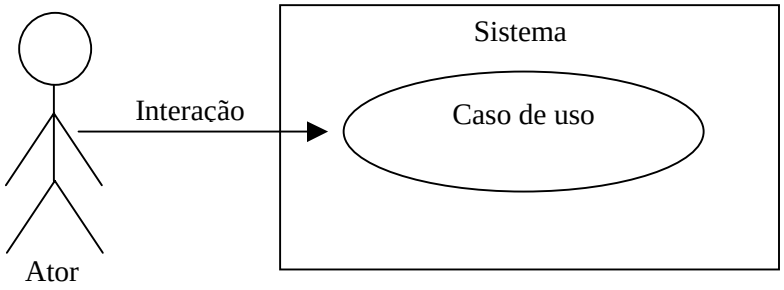

Figura 6: Diagrama de Caso de Uso.

2.4.1.1 Caso de Uso

A definição formal de caso de uso é; um conjunto de seqüências de ações que um sistema desempenha para produzir um resultado observável de valor a um ator específico [7] [8] [9].

Um caso de uso tem como propósito definir o comportamento de uma classe passiva, sem revelar a sua estrutura interna. As suas principais características são [8]:

1. Um caso de uso é sempre iniciado por um ator;
2. Um caso de uso é sempre completo;
3. Um caso de uso provê valor a um ator.

Cada caso de uso é uma seqüência completa de cenários de interação mostrando como eventos externos iniciais são respondidos no caso. Cada cenário é uma narrativa de uma parte do comportamento global do sistema. Já uma coleção completa de cenários é todo o sistema, relacionando uma seqüência de interações entre atores e o sistema com todas as decisões definidas.

Para identificar um caso de uso deve-se observar os seguintes aspectos [8]:

1. O ator precisa ler, criar, destruir, modificar ou armazenar algum tipo de informação no sistema?
2. O trabalho diário do ator pode ser simplificado ou tornado mais eficiente através de novas funções no sistema?
3. O ator tem de ser notificado sobre eventos no sistema ou ainda notificar o sistema em si?
4. Quais são as funções que o ator necessita no sistema?
5. O que o ator necessita fazer?
6. Quais são os principais problemas com a implementação atual do sistema?

7. Quais são as entradas e saídas, juntamente com sua origem e destino, que o sistema requer?

2.4.1.2 Ator

Um ator é uma entidade externa ao sistema que, de alguma, maneira, participa da história do caso de uso. Um ator, tipicamente, estimula o sistema com eventos de entrada ou recebe algo do mesmo [7] [8] [9].

Os atores são representados pelo papel que eles desempenham no caso de uso, tais como Clientes, Caixas, Usuários e assim por diante. É desejável começar com maiúsculas o nome dos atores no texto do caso de uso para facilidade de identificação.

Atores podem ser identificados a partir do seguinte questionamento [8]:

1. Quem utilizará a principal funcionalidade do sistema (atores principais)?
2. Quem irá manter, administrar e fazer com que o sistema permaneça operando (atores coadjuvantes)?
3. Quem proverá suporte ao sistema em seu processamento diário?
4. Quem ou o quê tem interesse nos resultados produzidos pelo sistema?
5. Quais dispositivos de hardware são necessários ao sistema?
6. Com quais outros sistemas o sistema em foco interage?

2.4.1.3 Interação em Caso de Uso

O ator comunica-se com o sistema através do envio e recebimento de mensagens, sendo que um caso de uso é sempre iniciado a partir do momento que um ator envia sua mensagem [7] [8] [9].

As seguintes interações são importantes dentro de um diagrama de caso de uso [8]:

1. Comunicação: Um ator comunica-se com o caso de uso, assim, cada participação sua é mostrada conectando-se o símbolo do ator ao símbolo do caso de uso, através de um caminho sólido.
2. Extensão; Usada para mostrar comportamento de exceção e casos especiais que aumentariam a quantidade de casos de uso no modelo. A extensão é desenhada através de uma seta de generalização como o esteriótipo <<extend>>, do caso de uso que fornece a extensão para o caso de uso básico. Um relacionamento de extensão seria, por exemplo, de um caso de uso A para outro caso de uso B, indicando que uma instância de B pode incluir um comportamento de A
3. Uso: Um relacionamento de uso entre casos de uso ocorre quando há uma parcela de comportamento similar entre eles sugerindo uma reutilização em vez de uma nova copia da descrição do comportamento. É desenhado com uma seta de generalização do caso de uso que faz o uso ao caso de uso que é usado com o seguinte esteriótipo <<usa>>. Um relacionamento de uso do caso de uso A para o caso de uso B indica que uma instância do caso de uso A também incluirá o comportamento como especificado por B.

2.4.2 Diagramas de Classes

Os diagramas de classes, a essência da UML, é resultado de uma combinação de diagramas propostos pela OMT, Booch e vários outros métodos [7] [8] [9].

Trata-se de uma estrutura lógica estática em uma superfície de duas dimensões mostrando uma coleção de elementos declarativos de modelo, tais

como, classes, tipos e seus respectivos conteúdos e relações. Há quatro tipos principais de relacionamentos no diagrama de classes [8]:

1. Generalização/Especificação: indica relacionamento entre um elemento mais geral e um elemento mais específico (respectivamente, superclasse e subclasse), também conhecido como herança ou classificação;
2. Agregação: usada para denotar relacionamentos todo/parte. (por exemplo, um mastro de um navio é parte do navio);
3. Associação: utilizada para denotar relacionamentos entre classes não correlatas (por exemplo, um cliente pode alugar varias fitas de vídeo). Na UML uma associação é definida como um relacionamento que descreve um conjunto de vínculos onde um vínculo é definido como uma conexão semântica entre tuplas de objetos.
4. Dependência: é um relacionamento entre elementos, um independente e outro dependente, onde uma mudança no elemento independente afetará o elemento dependente.

Estruturar atributos e operações em classes é fundamental para o trabalho de modelagem através do enfoque da orientação a objeto. Classes individuais são representadas na UML como um retângulo sólido com um, dois ou três compartimentos. O primeiro compartimento é obrigatório e é utilizado para colocar o nome da classe, o segundo e terceiro compartimentos são opcionais e podem ser usados para listar respectivamente os atributos e as operações definidas para a classe.

A Tabela 3 ilustra como é representado um diagrama de classe [8].

Tabela 3 – Exemplo de um Diagrama de Classe

Nome da Classe
atributo atributo: tipo de dado atributo: tipo de dado = valor inicial .....
operação operação (lista de argumentos): tipo de resultado .....

Na UML os diagramas geralmente não expõem os conteúdos totais de um modelo, mas somente algum subconjunto de detalhes do modelo.

2.4.2.1 Objeto

É uma unidade real ou abstrata, individualizada e identificável que modela um conceito presente na realidade humana ocupando espaço físico (mundo físico) ou lógico (memória), apresentando três características básicas: estado, identidade e comportamento [7] [8] [9].

2.4.2.2 Classe

É a representação de um conjunto de coisas reais ou abstratas que são reconhecidas como sendo do mesmo tipo por compartilhar as mesmas características de atributos, operações, relações e semântica [7] [8] [9].

2.4.2.3 Atributo

É a menor unidade que em si possui significância própria e inter-relacionada com o conceito de classe a qual pertence [7] [8] [9].

Um atributo é mostrado como uma seqüência de caracteres que pode ser analisada gramaticalmente nas várias propriedades de elemento cuja sintaxe padrão é:

Visibilidade NomeDoAtributo: TipoDeExpressao = ValorInicial {Propriedade}

Onde:

- **Visibilidade:** definida como pública, protegida ou privada (ver ANEXO B).
- **NomeDoAtributo:** é uma seqüência de caracteres de identificação começando tipicamente com letra minúscula.
- **TipoDeExpressao:** é uma especificação que depende da linguagem de programação e do tipo de implementação de um atributo.
- **ValorInicial:** é uma expressão que também depende da linguagem de programação utilizada para configurar o valor inicial a um objeto recentemente criado. O valor inicial é opcional (o sinal de igual também é omitido).
- **Propriedade:** possui características aplicáveis ao elemento mais opcional. Refere-se a um valor nomeado que denota uma característica de um elemento com impacto semântico.

2.4.2.4 Operação

Em UML, operação é um serviço resultante de um procedimento algorítmico. Há uma distinção importante entre operação e método: uma

operação é algo invocado por um objeto (procedimento de chamada) enquanto que um método é um corpo de procedimento [7] [8] [9].

As operações são executadas sempre que um objeto recebe uma mensagem de outro objeto.

A sintaxe padrão é:

Visibilidade NomeDaOperacao (Parametro): ExpressaoDeTipoDeRetorno {Propriedade}

Onde:

- **Visibilidade:** definida como pública, protegida ou privada (ver ANEXO B).
- **NomeDaOperacao:** seqüência de caracteres de identificação começando tipicamente com letra minúscula.
- **Parâmetro:** é uma lista de valores separada por vírgula de parâmetros formais usados para operações, mensagens e eventos.
- **ExpressaoDeTipoDeRetorno:** é uma especificação dependente de linguagem de programação sobre o tipo de implementação de valor retornado pela operação. Se o tipo de operação é omitido a operação não devolve um valor.
- **Propriedade:** indica valores de propriedade que se aplicam ao elemento.

2.4.3 Diagramas de Interação

São modelos que descrevem como grupos de objetos colaboram em algum comportamento [7] [8] [9].

Tipicamente, um diagrama de interação captura o comportamento de um único caso de uso. O diagrama mostra vários objetos e as mensagens que são passadas entre objetos em um caso de uso.

Existem dois tipos de diagramas de interação:

- Diagramas de Seqüência;
- Diagramas de Colaboração.

2.4.3.1 Diagrama de Seqüência

O diagrama de seqüência expõe o aspecto do modelo que enfatiza o comportamento dos objetos em um sistema, incluindo suas operações, interações, colaborações e histórias de estado em seqüência temporal de mensagem e representação explícita de ativação de operações [9] [10] [11].

Os objetos são desenhados como linhas verticais, as mensagens como linhas horizontais e a seqüência de mensagens é lida de cima para baixo na página.

No diagrama de seqüência um objeto é mostrado como uma caixa na parte superior de uma linha tracejada vertical como ilustra a Figura 7.

A linha vertical é chamada de linha de vida do objeto, representando a vida do objeto durante a interação.

Cada mensagem é representada por uma flecha entre as linhas de vida de dois objetos. Cada mensagem deve ser rotulada com no mínimo o nome da mensagem.

A linha de mensagem tracejada representada na Figura 7 chamada de retorno, representa o retorno de uma mensagem já enviada e não uma nova mensagem.

Uma mensagem que é mandada para si mesma é chamada de autochamada, ocorre quando uma flecha de mensagem volta para a mesma linha de vida.

Uma mensagem enviada entre parênteses, implica que, uma mensagem de condição está sendo enviada. Quando antes do parêntese ocorrer o símbolo "*", significa que uma mensagem de iteração esta sendo enviada.

O "**X**" no final da linha de vida do objeto 1 da Figura 7 representa a destruição deste objeto quando ele chegar no fim de sua linha de vida.

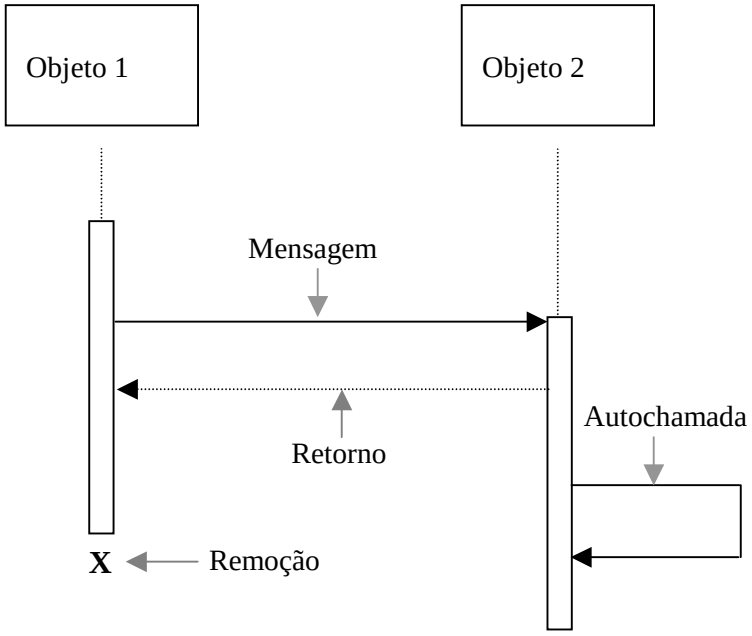

Figura 7: Diagrama de Seqüência

2.5 Componente de Comunicação Serial do Delphi 5

Como a ferramenta de desenvolvimento de sistemas Delphi 5 não possui componente de comunicação serial, é necessário que o mesmo seja desenvolvido.

Na internet é possível achar diversos componentes de comunicação, cada qual desenvolvido para uma comunicação serial especifica.

Dessa maneira foi necessário descobrir qual dos componentes disponíveis na internet seria o ideal para implementar a comunicação do computador com a Microbalança, em conseqüência do controle de fluxo de dados utilizado.

Três componentes foram testados: SerialPort, TSerial, VaComm, sendo escolhido o componente freeware VaComm da Variant-Software, como o mais apropriado para implementar a comunicação.

2.5.1 Componente VaComm da Variant-Software

O componente VaComm da Variant-Software pertence a um conjunto de nove componentes da paleta Variant Async32 [4] [15].

Todos o componentes dessa paleta são utilizados para estabelecer algum tipo de comunicação, desde comunicação serial até comunicação com servidores [15].

A Figura 8 ilustra o componente VaComm [5] [15]:

Figura 8: Componente VaComm.

Com este componente é possível configurar a taxa de transmissão de dados (baud rate), o número de bits de transmissão, os bits de parada, o bit de paridade, o número da porta de comunicação e o controle de fluxo de dados (principal vantagem deste componente) [15].

Na Microbalança as configurações da comunicação são as seguintes: taxa de transmissão 9600bps (br9600 forma do componente), 8 bits de transmissão (db8), bit de parada 1 (sb1) e o controle de fluxo RTS (fcRtsCts). O bit de paridade no caso não é usado, com isso utiliza-se à opção paNone [11] [15].

Os principais comandos utilizados são:

- Open – abre a porta de comunicação;
- Close – fecha a porta de comunicação;
- WriteText(const s: string): boolean – utilizado para enviar algum comando ou informação a Microbalança;
- ReadText(): string – utilizado para receber alguma informação da Microbalança;
- PurgeReadWrite – utilizado para limpar o buffer de entrada e de saída;
- RTSControl:= rtsEnable – realiza o chaveamento para receber os dados dos seis primeiros sensores;
- RTSControl:= rtsDisable – realiza o chaveamento para receber os dados dos seis últimos sensores.

2.6 Microbalança de Quartzo ou "Narizes Eletrônicos"

A Microbalança de Quartzo ou Nariz Eletrônico é utilizada para detecção e padronização de aromas ou padrões olfativos [12] [13] [14].

Aqui os odores são captados por sensores químicos, semelhante ao que ocorre com o nariz humano. Posteriormente os sinais obtidos pelos sensores são enviados para um órgão reconhecedor, o cérebro no caso do ser humano, e para o computador nos narizes eletrônicos [13].

Através de desenvolvidos sistemas estatísticos de análise de dados, as informações obtidas dos aparelhos podem ser interpretadas e usadas como padrões de diferenciação olfativa dos produtos submetidos à avaliação pelos narizes eletrônicos [13].

A metodologia empregada no desenvolvimento de um nariz eletrônico consiste na montagem de sensores piezoelétricos em paralelo acoplados a uma interface de aquisição de dados para um microcomputador. Os sensores

piezoelétricos utilizados são placas de quartzo oscilante recobertas com substâncias orgânicas, que são responsáveis pela especificidade da resposta obtida por reconhecimento molecular.

Os dados coletados de cada sensor são tratados por programas baseados em redes neurais. Este tipo de sistema é um dos mais modernos e apropriados ao estabelecimento e reconhecimento de padrões envolvendo um grande número de parâmetros e experimentos.Dessa maneira pode-se obter registros quantitativos para os diferentes tipos de perfis olfativos do café.

O nome Microbalança de Quartzo advém do fato deste tipo de arranjo de sensores, e em referência à sua alta sensibilidade (cerca de 10^{-12} g).

As Microbalanças de Quartzo são montadas no sistema BAW, com freqüências de trabalho variando entre 10 a 30 MHz [12]. Cada sensor químico é composto de uma fina fatia ou disco de cristal que tem fixado às suas faces dois eletrodos, normalmente de ouro ou prata revestida com ouro. Entre os eletrodos é aplicada uma diferença de potencial que promove a formação de um campo elétrico oscilante, perpendicular à superfície do cristal [14]. Este campo por sua vez, promove a vibração do cristal a uma freqüência muito estável, equivalente à voltagem aplicada ao sistema [12].

3. PROPOSIÇÃO

A proposta deste trabalho é desenvolver um sistema ou software para computador IBM-PC, com a finalidade de coletar os dados medidos pelo arranjo dos sensores químicos da Microbalança de Quartzo.

A Figura 9 ilustra o conjunto computador e Microbalança de Quartzo.

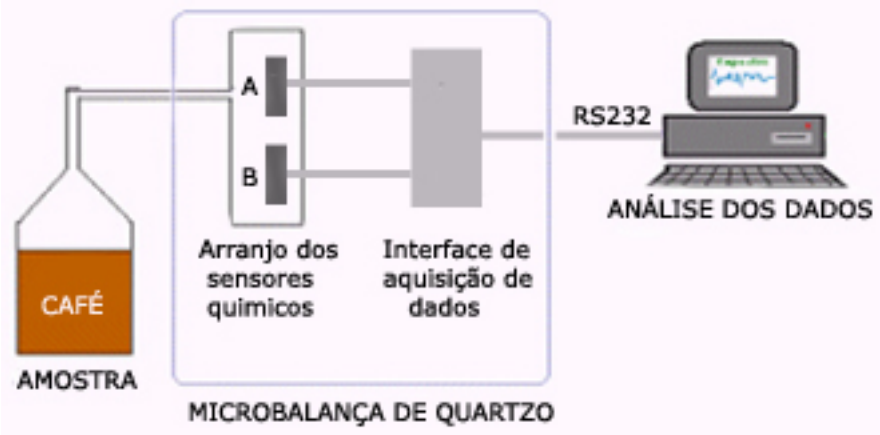

Figura 9: Diagrama Microbalança e Computador.

A comunicação entre o computador e a Microbalança de Quartzo será realizada via porta serial (RS232) através de um cabo db-9 macho de nove pinos.

Todos os dados medidos pela Microbalança serão visualizados através da interface gráfica do sistema. Também será possível visualizar toda a coleta dos dados em formato gráfico, tanto em tempo de execução ou através de uma coleta que já foi salva.

Será função do sistema controlar a recepção dos dados medidos pela Microbalança. Os dados coletados serão salvos em dois tipos de formatos de arquivo: o do sistema que será (*mbq) e o formato (*dat) que poderá ser utilizado em outros aplicativos.

Os dados salvos no formato (*mbq) poderão ser visualizados pelo sistema quando o mesmo estiver operando em um modo desconectado da Microbalança.

4. MATERIAIS E MÉTODOS

O Sistema de Coleta de Dados de uma Microbalança de Quartzo foi desenvolvido com bases nos princípios da UML de Jacobson, Booch e Rumbaugh.

Para a modelagem foram utilizados três diagramas da UML:

- Diagramas de Caso de Uso
- Diagramas de Classe
- Diagramas de Seqüência

A modelagem foi feita utilizando a ferramenta Rational Rose e o software implementado com a ferramenta Delphi 5.

Neste capitulo será abortado os diagramas utilizados na modelagem e os procedimentos utilizados em Delphi 5 para a implementação do sistema.

4.1 Diagrama de Caso de Uso

Através da descrição do sistema feita pelo Prof. Mauro dos Santos Carvalho, foi obtido o seguinte cenário que serviu de base para a elaboração do diagrama de caso de uso, ilustrado na Figura 10.

Cenário obtido:

*"O sistema deverá coletar todos os dados medidos pela Microbalança de Quartzo e visualizá-los em modo gráfico durante a coleta. Os dados coletados deverão ser salvos em dois formatos diferentes de arquivos: o formato (*dat) e formato (*mbq). Caso possa ocorrer algum problema com a Microbalança durante coleta é necessário que se faça à verificação das suas condições de funcionamento. Os dados salvos no formato (*mbq) poderão ser visualizados pelo sistema em modo gráfico quando o mesmo não estiver conectado com a Microbalança"*.

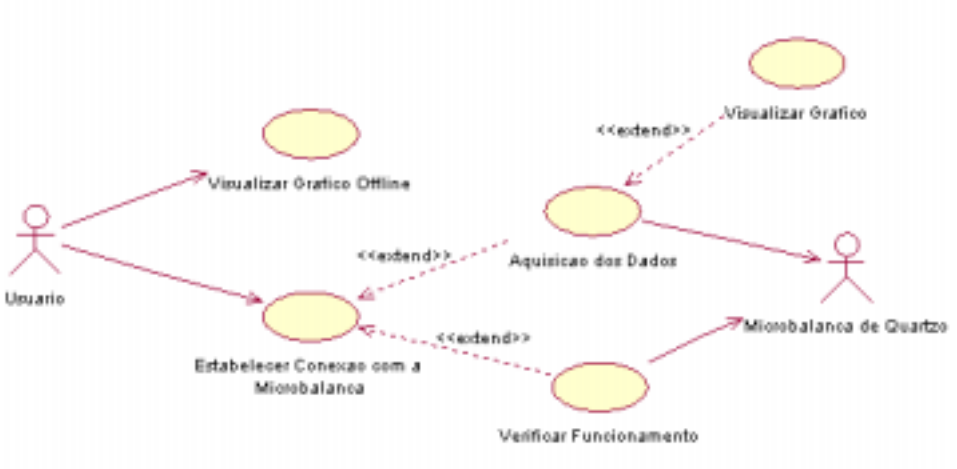

Figura 10: Diagrama de caso de uso do sistema.

Através deste diagrama foram obtidos dois atores e cinco casos de uso, tendo três casos de uso estendidos.

Onde:

- **Atores:** Usuário e Microbalança de Quartzo;
- **Casos de Uso:** Visualizar Grafico Offline, Estabelecer Conexao com a Microbalanca, Aquisicao dos Dados, Verificar Funcionamento e Visualizar Grafico.

4.1.1 Descrevendo os Casos de Uso

A descrição do processo dos casos de uso foi baseada no livro do Larman [9], onde é apresentado:

- Identificação do caso de uso;
- Pré-condições;
- Seqüência típica de eventos que irão se suceder;
- Seqüência alternativa que poderá ocorrer.

As Tabelas de 4 a 8 mostram a descrição dos casos de uso.

Tabela 4 – Caso de Uso Visualizar Grafico Offline

Identificação
Caso de Uso: Visualizar Grafico Offline. **Atores:** Usuário. **Finalidade:** Visualizar em modo gráfico os dados salvos no formato (*mbq). **Visão Geral:** O usuário ao iniciar o sistema poderá visualizar gráficos de arquivos que foram salvos no formato (*mbq).

Seqüência Típica de Eventos	
Ação do Ator	**Resposta do Sistema**
1. O usuário escolhe um arquivo salvo no formato (*mbq) e seleciona-o para abrir.	2. O sistema carrega o arquivo escolhido pelo usuário.
3. O usuário solicita visualização gráfica do arquivo salvo.	4. O sistema visualiza o gráfico para o usuário.

Seqüência Alternativa
Linha 2: não existe nenhum arquivo salvo no formato (*mbq), impedindo que a visualização seja realizada.

Tabela 5 – Caso de Uso Estabelecer Conexao com a Microbalanca

Identificação
Caso de Uso: Estabelecer Conexao com a Microbalanca. **Atores:** Usuário. **Finalidade:** Conectar o sistema com Microbalança. **Visão Geral:** O usuário configura os parâmetros e solicita a de conexão.

Pré-Condições
1. A Microbalança estar ligada e conectada ao computador via porta serial.

Seqüência Típica de Eventos	
Ação do Ator	**Resposta do Sistema**
1. O usuário faz a solicitação para estabelecer a comunicação	2. O sistema solicita ao usuário configurar os parâmetros de conexão.
3. O usuário estabelece os parâmetros para conexão.	
4. O usuário solicita a conexão do sistema com a Microbalança.	5. O sistema estabelece a conexão com a Microbalança.

Seqüência Alternativa
Linha 4: o usuário passa algum parâmetro errado para a conexão. **Linha 4:** a Microbalança pode estar desligada.

Tabela 6 – Caso de Uso Verificar Funcionamento

Identificação
Caso de Uso: Verificar Funcionamento. **Atores:** Usuário, Microbalança. **Finalidade:** Verificar as condições de funcionamento da Microbalança. **Visão Geral:** O usuário solicita saber quais as condições atuais de funcionamento da Microbalança para poder usa-la.

Pré-Condições
1. A conexão entre a Microbalança e o sistema deve ter ocorrido com sucesso

Seqüência Típica de Eventos	
Ação do Ator	**Resposta do Sistema**
1. O usuário solicita verificar as condições de funcionamento da Microbalança.	2. O sistema devolve o resultado após verificar a Microbalança.

Seqüência Alternativa
Linha 1: a conexão pode .ter sido desfeita.

Tabela 7 – Caso de Uso Aquisicao dos Dados

Identificação
Caso de Uso: Aquisicao dos Dados. **Atores:** Usuário, Microbalança. **Finalidade:** Coletar os dados medidos pela Microbalança. **Visão Geral:** O usuário solicita o inicio da aquisição dos dados.

Pré-Condições
1. A conexão entre a Microbalança e o sistema deve ter ocorrido com sucesso. 2. O arquivo onde os dados serão salvos tenha sido criado. 3. Ter configurado o tempo de coleta dos dados.

Seqüência Típica de Eventos	
Ação do Ator	**Resposta do Sistema**
1. O usuário solicita a aquisição dos dados. 3. O usuário encerra a aquisição dos dados.	2. O sistema realiza a coleta dos dados com o intervalo de tempo estabelecido pelo usuário, guardando os dados coletados no arquivo que foi criado.

Seqüência Alternativa
Linha 1: a conexão pode ter sido desfeita.

Tabela 8 – Caso de Uso Visualizar Grafico.

Identificação
Caso de Uso: Visualizar Gráfico. **Atores:** Usuário. **Finalidade:** Visualizar em modo gráfico os dados coletados. **Visão Geral:** O usuário solicita a visualização dos dados coletados em modo gráfico.

Pré-Condições
1. Estar realizando a aquisição dos dados.

Seqüência Típica de Eventos	
Ação do Ator	**Resposta do Sistema**
1. O usuário solicita a visualização dos dados coletados em modo gráfico.	2. O sistema desenha o gráfico para o usuário usando os dados coletados.

4.2 Diagrama de Classe

A modelagem deste sistema resultou no desenvolvimento de três classes:

- **TFormPrincipal:** a classe principal do sistema;
- **TFormConfiguracao:** classe que controla a conexão do sistema com a Microbalança;
- **TVAComm:** classe do componente utilizado para conexão.

4.2.1 Classe TFormPrincipal

É a classe principal do sistema, que interage com o usuário. Ela possui os seguintes atributos e operações, como ilustrado na Tabela 9.

Tabela 9 – Classe TFormPrincipal

TFormPrincipal
+ FormConfiguracao: TFormConfiguracao
- ItemGraficoOfflineClick() - BotaoConectarClick() - BotaoDisconectarClick() - BotaoVerificarClick() - BotaoIniciarClick() - BotaoPararClick() + TimerMedidasTimer() + MontaGraficoOffline(linhas:integer) + StatusBalanca(dado_recebido:string) + EnableMedidas(dado:string) + DisableMedidas(dado:string) + SalvarDat(caminho:string) + SalvarMbq(caminho:string) + MontaGrafico(tempo: real, linha: inteiro) + TempoTotal(tempo: string)

4.2.1.1 Atributos:

FormConfiguracao: objeto da classe TFormConfiguracao.

4.2.1.2 Operações:

ItemGraficoOfflineClick(): visualiza a interface do gráfico offline.

BotaoConectarClick(): visualiza a interface onde o usuário estabelece os parâmetros de conexão.

BotaoDisconectarClick(): encerra a conexão Microbalança com o sistema.

BotaoVerificarClick(): realiza a verificação das condições de funcionamento da Microbalança.

BotaoIniciarClick(): inicia a aquisição dos dados.

BotaoPararClick(): encerra a aquisição dos dados.

TimerMedidasTimer(): rotina para realizar a aquisição dos dados a cada intervalo de tempo.

MontaGraficoOffline(linhas:integer): traça o gráfico dos dados que foram salvos no formato (*mbq).

StatusBalanca(dado_recebido:string): faz a análise das condições de funcionamento da Microbalança.

EnableMedidas(dado:string): realiza o tratamento dos dados dos seis primeiros sensores.

DisableMedidas(dado:string): realiza o tratamento dos dados dos seis últimos sensores.

SalvarDat(caminho:string): salva os dados coletados no formato (*dat).

SalvarMbq(caminho:string) : salva os dados coletados no formato (*mbq).

MontaGrafico(tempo: real, linha: inteiro): traça o gráfico dos dados que estão sendo coletados.

TempoTotal(tempo: string): retorna o tempo total da coleta.

4.2.2 Classe TFormConfiguracao

É a classe que realiza a conexão do sistema com a Microbalança. Ela possui os seguintes atributos e operações, ilustrados na Tabela 10.

Tabela 10 – Classe TFormConfiguracao

TFormConfiguracao
+ VaComm1: TVAComm
- BotaoEstabelecerConexaoConectarClick() - FluxoControle(chave:boolean) - AquisicaoDados(mensagem:string): string - FormShowModal()

4.2.2.1 Atributos

VaComm1: objeto da classe TVAComm.

4.2.2.2 Operações:

BotaoEstabelecerConexao(): estabelece conexão da Microbalança como o computador.

FluxoControle(chave:boolean): realiza a troca do grupo de sensores (chaveamento) da Microbalança para a coleta dos dados. Quando chave for “true” o atributo RTSControl da classe TVAComm recebe rtsEnable, e os dados coletados são do grupo dos seis primeiros sensores. Quando chave for “false” o atributo RTSControl da classe TVAComm recebe rtsDisable, e os dados coletados são do grupo dos seis últimos sensores.

AquisicaoDados(mensagem:string):string: esta função tem como objetivo realizar a aquisição dos dados da Microbalança.

FormShowModal: utilizado na classe principal para visualizar a interface da classe TFormConfiguracao.

4.2.3 Classe TVAComm

É a classe do componente de comunicação serial. No desenvolvimento do sistema foram utilizados os seguintes atributos e operações, ilustrados na Tabela 11.

Tabela 11 – Classe TVAComm

TVAComm
+ Baudrate: TvaBaudrate = br9600 {br110, br300, br600, br1200, br2400, br4800, br9600, br14400, br19200, br38400, br56000, br57600, br115200, br128000, br256000} + Stopbits: TVaStopbits = sb1 {sb1, sb15, sb2} + Parity: TVaParity = paNone {paNone, paOdd, paEven, paMark, paSpace} + PortNum: Integer + FlowControl: TVaFlowControl = fcRtsCts {fcNone, fcRtsCts} + Databits: TVaDatabits = db8 {db4, db5, db6, db7, db8} + RTSControl: TVaRTSControl = rtsEnable {rtsEnable, rstDisable}
- Open() - PurgeReadWrite() - Close() - WriteTex(const s: string):Boolean - ReadText(): string

4.2.3.1 Atributos

Baudrate: utilizado para configurar a taxa de transmissão de dados.

Stopbits: bits de parada.

Parity: bits de paridade.

PortNum: número da porta de conexão que será utilizada.

FlowControl: realiza o controle do fluxo de dados através do chaveamento dos sensores.

Databits: número de bits.

RTSControl: realiza o chaveamento dos sensores na Microbalança, rtsEnable coleta os dados do sensor 1 ao 6, rtsDisable do sensor 7 ao 12.

4.2.3.2 Operações

Open(): realiza a conexão entre a Microbalança e o computador.

PurgeReadWrite(): limpa o buffer de entrada e saída.

Close(): encerra a conexão entre a .Microbalança e o computador.

WriteText(const s: string): Boolean: envia informações a Microbalança.

ReadText(): string: recebe informações da Microbalança.

4.2.4 Relacionando as Classes

Para a concretização do sistema, é necessário realizar um relacionamento entre as classes criadas.

A melhor forma de relacioná-las é utilizando a relação de agregação, que por teoria, agregação é uma forma especial de associação utilizada para mostrar que um tipo de objeto é composto, pelos menos em parte, de outro em uma relação de todo/parte [8].

Dessa forma a classe TVAComm é parte da classe TFormConfiguracao, e ambas, parte da classe TFormPrincipal. A relação entre elas é sempre de um para um.

A relação de agregação é implementada da seguinte forma: um diamante vazio é anexado próximo a classe que representa o todo no relacionamento [8].

A Figura 11 ilustra o relacionamento entre as classes:

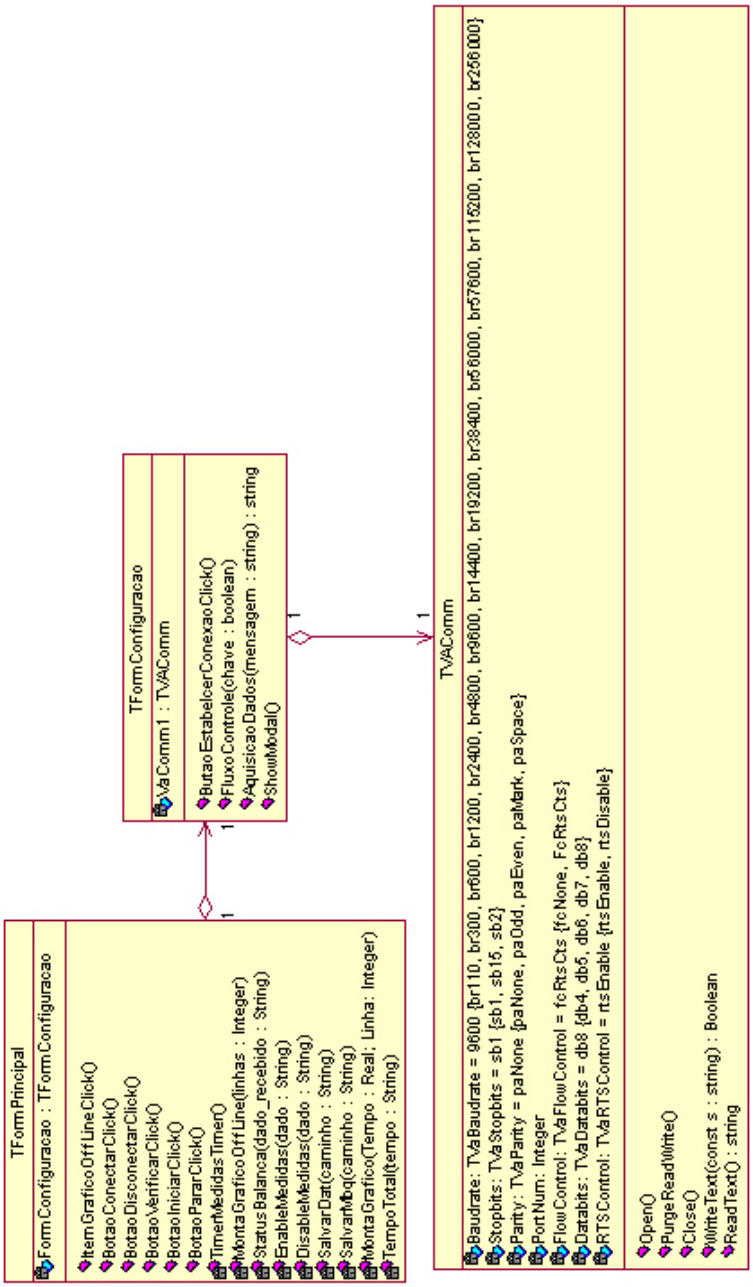

Figura 11: Diagrama de classe.

4.3 Diagrama de Seqüência

Os diagramas de seqüência têm como objetivos identificar as operações e eventos do sistema. São desenvolvidos com base nos casos de uso que foram criados [9].

Com base no diagrama de caso de uso, foram desenvolvidos os seguintes diagramas de seqüência:

- Visualizar Gráfico Offline;
- Estabelecer Conexão com a Microbalança;
- Verificar Funcionamento da Microbalança;
- Aquisição dos Dados;
- TimerMedidasTimer;
- Coletando os dados;
- Encerrar Aquisição;
- Desconectar o Sistema da Microbalança.

4.3.1 Visualizar Gráfico Offline

Inicialmente o usuário envia uma mensagem a classe TFormPrincipal com operação ItemGraficoOffline, e esta faz uma autochamada com a operação MontaGraficoOffline(Integer), como ilustra a Figura 12:

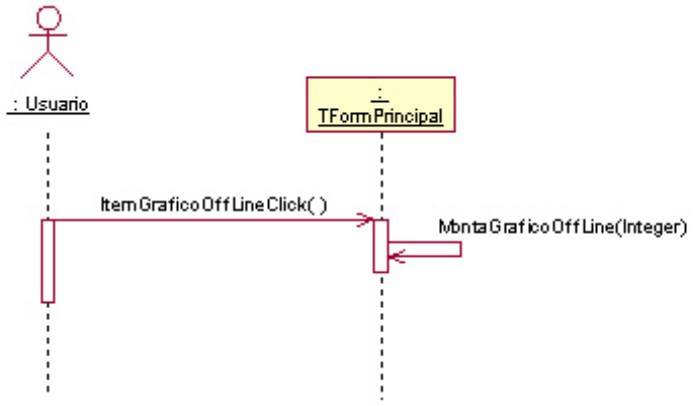

Figura 12: Visualizar gráfico offline.

4.3.2 Estabelecer Conexão com a Microbalança

Para conectar-se, o usuário envia uma mensagem a classe TFormPrincipal através da operação BotaoConectarClick(). Em seguida a classe TFormPrincipal envia uma mensagem a classe TFormConfiguracao através da operação ShowModal(), e esta solicita a conexão enviando uma mensagem a classe TVAComm através da operação Open(), ilustrado na Figura 13.

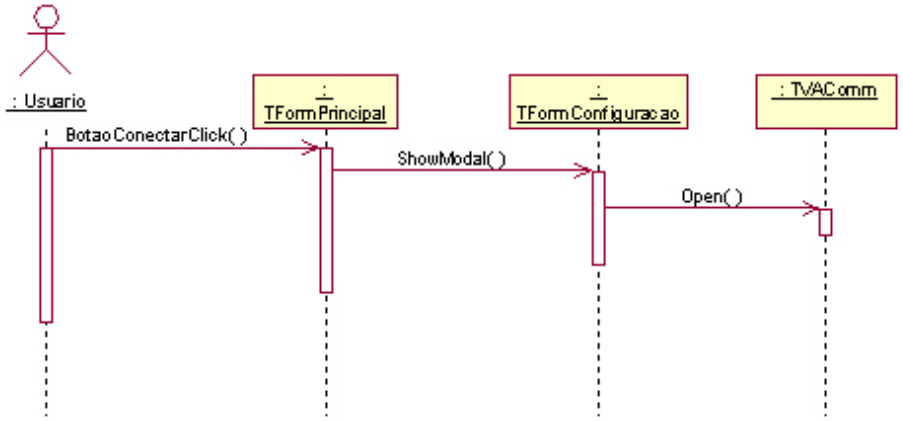

Figura 13: Estabelecer conexão com a Microbalança.

4.3.3 Verificar Funcionamento da Microbalança

Para realizar a verificação das condições de funcionamento da Microbalança, o usuário inicialmente envia uma mensagem a classe TFormPrincipal através da operação BotaoVerificarClick().

Solicitada à verificação, a classe TFormPrincipal envia uma mensagem a classe TVAComm com a operação WriteText(string) solicitando um pedido de verificação.

Após solicitar o pedido, a classe TFormPrincipal envia uma mensagem a classe TVAComm com a operação ReadText() para receber os dados solicitados.

Recebendo os dados, a classe TFormPrincipal faz uma autochamada com a operação StatusBalanca(string) para analisar os dados recebidos.

A Figura 14 ilustra o processo descrito.

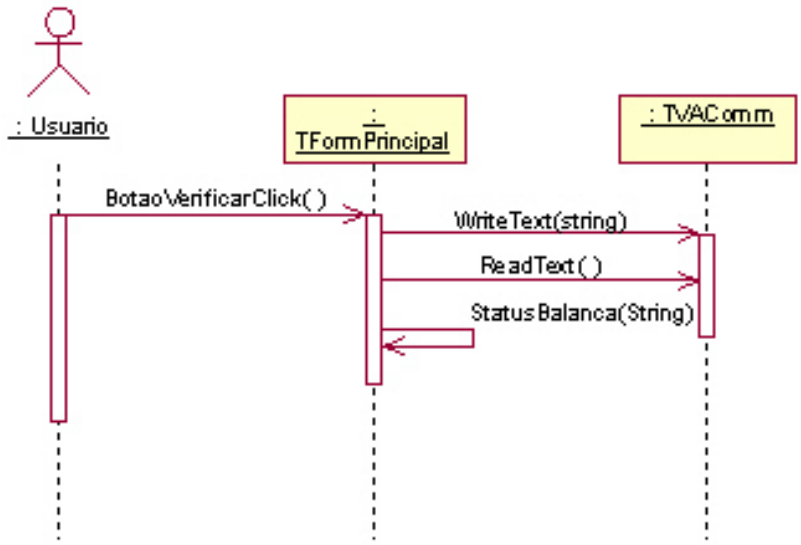

Figura 14: Verificar funcionamento da Microbalança.

4.3.4 Aquisição dos Dados

Para iniciar o processo de aquisição dos dados, o usuário, inicialmente envia uma mensagem a classe TFormPrincipal através do BotaoIniciarClick(). Esta envia uma mensagem a classe TFormConfiguracao através da operação AquisicaoDados(string) para coletar os dados do grupo dos seis primeiros sensores.

Coletado os dados, a classe TFormPrincipal faz uma autochamada utilizando a operação EnableMedidas(string).

A próxima coleta será dos seis últimos sensores, para isso, a classe TFormPrincipal envia uma mensagem a classe TFormConfiguracao com a operação FluxoControle(boolean).

Com o chaveamento trocado, a classe TFormPrincipal solicita uma nova aquisição de dados a classe TFormConfiguracao através da operação AquisicaoDados(string) para coletar os dados do grupo dos seis últimos sensores.

Coletado os dados, a classe TFormPrincipal faz uma autochamada utilizando a operação DisableMedidas(string).

Para que a aquisição dos dados continue em intervalos constantes de tempo, a classe TFormPrincipal faz uma autochamada com a operação TimerMedidasTimer().

A Figura 15 ilustra o processo descrito acima.

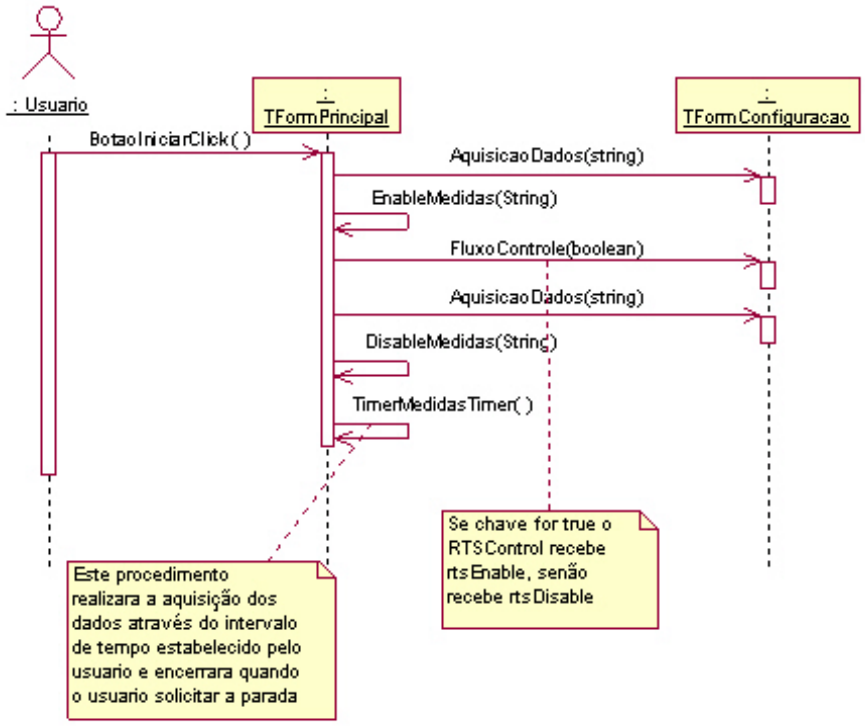

Figura 15: Aquisição dos dados.

4.3.5 TimerMedidasTimer

Ilustra a seqüência de eventos da operação TimerMedidasTimer() utilizado no diagrama de Aquisição dos Dados.

Como no diagrama de Aquisição dos Dados o chaveamento terminou no grupo dos seis últimos sensores, o primeiro evento deste diagrama é mudar o chaveamento para os seis primeiros sensores. Este processo é semelhante ao do diagrama de Aquisição dos Dados.

Os próximos eventos de aquisição são semelhantes aos utilizado no diagrama de Aquisição.

Terminada a aquisição de todos os dados, é necessário limpar o buffer de entrada e saída. Para realizar este evento, a classe TFormPrincipal envia uma mensagem a classe TVaComm através da operação PurgeReadWrite().

Os próximos eventos serão autochamadas realizadas na classe TFormPrincipal.

A primeira autochamada será o SalvarDat(string), que salva os dados coletados no formato (*dat) e logo em seguida o SalvarMbq(string), que salva os dados coletados no formato (*mbq).

O gráfico dos dados coletados, é traçado através da operação MontaGrafico(real, integer) e o tempo total da coleta, calculado pelo TempoTotal(string).

A Figura 16 ilustra o processo descrito:

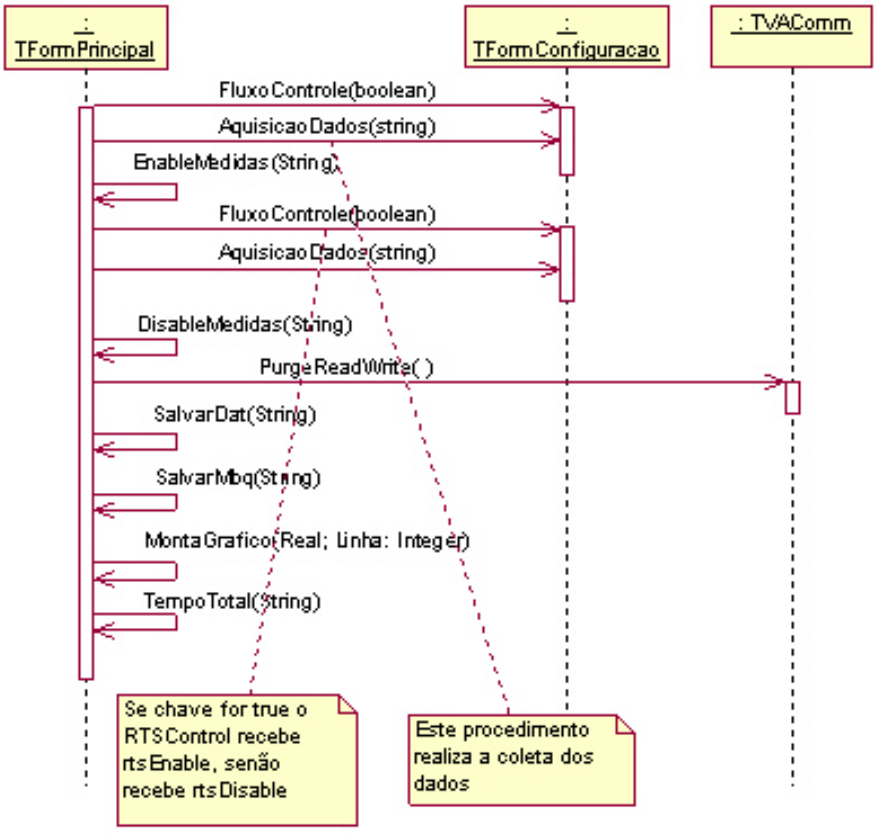

Figura 16: TimerMedidasTimer.

4.3.6 Coletando os Dados

Este diagrama ilustra a seqüência de eventos utilizados pela operação AquisicaoDados(string).

A classe TFormPrincipal envia uma mensagem a classe TVAComm com a operação WriteText(string) solicitando a aquisição dos dados.

Logo em seguida, outra mensagem é enviada a classe TVAComm com a operação ReadText() para receber os dados solicitados.

A Figura 17 ilustra o processo descrito:

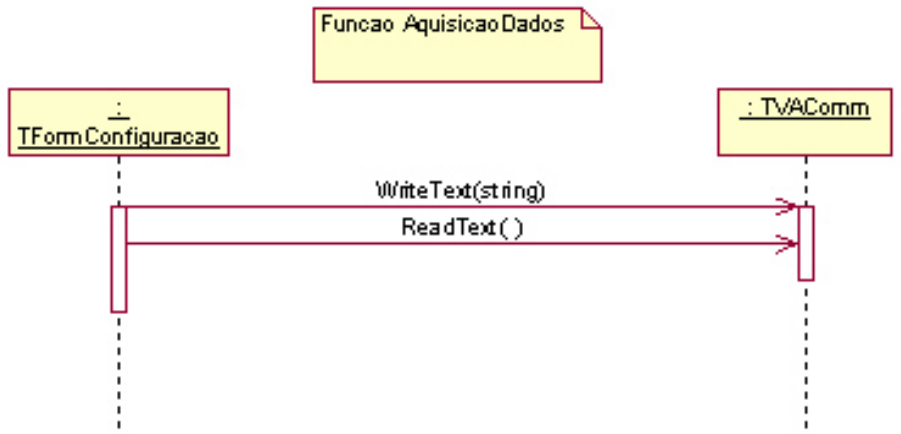

Figura 17: Coletando os dados.

4.3.7 Encerrar Aquisição

Para encerrar a aquisição dos dados, o usuário envia uma mensagem a classe TFormPrincipal com a operação BotaoPararClick(), que encerra a operação TimerMedidasTimer().

Como o chaveamento termina voltado para o grupo dos seis últimos sensores, a classe TFormPrincipal envia uma mensagem a classe TFormConfiguracao com a operação FluxoControle(boolean).

O próximo evento é limpar o buffer de entrada e saída, para isso, a classe TFormPrincipal envia uma mensagem a classe TVAComm com a operação PurgeReadWrite().

A Figura 18 ilustra o processo descrito:

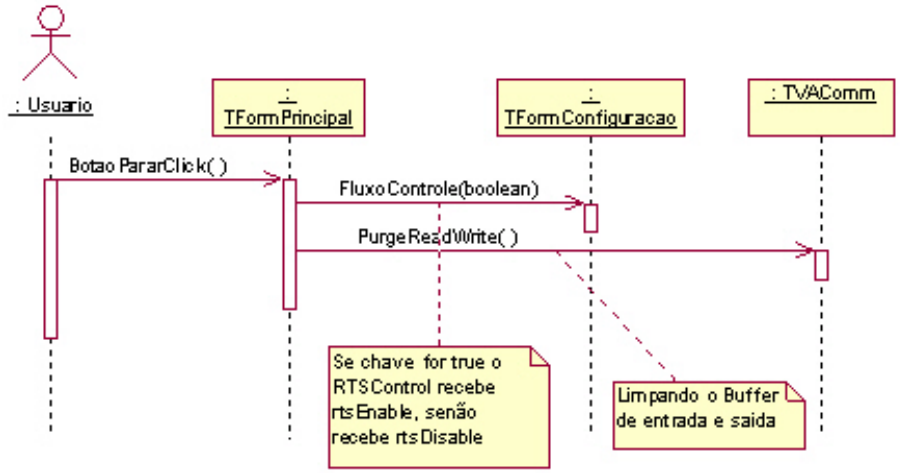

Figura 18: Encerrar aquisição.

4.3.8 Desconectar o Sistema da Microbalança

Para encerrar a conexão, o usuário envia uma mensagem a classe TFormPrincipal com a operação BotaoDisconectarClick() e em seguida essa envia uma mensagem a classe TVAComm com a operação Close() para finalizar a conexão do sistema com a Microbalança, como ilustra a Figura 19.

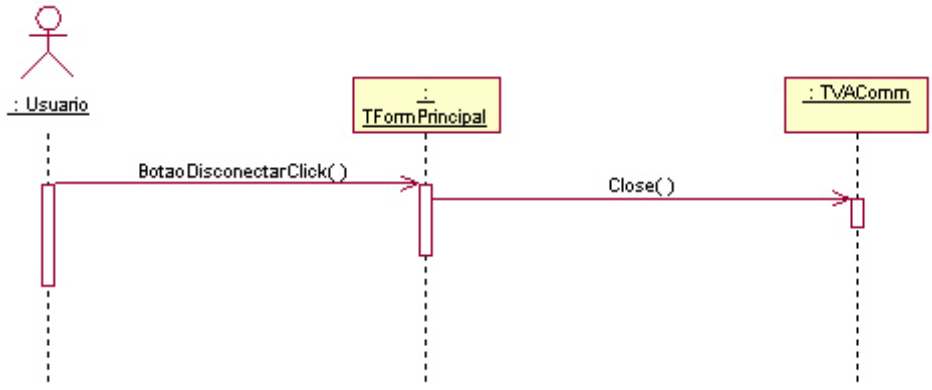

Figura 19: Desconectar o sistema da Microbalança.

4.4 Arquivos (*dat) e (*mbq)

Toda coleta de dado realizada pelo sistema é salva sempre que uma nova aquisição é realizada.

O nome do arquivo que salva a coleta é definido pelo usuário antes do inicio da aquisição.

Os dados coletados são salvos em dois formatos diferentes de arquivos: o formato (*dat) que é compatível a outros aplicativos como o Excel, e o formato (*mbq) que é o formato exclusivo do sistema.

Os dois arquivos são salvos sempre com o mesmo nome.

4.4.1 Formato (*dat)

O formato (*dat) foi criado com o intuído de ser utilizados em outros aplicativos como o caso do Excel. O nome (*dat) foi definido pelo prof. Mauro dos Santos Carvalho.

Como os dados coletados são salvos temporariamente em uma tabela, como ilustra a Figura 20, a melhor forma para salvar os dados, já que eles podem ser utilizados em outros aplicativos, é salvar a linha inteira da tabela em uma linha do arquivo.

Tempo e Frequências

Tempo	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7	Sensor 8	Sensor 9	Sensor 10	Sensor 11	Sensor 12
9,5	9985529	9988179	9943760	9991587	9942041	9973207	9870161	9979827	9868082	9908119	9962470	9981151
9,55	9985529	9988178	9943760	9991587	9942041	9973207	9870161	9979827	9868082	9908119	9962469	9981152
9,6	9985529	9988179	9943760	9991586	9942041	9973208	9870161	9979826	9868082	9908119	9962469	9981151
9,65	9985528	9988179	9943760	9991587	9942041	9973208	9870161	9979827	9868081	9908119	9962468	9981151
9,7	9985528	9988179	9943760	9991587	9942041	9973208	9870160	9979827	9868082	9908119	9962469	9981151
9,75	9985529	9988176	9943760	9991586	9942040	9973208	9870160	9979827	9868082	9908119	9962469	9981151
9,8	9985529	9988176	9943760	9991587	9942041	9973208	9870160	9979827	9868081	9908118	9962469	9981151
9,85	9985529	9988175	9943760	9991587	9942041	9973208	9870159	9979827	9868081	9908118	9962469	9981151

Figura 20: Tabela onde os dados são armazenados temporariamente.

Por exemplo, suponhamos que a sistema receba os seguintes dados: 001 002 003 004 005 006 no tempo zero de coleta e estes dados no tempo três (segundos) de coleta: 011 012 013 014 015 016.

O formato em que o arquivo será salvo é apresentado na Tabela 12:

Tabela 12 – Formato do arquivo (*dat)

Tempo	Dados Coletados
0	001 002 003 004 005 006
3	011 012 013 014 015 016

4.4.2 Formato (*mbq)

O formato (*mbq) foi criado com o intuído de ser utilizado para visualizar os dados salvos no próprio sistema. O nome (*mbq) foi definido pelo prof. Mauro dos Santos Carvalho.

Como os dados coletados são armazenados temporariamente em uma tabela como ilustra a Figura 20, e com o intuito de usar esta mesma tabela para visualizar os dados salvos, basta salvar o número de linhas da tabela menos um na primeira linha do arquivo, e cada dado da tabela em uma linha do arquivo (tempo e dados coletados).

Por exemplo, suponhamos que a sistema receba os seguintes dados: 001 002 003 004 005 006 no tempo zero de coleta e estes dados no tempo três (segundos) de coleta: 011 012 013 014 015 016.

Dessa maneira teríamos um arquivo com os seguintes dados:

3 – numero de linhas menos um

0 – tempo

001

002

003

004

005

006

3 – tempo

011

012

013

014

015

016

4.5 Método para Traçar o Gráfico

Para uma melhor compreensão dos dados coletados, é necessário que os mesmos sejam visualizados em modo um gráfico.

A cada intervalo de tempo coleta-se doze dados da Microbalança, sendo assim, para que haja uma melhor interpretação química dos dados é necessário que eles sejam tratados separadamente em gráfico. Dessa forma cada dado coletado tem uma linha no gráfico, sendo este gráfico traçado com dados adquiridos (freqüência de oscilação dos sensores) em função de seu tempo de coleta, como ilustra a Figura 21.

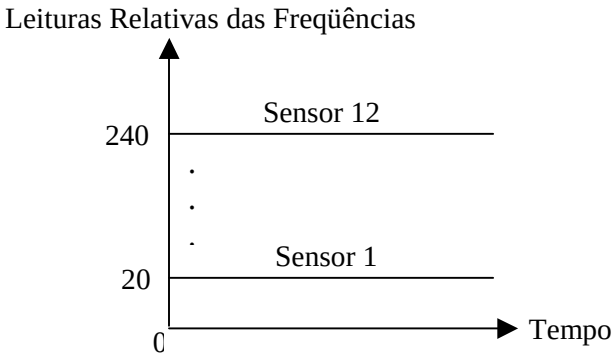

Figura 21: Gráfico dos dados coletados.

4.5.1 Fórmula para Traçar o Gráfico

Cada dado coletado é traçado em uma linha separada no gráfico, como ilustra a Figura 21. A seguir será descrito como cada linha é traçada.

Toda coleta inicia-se no tempo zero, assim cada um dos doze dados coletados são armazenados em uma variável chamada de “FA”. Os demais dados coletados a cada intervalo de tempo, são armazenados em uma variável temporária chamada de “FP”.

Cada dado coletado possui um índice. O dado do sensor 1 possui índice 20, o dado do sensor 2 possui índice 40 indo até o sensor 12 que possui índice 240. Nota-se que os índices variam de 20 em 20.

Estes índices é que irão delimitar o gráfico não deixando que a linha de um dado inicie no mesmo local de uma outra, como é ilustrado na Figura 21.

Para traçar o gráfico basta subtrair “FA” por “FP” e somar ao índice do sensor. Com isso teremos:

- Sensor 1: FA – FP + 20;
- Sensor 2: FA – FP + 40;
- Sensor 3: FA – FP + 60;
- Sensor 4: FA – FP + 80;
- Sensor 5: FA – FP + 100;
- Sensor 6: FA – FP + 120;
- Sensor 7: FA – FP + 140;
- Sensor 8: FA – FP + 160;
- Sensor 9: FA – FP + 180;
- Sensor 10: FA – FP + 200;
- Sensor 11: FA – FP + 220;
- Sensor 12: FA – FP + 240.

Nota-se que até o primeiro intervalo de tempo todas as linhas do gráfico serão uma reta porque o “FA” será igual ao “FP”. É importante lembrar que

somente a variável “FP” varia a cada intervalo de tempo e que o gráfico é traçado em tempo real.

4.6 Principais Comandos da Microbalança

Dois comandos são primordiais para trabalhar com a Microbalança [11]:

- “? GDA”
- “? GST”

Lembrando que todo comando enviado a Microbalança deve ser terminado com os caracteres CR (carrier return, código ASCII 13) e LF (line feed, código ASCII 10), conforme a tabela do ANEXO A.

4.6.1 Comando “? GDA”

Este comando tem como função solicitar a aquisição dos dados a Microbalança [11].

Após o envio do comando “? GDA”, a Microbalança retorna ao sistema a seguinte string, por exemplo:

!		GDA			1234567			1234567			1234567			1234567			1234567			

Note que após a letra A e cada um dos dado, existem dois espaços em branco, isto porque, os dados recebidos poderão ter sete ou oito caracteres.

O primeiro espaço em branco é utilizado como caracter de separação. O segundo é utilizado para incluir o oitavo carcter caso ele apareça.

Os procedimentos que fazem o tratamento destes dados são:

- EnableMedidas(dado: string) – (ver ANEXO C)
- DisableMedidas(dado:string)

4.6.2 Comando “? GST”

Este comando tem como função solicitar a Microbalança, a verificação das suas condições de funcionamento [11].

Após o seu envio a Microbalança devolve a seguinte string:

“! GST XXXX”

Onde “XXXX” representa uma seqüência de números hexadecimais.

Esta seqüência de números deve ser convertida para binário, para que uma análise mais profunda possa ser feita.

O processo de análise inicia-se com o procedimento StatusBalanca(dado_recebido: string), onde dado_recebido será o “! GST XXXX” (ver ANEXO C).

Como o número “XXXX” deve ser convertido para binário, o próximo passo é chamar a função Conversao(palavra: string): string que devolve uma string com o número “XXXX”convertido para binário.

Por exemplo, caso seja recebido a string “! GST AB23”.

Com base na Tabela 1 teríamos:

- A = 1010
- B = 1011
- 2 = 0010
- 3 = 0011

Sendo formado o seguinte número: 1010101100100011.

Após converter o número “XXXX” para binário, deve-se converter a string para inteiro e realizar um somatório. Este processo é feito pela função Somador(palavra: string): integer, que devolve um inteiro.

Se o resultado da soma for zero quer dizer que a Microbalança esta OK, sem nenhum problema.

Se o resultado não for zero deve-se olhar quais bits estão em 1, porque serão neles que os problemas da Microbalança serão apontados. Lembrando que o bit menos significativo é o da direita.

Os problemas apresentados quando o bit for 1 são os seguintes:

- Bit 0 - Oscilador 1 não propriamente rodando ou não conectado;
- Bit 1 - Oscilador 2 não propriamente rodando ou não conectado;
- Bit 2 - Oscilador 3 não propriamente rodando ou não conectado;
- Bit 3 - Oscilador 4 não propriamente rodando ou não conectado;
- Bit 4 - Oscilador 5 não propriamente rodando ou não conectado;
- Bit 5 - Oscilador 6 não propriamente rodando ou não conectado;
- Bit 8 - Falha na configuração do FPGA;
- Bit 9 - Falha enquanto lê contadores;
- Bit 10 - ROM mal, erro de checksum;
- Bit 11 - RAM mal, erro de leitura/escrita.

Caso o número recebido seja este 1010101100100011, obteríamos as seguintes respostas:

- Oscilador 1 não propriamente rodando ou não conectado;
- Oscilador 2 não propriamente rodando ou não conectado;
- Oscilador 6 não propriamente rodando ou não conectado;
- Falha na configuração do FPGA;
- Falha enquanto lê contadores;
- RAM mal, erro de leitura/escrita.

4.7 Chaveamento dos Dados

Na Microbalança não é possível coletar todos os dados ao mesmo tempo. Os dados são coletados do sensor 1 ao 6 e depois do sensor 7 ao 12 [11].

Para que a coleta ocorra perfeitamente é necessário utilizar o controle de fluxo de dados fcRtsCts. Utilizando este controle é possível informar a Microbalança qual a seqüência de dados a ser coletada.

Este controle é feito pelo pino 7 do cabo serial RS232, que pode ser ativado e desativado executando o comando rtsEnable ou rtsDisable.

Quando o controle enviado pelo pino 7 for o rtsEnable, ele irá para a interface de comunicação da Microbalança com o computador, como ilustra a Figura 22. A interface enviará o sinal para o Multiplexador e ele interpretará que os dados solicitados são os do grupo A. Estes dados são enviados a interface de comunicação e passados ao computador através da porta serial.

Quando o controle enviar rtsDisable ocorrerá o mesmo processo só que enviando os dados do grupo B.

A configuração inicial do controle de fluxo de dados é rtsEnable.

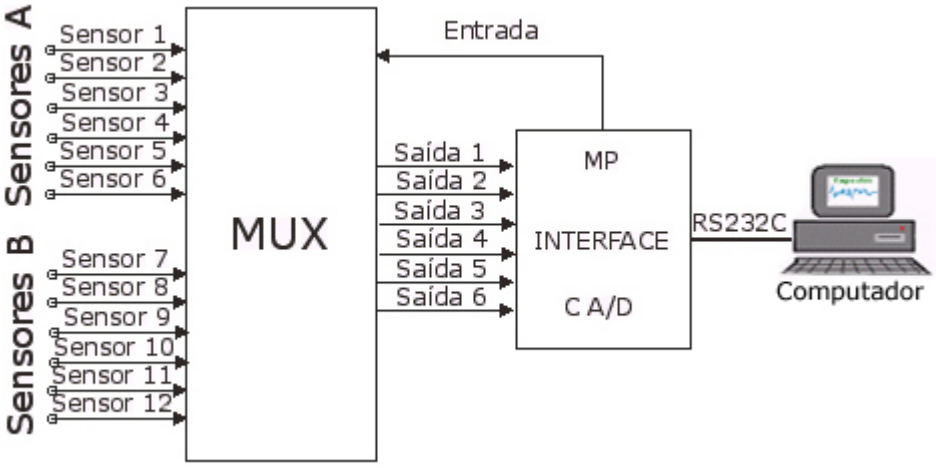

Figura 22: Chaveamento dos dados entre a Microbalança e o computador.

4.8 Aquisição dos Dados

A aquisição dos dados ocorre em intervalos de tempo determinado pelo usuário, dessa forma, a cada intervalo são coletados 12 dados da Microbalança.

A coleta inicia-se no tempo zero, utilizando a função AquisicaoDados(mensagem: string): string, que faz a aquisição dos dados do sensor 1 ao 6 (ver ANEXO C).

Depois, deve-se realizar o chaveamento para coletar os dados do sensor 7 ao 12, utilizando o procedimento FluxoControle(chave: boolean) com o parâmetro chave sendo falso. Quando o parâmetro chave for verdadeiro fará o chaveamento para rtsEnable.

Feito o chaveamento a função AquisicaoDados(mensagem: string): string, fará a aquisição dos dados do sensor 7 ao 12.

Terminado este processo um contador de tempo chamado Timer é acionado para que os dados sejam coletados em intervalos constantes de tempo.

O processo de coleta do Timer é semelhante ao processo de coleta inicial, porém antes de coletar os dados, deve-se utilizar o procedimento FluxoControle(chave: boolean) com o parâmetro chave sendo verdadeiro.

Depois de realizado o processo é necessário que o buffer de entrada e saída seja limpo para que ele não estoure e o sistema não acesse endereços inválidos de memória. O buffer é limpo com o procedimento PurgeReadWrite do componente VaComm.

Os dados ainda serão salvos nos formatos (*dat) e (*mbq) através dos procedimentos SalvarDat(caminho: string) e SalvarMbq(caminho: string), e o gráfico dos dados coletados traçado através do procedimento MontaGrafico(Tempo: real; Linha: integer).

A aquisição dos dados só encerra quando o usuário desejar parar. Ao parar a coleta o buffer é limpo e o controle de fluxo volta para rtsEnable.

5. RESULTADOS E DISCUSSÕES

Todos os objetivos almejados para o desenvolvimento deste sistema foram alcançados.

Os dados são coletados de acordo com o intervalo de tempo estabelecido pelo usuário, são salvos nos dois formatos de arquivos definidos, podem ser visualizados em formato gráfico em tempo de execução ou após serem salvos.

5.1 Arquivos Salvos

Todos os arquivos coletados são salvos independente ou não da vontade do usuário seguindo o formato definido.

O resultado de um arquivo salvo durante a coleta no formato (*dat) é apresentado abaixo:

0 9985591 9988264 9943827 9991643 9942057 9973228 9870208 9979932 9868111 9909144 9962492 9991167 – primeira linha do arquivo
0,05 9985591 9988264 9943827 9991642 9942056 9973229 9870208 9979932 9868111 9909144 9962491 9991167 – segunda linha do arquivo
0,1 9985592 9988264 9943828 9991642 9942057 9973229 9870208 9979932 9868111 9909144 9962491 9991167 – terceira linha do arquivo
0,15 9985591 9988263 9943827 9991642 9942056 9973228 9870209 9979933 9868110 9909144 9962491 9991167 – quarta linha do arquivo
0,2 9985591 9988264 9943828 9991642 9942056 9973228 9870208 9979932 9868111 9909145 9962492 9991167 – quinta linha do arquivo

Lembrando que o primeiro número em cada linha é o tempo de coleta e que o resultado apresentado é apenas uma parte do arquivo.

A abertura deste arquivo em outros aplicativos ocorreu corretamente como se esperava.

O próximo resultado apresentado é da mesma coleta salva no formato (*mbq):

422 – número de linhas
0 – tempo de coleta
9985591
9988264
9943827

9991643
9942057
9973228
9870208
9979932
9868111
9909144
9962492
9991167
0,05
9985591
9988264
9943827
9991642
9942056
9973229
9870208
9979932
9868111
9909144
9962491
9991167
0,1
9985592
9988264
9943828
9991642
9942057
9973229
9870208
9979932
9868111
9909144
9962491
9991167
0,15
9985591
9988263
9943827
9991642
9942056
9973228
9870209
9979933
9868110
9909144
9962491
9991167
0,2
9985591
9988264

9943828
9991642
9942056
9973228
9870208
9979932
9868111
9909145
9962492
9991167

5.2 Utilizando o Sistema

O sistema pode ser operado em dois modos:

- O modo desconectado
- O modo conectado

A Figura 23 ilustra a interface gráfica do sistema.

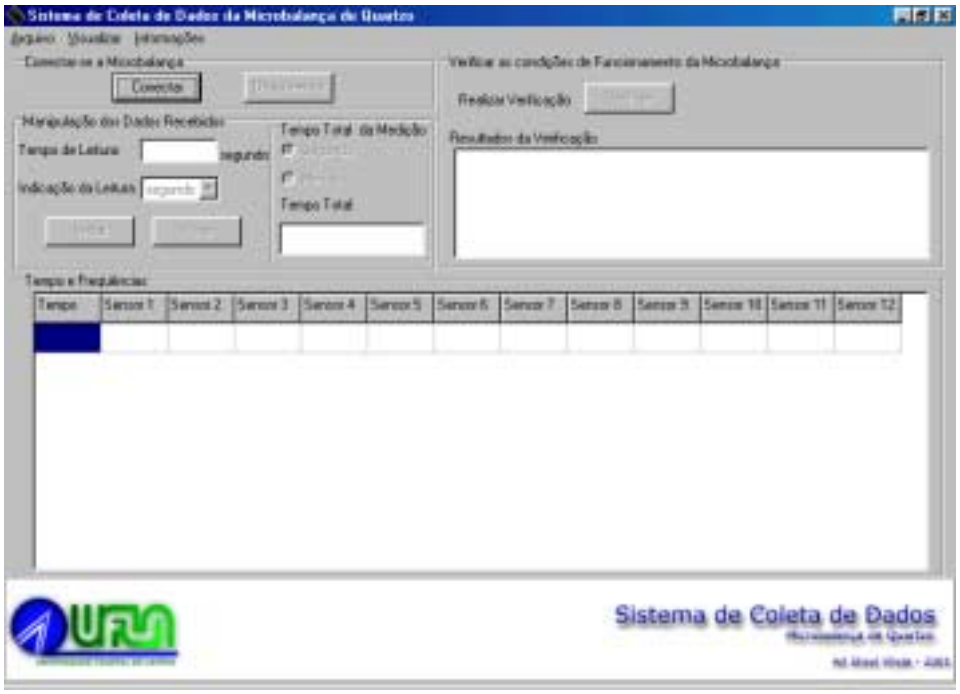

Figura 23: Interface gráfica do sistema.

5.2.1 Operando o Sistema no Modo Desconectado

O modo desconectado é operado para uma única finalidade; visualizar em gráfico os dados que foram salvos no formato (*mbq).Para realizar a visualização, o usuário deverá primeiro escolher o arquivo que deseja visualizar. Para isso ele deve ir ao menu Arquivo e escolher a opção Abrir, como ilustra a Figura 24.

Figura 24: Abrindo arquivo salvo.

Após clicar no item Abrir, o usuário poderá escolher o arquivo que deseja visualizar. Depois de escolher o arquivo, ele deverá clicar no botão Abrir, como ilustra a Figura 25.

O arquivo será aberto na própria interface do sistema, como ilustra a Figura 26.

Agora para que a visualização ocorra em modo gráfico, o usuário deve ir ao menu Visualizar e escolher a opção Gráfico off-line ou teclar a tecla “F6”, para que uma nova tela com o gráfico seja aberta, como ilustra as Figuras 27 e 28 respectivamente.

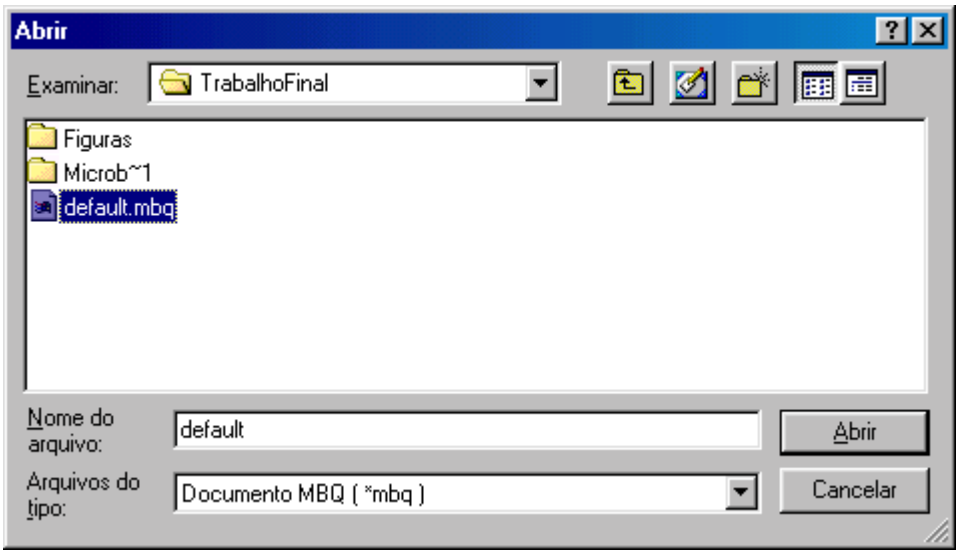

Figura 25: Escolhendo arquivo para ser aberto.

Figura 26: Arquivo aberto.

Figura 27: Visualizando o gráfico.

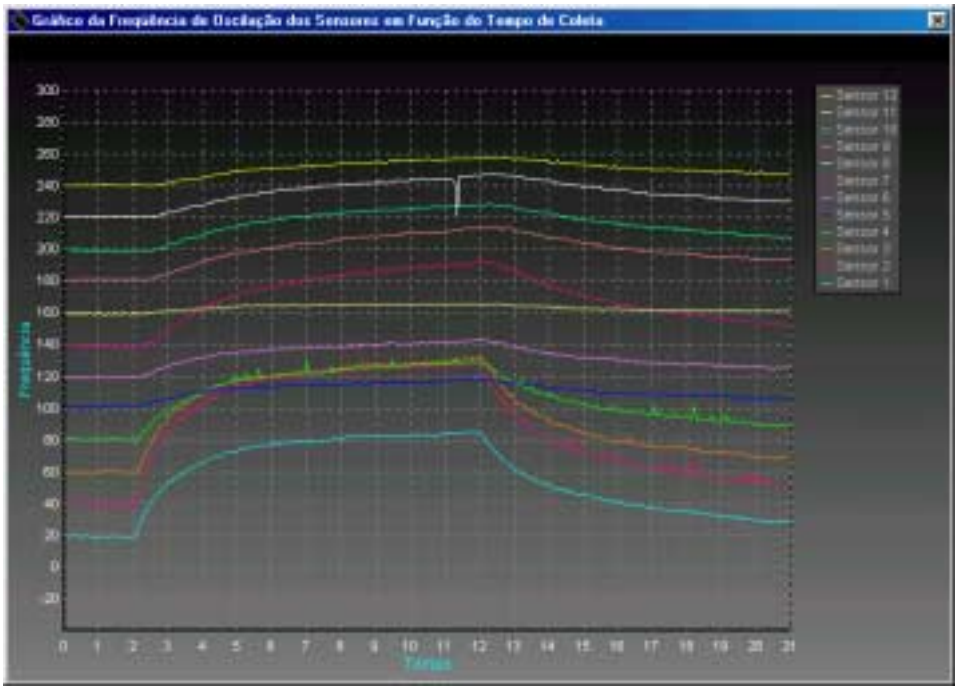

Figura 28: Interface de visualização do gráfico.

5.2.2 Operando no Modo Conectado

Operando no modo conectado o usuário poderá realizar:

- A verificação das condições de funcionamento da Microbalança;
- Realizar a aquisição dos dados.

Mas antes de tudo ele deve se conectar a Microbalança. Para isso basta clicar no botão conectar do groupbox Conectar-se a Microbalança, como ilustra a Figura 29.

Figura 29: Conectando a Microbalança.

Após clicar no botão Conectar, será aberta uma tela onde o usuário estabelece os parâmetros para a conexão e clica no botão OK, como ilustra a Figura 30.

É importante lembrar que os parâmetros para conexão já vêm estabelecidos, restando ao usuário definir somente qual a porta de comunicação a Microbalança esta conectada.

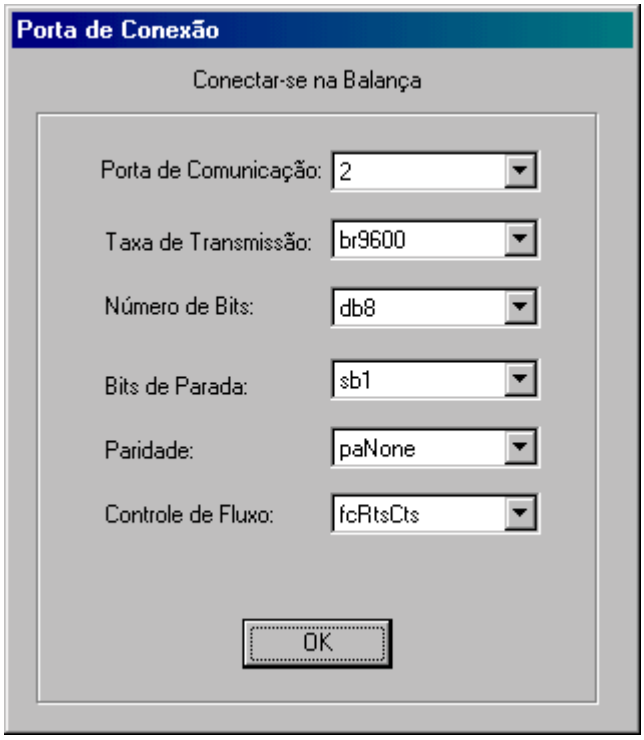

Figura 30: Parâmetros para estabelecer conexão.

5.2.2.1 Verificar Condições de Funcionamento da Microbalança

Estabelecida à conexão, o usuário já pode realizar a aquisição dados ou verificação da condições de funcionamento da Microbalança.

Para realizar as verificação das condições de funcionamento, o usuário deve ir ao groupbox Verificar Condições de Funcionamento da Microbalança e clicar no botão Verificar, como ilustra a Figura 31.

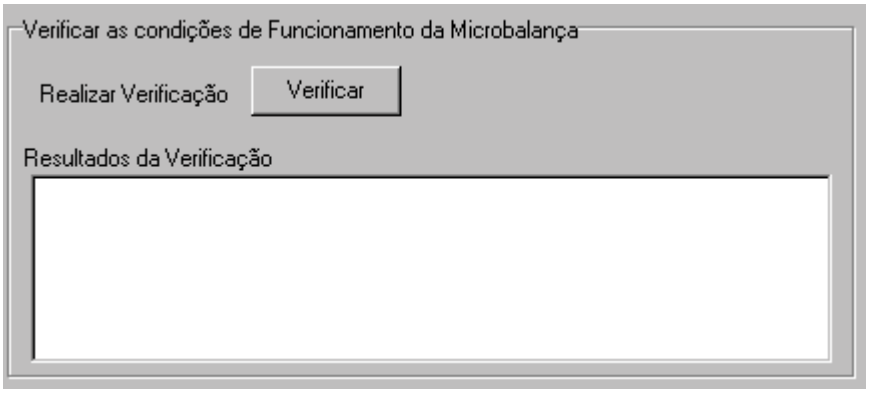

Figura 31: Verificando as condições de funcionamento da Microbalança.

Os resultados obtidos da verificação aparecerão na caixa de resposta Resultados da Verificação, como ilustra a Figura 31.

Lembrando que quando estiver tudo em ordem a resposta será Ok. Caso ocorra algum problema uma dessas resposta aparecerão:

- Oscilador 1 não propriamente rodando ou não conectado.
- Oscilador 2 não propriamente rodando ou não conectado.
- Oscilador 3 não propriamente rodando ou não conectado.
- Oscilador 4 não propriamente rodando ou não conectado.
- Oscilador 5 não propriamente rodando ou não conectado.
- Oscilador 6 não propriamente rodando ou não conectado.
- Falha na configuração do FPGA.
- Falha enquanto lê contadores.
- ROM mal, erro de checksum.
- RAM mal, erro de leitura/escrita.

5.2.2.2 Aquisição dos Dados

Um dos pré-requisitos para que a aquisição dos dados se inicie é a conexão do sistema com a Microbalança.

Para iniciar a aquisição, o usuário primeiro terá que criar o arquivo onde os dados serão salvos. Para isso, basta ir ao menu Arquivo e escolher a opção Novo, com ilustrado na Figura 32.

Figura 32: Criando o arquivo para salvar os dados.

Após clicar na opção Novo, uma nova tela irá aparecer para que o usuário escolha o local e nome onde deseja salvar os arquivos. Escolhido o local e o nome é só clicar no botão Salvar, como ilustra a Figura 33.

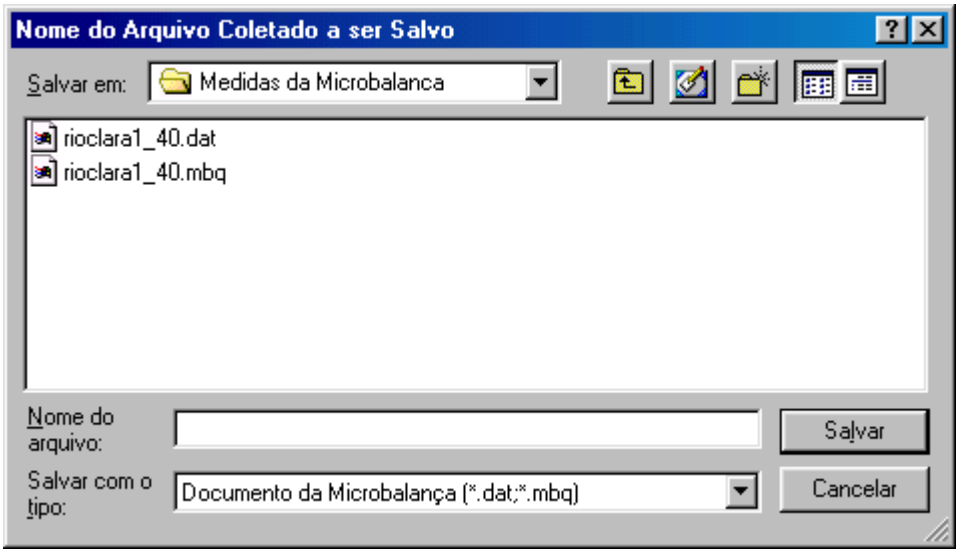

Figura 33: Nomeando arquivo de coleta que será salvo.

Somente depois de clicar no botão Salvar é que o botão Iniciar no groupbox Manipulação dos Dados Recebidos ficará disponível para uso.

Antes de iniciar a aquisição o usuário deve determinar o tempo de coleta (em segundos) e escolher no Indicação da Leitura a maneira como o tempo será exibido, como ilustra a Figura 34.

Após estabelecer o Tempo de Leitura e a Indicação da Leitura é só clicar no botão Iniciar. Neste momento o botão Parar ficará disponível para que o usuário possa encerrar a aquisição.

Figura 34: Inicio do processo de aquisição dos dados.

O outro groupbox da Figura 34 é utilizado para que o usuário possa ver o tempo total da coleta em minutos ou segundos. A escolha do tempo pode mudar na execução da aquisição.

Caso o usuário esqueça de passar o Tempo de Leitura, uma mensagem de alerta será exibida para ele, como ilustra a Figura 35.

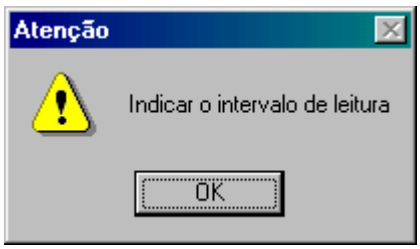

Figura 35: Mensagem de alerta.

A Figura 36 ilustra como fica o sistema no momento da aquisição dos dados.

Figura 36: Sistema realizando aquisição dos dados.

5.2.2.3 Visualizando os Dados em Modo Gráfico

Para que os dados coletados sejam analisados adequadamente, basta que o usuário visualize-os em modo gráfico.

Para visualizar os dados em modo gráfico, o usuário deverá ir ao menu Visualizar, na opção Gráfico on-line, ou teclar a tecla “F5”, como ilustra a Figura 37.

Figura 37: Visualização dos dados coletados em modo gráfico.

Uma nova interface será aberta para visualizar o gráfico, como o da Figura 28. É importante observar que o gráfico é traçado a cada aquisição de dados, portanto o usuário pode acompanhar a chegada dos dados através do gráfico.

5.2.2.4 Desconectando o Sistema da Microbalança

Para realizar a desconexão deve-se clicar no botão Desconectar no groupbox Conectar-se a Microbalança, como lustra a Figura 38.

A desconexão só pode ser concretizada com sucesso quando o usuário já tiver encerrado a aquisição dos dados.

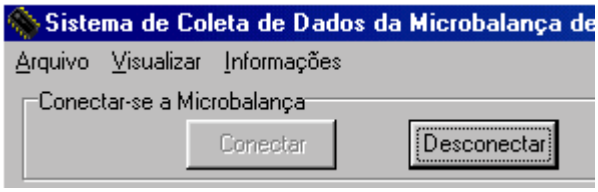

Figura 38: Desconectando o sistema da Microbalança.

6. CONCLUSÃO

O Sistema de Coleta de Dados de uma Microbalança de Quartzo, foi aceito com sucesso coletando, armazenando e visualizando os dados coletados em modo gráfico.

A utilização dos princípios de UML no desenvolvimento do sistema facilitou muito sua compreensão, e também auxiliou bastante para que sistema fosse implementado corretamente.

O componente de comunicação serial também alcançou todas as expectativas esperadas, auxiliando na coleta dos dados e no chaveamento do grupo dos sensores, sendo recomendado para outros desenvolvedores a sua utilização.

A utilização do arquivo (*dat) em outros aplicativos sucedeu-se com sucesso, cumprindo outra meta traçada para o desenvolvimento do sistema.

Por ser um sistema de código aberto, é possível que futuras atualizações ou melhorias possam acontecer beneficiando a Universidade.

O sistema hoje esta em uso no Departamento de Química da UFLA sob os cuidados do Prof. Mauro dos Santos Carvalho.

7. PROPOSTA DE CONTINUIDADE

Sugere-se para continuação do projeto, o desenvolvimento de um sistema de rede neural para analisar os dados coletados e identificar o tipo de aroma da amostra de café.

8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] ARC ELETRONICS. Padrão serial RS232. Disponível em <http://www.arcelect.com/rs232.htm>. Acesso em 20 nov. 2001.

[2] BIGNELL, J. W.; DONOVAN, R. L. **Eletrônica Digital: Lógica Combinacional**. São Paulo; SP: MAKRON Books do Brasil Editora Ltda, 1995. 432p.

[3] BIGNELL, J. W.; DONOVAN, R. L. **Eletrônica Digital: Lógica Seqüencial**. São Paulo; SP: MAKRON Books do Brasil Editora Ltda, 1995. 383p.

[4] BORLAND. Página com informações da ferramenta de desenvolvimento Delphi 5. Disponível em: <http://www.borland.com/delphi/>. Acesso em 28 ago. 2001.

[5] CANTÙ, M. **Dominando o Delphi 5: A Bíblia**. São Paulo, SP. MAKRON Books do Brasil Editora Ltda, 1999. 860p.

[6] COMPUTER TIPS. Interface serial RS232. Disponível em: <http://www.ctips.com/rs232.html>. Acesso em 21 nov. 2001.

[7] FOWLER, M.; SCOTT, K. **UML Essencial: Um breve guia para a linguagem padrão de modelagem de objetos.** 2. ed. São Paulo, SP. Bookman Companhia Editora, 2000. 169p.

[8] FURLAN, J. D.; **Modelagem de Objetos através da UML.** São Paulo, SP. MAKRON Books do Brasil Editora Ltda, 1998. 329p.

[9] LARMAN, G. **Utilizando UML e Padrões.** Porto Alegre, RS. Bookman Companhia Editora, 2000. 492p

[10] MALVINO, A. P.; LEACH, D. P. **Eletrônica Digital Princípios e Aplicações: Lógica Combinacional**. São Paulo, SP: Editora McGraw-Hill Ltda, 1987. 355p.

[11] MARIC, S. **Informações sobre detalhes de comunicação entre o PC e a Microbalança de Quartzo** [mensagem pessoal]. Mensagem recebida por <s.maric@ndh.net> em 04 set. 2001.

[12] SCHALLER, E.; O BOSSET, J.; ESCHER, F. **Electronic Noses and their Applications to Food**. Lebensm-Wisse. U.-Technol., 31, 1998, 305 - 316p.

[13] STEPHAM, A.; BUCKING, M.; STEINHART, H. **Novel analytical tools for flavours**. Food Research International, 33, 2000, 199 - 209p.

[14] ULMER, H.; MITROVICS, J.; NOETZEL, G.; WEIMAR, U.; GOPEL, W. **Odours and Flavours Identified with Hybrid Modular Sensor Systems**. Sensors and Actuators B, 43, 1997, 24 - 33p.

[15] VARIAN-SOFTWARE. Página com componentes de comunicação serial do Delphi. Disponível em: <http://www.varian-software.com>. Acesso em 13 set. 2001.

[16] ZELENOVSKY, R.; MENDONÇA, A. **PC: Um Guia Prático de Hardware e Interfaceamento**. 2. ed. rev. e atual. Rio de Janeiro, RJ: MZ Editora Ltda, 1999. 762p.

[17] ZUFFO, J. A. **Microprocessadores: Dutos de sistema, técnicas de interface e sistema de comunicação de dados**. São Paulo, SP: Editora Edgar Blüncher Ltda, 1981. 535p.

ANEXO A

Tabela ASCII

Tabela ASCII					
Decimal	Hexadecimal	**Caracter**	Decimal	Hexadecimal	**Caracter**
0	0	**NUL**	64	40	**@**
1	1	**SOH**	65	41	**A**
2	2	**STX**	66	42	**B**
3	3	**ETX**	67	43	**C**
4	4	**EOT**	68	44	**D**
5	5	**ENQ**	69	45	**E**
6	6	**ACK**	70	46	**F**
7	7	**BEL**	71	47	**G**
8	8	**BS**	72	48	**H**
9	9	**HT**	73	49	**I**
10	A	**LF**	74	4A	**J**
11	B	**VT**	75	4B	**K**
12	C	**FF**	76	4C	**L**
13	D	**CR**	77	4D	**M**
14	E	**SO**	78	4E	**N**
15	F	**SI**	79	4F	**O**
16	10	**DLE**	80	50	**P**
17	11	**DC1**	81	51	**Q**
18	12	**DC2**	82	52	**R**
19	13	**DC3**	83	53	**S**
20	14	**DC4**	84	54	**T**
21	15	**NAK**	85	55	**U**
22	16	**SYN**	86	56	**V**
23	17	**ETB**	87	57	**W**
24	18	**CAN**	88	58	**X**
25	19	**EM**	89	59	**Y**
26	1A	**SUB**	90	5A	**Z**
27	1B	**ESC**	91	5B	**[**
28	1C	**FS**	92	5C	\
29	1D	**GS**	93	5D	**]**
30	1E	**RS**	94	5E	^
31	1F	**US**	95	5F	**_**
32	20	**SPC**	96	60	**`**
33	21	**!**	97	61	**a**
34	22	**“**	98	62	**b**
35	23	**#**	99	63	**c**
36	24	**$**	100	64	**d**
37	25	**%**	101	65	**e**

38	26	**&**	102	66	**f**
39	27	**‘**	103	67	**g**
40	28	**(**	104	68	**h**
41	29	**)**	105	69	**i**
42	2A	*****	106	6A	**j**
43	2B	+	107	6B	**k**
44	2C	**‘**	108	6C	**l**
45	2D	-	109	6D	**m**
46	2E	**.**	110	6E	**n**
47	2F	/	111	6F	**o**
48	30	**0**	112	70	**p**
49	31	**1**	113	71	**q**
50	32	**2**	114	72	**r**
51	33	**3**	115	73	**s**
52	34	**4**	116	74	**t**
53	35	**5**	117	75	**u**
54	36	**6**	118	76	**v**
55	37	**7**	119	77	**w**
56	38	**8**	120	78	**x**
57	39	**9**	121	79	**y**
58	3A	**:**	122	7A	**z**
59	3B	**;**	123	7B	**(**
60	3C	<	124	7C	**|**
61	3D	=	125	7D	**)**
62	3E	>	126	7E	**~**
63	3F	**?**	127	7F	**DEL**

ANEXO B

Símbolos de Visibilidade

Um atributo ou uma operação em um diagrama de classe pode ser declarado como público, protegido ou privado.

A Tabela abaixo ilustra a simbologia utilizada nos diagramas de classe para declarar um atributo ou uma operação.

Símbolo	Nome
+	Pública
#	Protegida
-	Privada

ANEXO C

Código Fonte do Programa

```
{  -- SISTEMA DE COLETA DE DADOS DE UMA MICROBALANCA DE QUARTZO
   -- Programa desenvolvido para a disciplina de Projeto Orientado
   -- Ari Elisei Vilela
   -- Orientador: Wilian Soares Lacerda
   -- Co-orientador: Mauro dos Santos Carvalho
   -- Co-orientador: Antonio Maria Pereira de Resende
   -- Outubro de 2001
   -- Universidade Federal de Lavras - Minas Gerais }

unit UnitPrincipal;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ImgList, Menus, StdCtrls, Grids, ExtCtrls, VaClasses, VaComm, jpeg;

type
  TFormPrincipal = class(TForm)
    Menu: TMainMenu;
    ItemArquivo: TMenuItem;
    ItemNovo: TMenuItem;
    ItemAbrir: TMenuItem;
    N1: TMenuItem;
    ItemSair: TMenuItem;
    ItemVisualizar: TMenuItem;
    ItemGrafico: TMenuItem;
    ItemGraficoOffLine: TMenuItem;
    ItemInformacoes: TMenuItem;
    ItemSobre: TMenuItem;
    N2: TMenuItem;
    ItemBalanca: TMenuItem;
    ImageListMenu: TImageList;
    OpenDialog1: TOpenDialog;
    SaveDialogColeta: TSaveDialog;
    GroupConectar: TGroupBox;
    GroupVerificar: TGroupBox;
    GroupBoxTempoSensores: TGroupBox;
    GroupTempoTotal: TGroupBox;
    GroupAquisicao: TGroupBox;
    BotaoConectar: TButton;
    BotaoDisconectar: TButton;
    BotaoVerificar: TButton;
    BotaoIniciar: TButton;
    BotaoParar: TButton;
    StringGrid1: TStringGrid;
    Panel1: TPanel;
```

```
    Label1: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    Label3: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    EditTempo: TEdit;
    EditTempoTotal: TEdit;
    ListBoxVerificador: TListBox;
    ComboTempo: TComboBox;
    TimerMedidas: TTimer;
    RadioSegundo: TRadioButton;
    RadioMinuto: TRadioButton;
    ImageLogo: TImage;
    Image1: TImage;

    procedure FormCreate(Sender: TObject);
    procedure ItemNovoClick(Sender: TObject);
    procedure ItemGraficoOffLineClick(Sender: TObject);
    procedure ItemAbrirClick(Sender: TObject);
    procedure ItemSairClick(Sender: TObject);
    procedure ItemGraficoClick(Sender: TObject);
    procedure ItemSobreClick(Sender: TObject);
    procedure ItemBalancaClick(Sender: TObject);
    procedure BotaoConectarClick(Sender: TObject);
    procedure BotaoDisconectarClick(Sender: TObject);
    procedure BotaoVerificarClick(Sender: TObject);
    procedure BotaoIniciarClick(Sender: TObject);
    procedure BotaoPararClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure TimerMedidasTimer(Sender: TObject);

  private
// procedimentos
    procedure MontaGrafico(Tempo: Real; Linha: Integer);
    procedure MontaGraficoOffLine(linhas: Integer);
    procedure EnableMedidas(dado: String);
    procedure DisableMedidas(dado: String);
    procedure SalvarDat(caminho: String);
    procedure SalvarMbq(caminho: String);
    procedure StatusBalanca(dado_recebido: String);
    procedure TempoTotal(tempo: String);

// funcoes
    function Somador(palavra: String): Integer;
    function SomaTempo(tempo: Real): Real;
    function Conversao(palavra: String): String;

  public
  end;
var
```

```
  FormPrincipal: TFormPrincipal;

implementation

uses UnitGrafico, UnitSobre, UnitConfiguracao, UnitGraficoOffLine,
  UnitApres;

{$R *.DFM}

//---------------------------------------------------------------------------
procedure TFormPrincipal.FormCreate(Sender: TObject);
begin
   BotaoDisconectar.Enabled:= false;
   BotaoVerificar.Enabled:= false;
   BotaoIniciar.Enabled:= false;
   BotaoParar.Enabled:= false;

   ItemNovo.Enabled:= false;
   ItemGrafico.Enabled:= false;
   ItemGraficoOffLine.Enabled:= false;
   ItemBalanca.Enabled:= false;

   ComboTempo.Enabled:= false;
   with ComboTempo do
     ItemIndex:= Items.IndexOF('segundo');
   EditTempo.Enabled:= false;
   TimerMedidas.Enabled:= false;

   RadioSegundo.Checked:= true;
   RadioSegundo.Enabled:= false;
   RadioMinuto.Enabled:= false;

// Montando a StringGrid com seus nomes
   StringGrid1.Cells[0,0]:= 'Tempo';
   StringGrid1.Cells[1,0]:= 'Sensor 1';
   StringGrid1.Cells[2,0]:= 'Sensor 2';
   StringGrid1.Cells[3,0]:= 'Sensor 3';
   StringGrid1.Cells[4,0]:= 'Sensor 4';
   StringGrid1.Cells[5,0]:= 'Sensor 5';
   StringGrid1.Cells[6,0]:= 'Sensor 6';
   StringGrid1.Cells[7,0]:= 'Sensor 7';
   StringGrid1.Cells[8,0]:= 'Sensor 8';
   StringGrid1.Cells[9,0]:= 'Sensor 9';
   StringGrid1.Cells[10,0]:= 'Sensor 10';
   StringGrid1.Cells[11,0]:= 'Sensor 11';
   StringGrid1.Cells[12,0]:= 'Sensor 12';

end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
```

```
// Iniciar todos os procedimentos para novas coletas
procedure TFormPrincipal.ItemNovoClick(Sender: TObject);
var
  x, y: integer;
  aux_dat, aux_mbq: string;
begin

  SaveDialogColeta.FileName:= '';
// Limpa a StringGrid
  for x:= 0 to StringGrid1.ColCount do
    for y:=1 to StringGrid1.RowCount - 1 do
      StringGrid1.Cells[x, y]:= '';

// Restaura a StringGrid
  StringGrid1.RowCount:= 2;
  StringGrid1.ColCount:= 13;

// Limpa o grafico
  FormGrafico.Series1.Clear;
  FormGrafico.Series2.Clear;
  FormGrafico.Series3.Clear;
  FormGrafico.Series4.Clear;
  FormGrafico.Series5.Clear;
  FormGrafico.Series6.Clear;
  FormGrafico.Series7.Clear;
  FormGrafico.Series8.Clear;
  FormGrafico.Series9.Clear;
  FormGrafico.Series10.Clear;
  FormGrafico.Series11.Clear;
  FormGrafico.Series12.Clear;

  EditTempoTotal.Text:= '';
  EditTempo.Clear;
  EditTempo.Enabled:= true;
  ComboTempo.Enabled:= true;
  RadioSegundo.Enabled:= true;
  RadioMinuto.Enabled:= true;
  ListBoxVerificador.Clear;

// Inicia o preocesso de salvamento dos dados
  if SaveDialogColeta.Execute then
  begin
    aux_dat:= SaveDialogColeta.FileName + '.dat';
    aux_mbq:= SaveDialogColeta.FileName + '.mbq';

    // Verificar arquivo existente no formato *.dat ou mbq
    if (FileExists(aux_dat)) or (FileExists(aux_mbq)) then
    begin
      if MessageDlg('Deseja substituir o arquivo já existente?', mtInformation, [mbYes, mbNo], 0)
= mrYes then
        BotaoIniciar.Enabled:= true
```

```
    end;
    BotaoIniciar.Enabled:= true;
  end // if
  else
    BotaoIniciar.Enabled:= false
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Utilizado para abrir uma medida salva, operando desconectado do equipamento
procedure TFormPrincipal.ItemAbrirClick(Sender: TObject);
var
  arquivo: TextFile;
  numero_linhas, x, y, i, j: integer;
  linha: string;
begin
// Abrindo o arquivo
  if OpenDialog1.Execute then
  begin
    // Limpa a StringGrid
    for i:= 0 to StringGrid1.ColCount do
    for j:=1 to StringGrid1.RowCount - 1 do
      StringGrid1.Cells[i, j]:= '';

    // Restaura a StringGrid
    StringGrid1.RowCount:= 2;
    StringGrid1.ColCount:= 13;

    Assignfile (arquivo,OpenDialog1.FileName);
    Reset(arquivo);
    Readln(arquivo,numero_linhas);
    StringGrid1.RowCount:= numero_linhas;

    // Passa o arquivo lido para a StringGrid
    for y:=1 to StringGrid1.RowCount do
      for x:=0 to StringGrid1.ColCount-1 do
      begin
        Readln(arquivo, linha);
        StringGrid1.Cells[x,y]:=linha;
      end;// for
    Closefile(arquivo);
    ItemGraficoOffLine.Enabled:= true;
  end;// if

  // Limpa o grafico caso ja tenha aberto algum arquivo antes
  FormGraficoOffLine.Series1.Clear;
  FormGraficoOffLine.Series2.Clear;
  FormGraficoOffLine.Series3.Clear;
  FormGraficoOffLine.Series4.Clear;
  FormGraficoOffLine.Series5.Clear;
  FormGraficoOffLine.Series6.Clear;
```

```
  FormGraficoOffLine.Series7.Clear;
  FormGraficoOffLine.Series8.Clear;
  FormGraficoOffLine.Series9.Clear;
  FormGraficoOffLine.Series10.Clear;
  FormGraficoOffLine.Series11.Clear;
  FormGraficoOffLine.Series12.Clear;
end;
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
// Sair do programa
procedure TFormPrincipal.ItemSairClick(Sender: TObject);
begin
// Verificar se o usuario esta saindo do programa antes de encerrar a coleta
// dos dados
  if BotaoParar.Enabled = true then
     Application.MessageBox('Não é possível encerrar o programa enquanto o mesmo estiver
coletando os dados', 'Atenção', 48)
  else begin
  FormConfiguracao.VaComm1.Close;// Realizando a Desconexao
  Application.Terminate;// Finalizando o Programa
  end;// else
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Visualizar o grafico dos dados coletados em tempo de execucao
procedure TFormPrincipal.ItemGraficoClick(Sender: TObject);
begin
  FormGrafico.showmodal;
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Visualizacao dos dados salvos em grafico(Utilizado somente quando o
// programa nao estiver conectado à Microbalanca)
procedure TFormPrincipal.ItemGraficoOffLineClick(Sender: TObject);
var
  numero_linhas: integer;
begin
  numero_linhas:= StringGrid1.RowCount - 1;
// Procedimento utilizado para tracar o grafico
  MontaGraficoOffLine(numero_linhas);
  FormGraficoOffLine.ShowModal;
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Visualizar as informacoes sobre o programa
procedure TFormPrincipal.ItemSobreClick(Sender: TObject);
begin
  FormSobre.showmodal;
```

```
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Para receber as inforamcoes da balanca
procedure TFormPrincipal.ItemBalancaClick(Sender: TObject);
var
  receber: string;
begin
  FormConfiguracao.VaComm1.WriteText('T0' + #13#10);
  sleep(200);
  receber:= FormConfiguracao.VaComm1.ReadText;
  showmessage(receber);
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Finalizar o programa
procedure TFormPrincipal.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  FormConfiguracao.VaComm1.Close;// Realiza a Desconexao
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Conectar-se a balanca - chamara uma tela de configuracao para conectar-se
procedure TFormPrincipal.BotaoConectarClick(Sender: TObject);
var x,y: integer;
begin
// Limpa a StringGrid, caso o usuario tenha aberto algum arquivo antes
  for x:= 0 to StringGrid1.ColCount do
    for y:=1 to StringGrid1.RowCount - 1 do
      StringGrid1.Cells[x, y]:= '';

// Restaura a StringGrid
  StringGrid1.RowCount:= 2;
  StringGrid1.ColCount:= 13;
  ListBoxVerificador.Clear;

  FormConfiguracao.ShowModal;

  BotaoConectar.Enabled:= false;
  BotaoDisconectar.Enabled:= true;
  BotaoVerificar.Enabled:= true;
  BotaoIniciar.Enabled:= false;
  BotaoParar.Enabled:= false;

  ItemNovo.Enabled:= true;
  ItemAbrir.Enabled:= false;
  ItemGrafico.Enabled:= true;
  ItemGraficoOffLine.Enabled:= false;
```

```
  ItemBalanca.Enabled:= true;
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Botao para desconectar-se da balanca
procedure TFormPrincipal.BotaoDisconectarClick(Sender: TObject);
begin
// Verificar se o usuario esta saindo do programa antes de encerrar a coleta
// dos dados
  if BotaoParar.Enabled = true then
     Application.MessageBox('Não é possível desconectar com a coleta de dados em andamento',
'Atenção',48)
  else begin
     FormConfiguracao.VaComm1.Close;
     BotaoDisconectar.Enabled:= false;
     BotaoConectar.Enabled:= true;
     BotaoVerificar.Enabled:= false;
     BotaoIniciar.Enabled:= false;
     BotaoParar.Enabled:= false;
     ItemNovo.Enabled:= false;
     ItemAbrir.Enabled:= true;
     ItemBalanca.Enabled:= false;
     ItemGrafico.Enabled:= false;
     ItemGraficoOffLine.Enabled:= true;
     ComboTempo.Enabled:= false;
     EditTempo.Enabled:= false;
     RadioSegundo.Enabled:= false;
     RadioMinuto.Enabled:= false;
  end;// else
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Butao utilizado para realizar a verificacao das condicoes da Microbalanca
procedure TFormPrincipal.BotaoVerificarClick(Sender: TObject);
var
  receber: string;
begin
  FormConfiguracao.VaComm1.WriteText('? GST' + #13#10);
  sleep(150);
  receber:= FormConfiguracao.VaComm1.ReadText;
// Procedimento para analisar o dado recebido da Microbalanca
  StatusBalanca(receber);
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Iniciar o processo de recebimento dos dados dos sensores da balanca
procedure TFormPrincipal.BotaoIniciarClick(Sender: TObject);
var
```

```
  tempo, dado_coletado: string;
  intervalo: integer;
begin
  if EditTempo.Text = '' then
  begin
    beep;
    Application.MessageBox('Indicar o intervalo de leitura', 'Atenção', 48);
  end// if
  else begin
    StringGrid1.Cells[0,StringGrid1.RowCount - 1]:= '0';
// muda o chaveamento para os seis primeiros sensores
    dado_coletado:= FormConfiguracao.AquisicaoDados('? GDA' +#13#10);
// Procedimento para receber os seis primeiros dados da Microbalanca
    EnableMedidas(dado_coletado);
// muda o chaveamento para os seis ultimos sensores
    FormConfiguracao.FluxoControle(false);
    dado_coletado:= FormConfiguracao.AquisicaoDados('? GDA' +#13#10);
// Procedimento para receber os seis ultimos dados da Microbalanca
    DisableMedidas(dado_coletado);
    StringGrid1.RowCount:= StringGrid1.RowCount + 1;

    tempo:=EditTempo.Text;
    intervalo:= StrToInt(tempo) * 1000;
    TimerMedidas.Interval:= intervalo;
    TimerMedidas.Enabled:= true;
    BotaoIniciar.Enabled:= false;
    BotaoParar.Enabled:= true;
    ItemNovo.Enabled:= false;
    EditTempo.Enabled:= false;
    ComboTempo.Enabled:= false;
  end;
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Butao utilizado para encerrar a coleta dos dados
procedure TFormPrincipal.BotaoPararClick(Sender: TObject);
begin
  BotaoParar.Enabled:= false;
  TimerMedidas.Enabled:= false;

  ItemNovo.Enabled:= true;
  EditTempo.Enabled:= false;
  ComboTempo.Enabled:= false;
  RadioSegundo.Enabled:= false;
  RadioMinuto.Enabled:= false;

// Utilizado para retornar o chaveamento para os seis primeiros sensores,
// devido ao fato que ao encerrar a coleta o chaveamento terminar nos seis
// ultimos sensores
  FormConfiguracao.FluxoControle(true);
```

```
// Limpar o buffer
  FormConfiguracao.VaComm1.PurgeReadWrite;
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Contador de tempo utilizado para a coleta dos dados
procedure TFormPrincipal.TimerMedidasTimer(Sender: TObject);
var
  valor_tempo, dado_coletado: string;
  linha_grafico: integer;
  tempo_grafico, tempo: real;
begin
// muda o chaveamento para os seis primeiros sensores
  FormConfiguracao.FluxoControle(true);

  if ComboTempo.Items[ComboTempo.ItemIndex] = 'segundo' then begin
    tempo:= TimerMedidas.Interval / 1000;
    tempo:= SomaTempo(tempo);
    valor_tempo:= FloatToStr(tempo);
  end
  else begin
    tempo:= TimerMedidas.Interval / 60000;
    tempo:= SomaTempo(tempo);
    valor_tempo:= FloatToStr(tempo);
  end;

  StringGrid1.Cells[0,StringGrid1.RowCount - 1]:= valor_tempo;

// Receber do sensor 1 ao 6
  dado_coletado:= FormConfiguracao.AquisicaoDados('? GDA' + #13#10);
// Procedimento para receber os seis primeiros dados da Microbalanca
  EnableMedidas(dado_coletado);

// Mudar o chaveamento
  FormConfiguracao.FluxoControle(false);

  // Receber do sensor 7 ao 12
  dado_coletado:= FormConfiguracao.AquisicaoDados('? GDA' + #13#10);
// Procedimento para receber os seis ultimos dados da Microbalanca
  DisableMedidas(dado_coletado);

// Limpar o buffer
  FormConfiguracao.VaComm1.PurgeReadWrite;

// Utilizado para salvar os dados coletados em seus formatos
  SalvarDat(SaveDialogColeta.FileName + '.dat');
  SalvarMbq(SaveDialogColeta.FileName + '.mbq');

// Sera utilizado para tracar o grafico em tempo de execucao
```

```
  tempo_grafico:= StrToFloat(StringGrid1.Cells[0,StringGrid1.RowCount - 1]);
  linha_grafico:= StringGrid1.RowCount - 1;

// Procedimento para tracar o grafico
  MontaGrafico(tempo_grafico, linha_grafico);

// Procedimento para calcular o tempo total
  TempoTotal(StringGrid1.Cells[0,StringGrid1.RowCount - 1]);

  StringGrid1.RowCount:= StringGrid1.RowCount + 1;
end;
//---------------------------------------------------------------------------

//              Procedimento utilizados pelo programa              //

//---------------------------------------------------------------------------
// Procedimento utilizado para retornar as condicoes de operacao da Microbalanca
procedure TFormPrincipal.StatusBalanca(dado_recebido: String);
var
  palavra_convertida, verificador: string;
  i: integer;
begin
// Realizar a conversao de hexadecimal para binario
  palavra_convertida:= Conversao(dado_recebido);
// Realiza o somatorio da palavra convertida para binario

  verificador:= IntToStr(Somador(palavra_convertida));

// inicio da verificacao do status mostrando os valores ansalisados
  if verificador = '0' then
    ListBoxVerificador.Items.Add('Ok')
  else begin
    for i:=16 downto 1 do
    begin
      if (palavra_convertida[i] = '1') and (i = 16)  then
        ListBoxVerificador.Items.Add('Oscilador 1 não propriamente rodando ou não conectado')
      else if (palavra_convertida[i] = '1') and (i = 15)  then
        ListBoxVerificador.Items.Add('Oscilador 2 não propriamente rodando ou não conectado')
      else if (palavra_convertida[i] = '1') and (i = 14) then
        ListBoxVerificador.Items.Add('Oscilador 3 não propriamente rodando ou não conectado')
      else if (palavra_convertida[i] = '1') and (i = 13) then
        ListBoxVerificador.Items.Add('Oscilador 4 não propriamente rodando ou não conectado')
      else if (palavra_convertida[i] = '1') and (i = 12) then
        ListBoxVerificador.Items.Add('Oscilador 5 não propriamente rodando ou não conectado')
      else if (palavra_convertida[i] = '1') and (i = 11) then
        ListBoxVerificador.Items.Add('Oscilador 6 não propriamente rodando ou não conectado')
      else if (palavra_convertida[i] = '1') and (i = 8) then
        ListBoxVerificador.Items.Add('Falha na configuração do FPGA')
      else if (palavra_convertida[i] = '1') and (i = 7) then
        ListBoxVerificador.Items.Add('Falha lendo contadores')
      else if (palavra_convertida[i] = '1') and (i = 6) then
```

```
          ListBoxVerificador.Items.Add('ROM mal, erro de somador')
        else if (palavra_convertida[i] = '1') and (i = 5) then
          ListBoxVerificador.Items.Add('RAM mal, erro de leitura/escrita')
      end; // fim do for

    end; // fim do if
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Procedimento para exibir os seis primeiros dados dos sensores
procedure TFormPrincipal.EnableMedidas(dado: String);
begin
  if (dado[1] = '!') and (dado[4] = 'D') then begin
    if dado[7] <> '' then
      StringGrid1.Cells[1,StringGrid1.RowCount - 1]:= Copy(dado,7,8)
    else
      StringGrid1.Cells[1,StringGrid1.RowCount - 1]:= Copy(dado,8,7);
    if dado[16] <> '' then
      StringGrid1.Cells[2,StringGrid1.RowCount - 1]:= Copy(dado,16,8)
    else
      StringGrid1.Cells[2,StringGrid1.RowCount - 1]:= Copy(dado,17,7);
    if dado[25] <> '' then
      StringGrid1.Cells[3,StringGrid1.RowCount - 1]:= Copy(dado,25,8)
    else
      StringGrid1.Cells[3,StringGrid1.RowCount - 1]:= Copy(dado,26,7);
    if dado[34] <> '' then
      StringGrid1.Cells[4,StringGrid1.RowCount - 1]:= Copy(dado,34,8)
    else
      StringGrid1.Cells[4,StringGrid1.RowCount - 1]:= Copy(dado,35,7);
    if dado[43] <> '' then
      StringGrid1.Cells[5,StringGrid1.RowCount - 1]:= Copy(dado,43,8)
    else
      StringGrid1.Cells[5,StringGrid1.RowCount - 1]:= Copy(dado,44,7);
    if dado[52] <> '' then
      StringGrid1.Cells[6,StringGrid1.RowCount - 1]:= Copy(dado,52,8)
    else
      StringGrid1.Cells[6,StringGrid1.RowCount - 1]:= Copy(dado,53,7);
  end;// fim do if dos seis primeiros sensores}
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Procedimento para exibir os seis ultimos dados dos sensores
procedure TFormPrincipal.DisableMedidas(dado: String);
begin
  if (dado[1] = '!') and (dado[4] = 'D') then begin
    if dado[7] <> '' then
      StringGrid1.Cells[7,StringGrid1.RowCount - 1]:= Copy(dado,7,8)
    else
      StringGrid1.Cells[7,StringGrid1.RowCount - 1]:= Copy(dado,8,7);
```

```
    if dado[16] <> '' then
      StringGrid1.Cells[8,StringGrid1.RowCount - 1]:= Copy(dado,16,8)
    else
      StringGrid1.Cells[8,StringGrid1.RowCount - 1]:= Copy(dado,17,7);
    if dado[25] <> '' then
      StringGrid1.Cells[9,StringGrid1.RowCount - 1]:= Copy(dado,25,8)
    else
      StringGrid1.Cells[9,StringGrid1.RowCount - 1]:= Copy(dado,26,7);
    if dado[34] <> '' then
      StringGrid1.Cells[10,StringGrid1.RowCount - 1]:= Copy(dado,34,8)
    else
      StringGrid1.Cells[10,StringGrid1.RowCount - 1]:= Copy(dado,35,7);
    if dado[43] <> '' then
      StringGrid1.Cells[11,StringGrid1.RowCount - 1]:= Copy(dado,43,8)
    else
      StringGrid1.Cells[11,StringGrid1.RowCount - 1]:= Copy(dado,44,7);
    if dado[52] <> '' then
      StringGrid1.Cells[12,StringGrid1.RowCount - 1]:= Copy(dado,52,8)
    else
      StringGrid1.Cells[12,StringGrid1.RowCount - 1]:= Copy(dado,53,7);
  end;// fim fo if dos seis ultimos sensores
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Procedimento utilizada para desevolver o tempo total das medidas
procedure TFormPrincipal.TempoTotal(tempo: String);
var
  aux: real;
begin
  if (ComboTempo.Items[ComboTempo.ItemIndex] = 'segundo') and (RadioSegundo.Checked =
true) then
    EditTempoTotal.Text:= tempo
  else if (ComboTempo.Items[ComboTempo.ItemIndex] = 'minuto') and (RadioMinuto.Checked =
true) then
    EditTempoTotal.Text:= tempo
  else if (ComboTempo.Items[ComboTempo.ItemIndex] = 'segundo') and (RadioMinuto.Checked
= true) then
  begin
    aux:= StrToFloat(tempo) / 60;
    EditTempoTotal.Text:= FloatToStr(aux);
  end// else if
  else begin
    aux:= StrToFloat(tempo) * 60;
    EditTempoTotal.Text:= FloatToStr(aux);
  end;// else

end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
```

```
// Utilizado para salvar os dados recebidos no formato dat
procedure TFormPrincipal.SalvarDat(caminho: String);
var
  arquivo: TextFile;
  i,j: integer;
begin
  AssignFile(arquivo,caminho);
  Rewrite(arquivo);
    for j:=1 to StringGrid1.RowCount - 2 do
    begin
      for i:=0 to StringGrid1.ColCount - 1 do
      begin
        Write(arquivo,StringGrid1.Cells[i,j]);
      end;// for
      Writeln(arquivo);
    end;// for
  CloseFile(arquivo);
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Utilizado para salvar os dados recebidos no formato mbq
procedure TFormPrincipal.SalvarMbq(caminho: String);
var
  arquivo: TextFile;
  i,j: integer;
begin
  AssignFile(arquivo,caminho);
  Rewrite(arquivo);
  Writeln(arquivo,StringGrid1.RowCount - 1);
  for j:=1 to StringGrid1.RowCount - 2 do
  begin
    for i:=0 to StringGrid1.ColCount - 1 do
      Writeln(arquivo,StringGrid1.Cells[i,j]);
  end;// for
  CloseFile(arquivo);
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Procedimento utilizado para tracar grafico em tempo real
procedure TFormPrincipal.MontaGrafico(Tempo: Real; Linha: Integer);
var
  FA, FP, Resultado: Real;
begin
    FA:= StrToFloat(StringGrid1.Cells[1,1]);
    FP:= StrToFloat(StringGrid1.Cells[1,Linha]);
    Resultado:= FA - FP + 20;// sensor 1
    FormGrafico.Series1.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[2,1]);
    FP:= StrToFloat(StringGrid1.Cells[2,Linha]);
```

```
    Resultado:= FA - FP + 40;// sensor 2
    FormGrafico.Series2.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[3,1]);
    FP:= StrToFloat(StringGrid1.Cells[3,Linha]);
    Resultado:= FA - FP + 60;// sensor 3
    FormGrafico.Series3.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[4,1]);
    FP:= StrToFloat(StringGrid1.Cells[4,Linha]);
    Resultado:= FA - FP + 80;// sensor 4
    FormGrafico.Series4.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[5,1]);
    FP:= StrToFloat(StringGrid1.Cells[5,Linha]);
    Resultado:= FA - FP + 100;// sensor 5
    FormGrafico.Series5.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[6,1]);
    FP:= StrToFloat(StringGrid1.Cells[6,Linha]);
    Resultado:= FA - FP + 120;// sensor 6
    FormGrafico.Series6.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[7,1]);
    FP:= StrToFloat(StringGrid1.Cells[7,Linha]);
    Resultado:= FA - FP + 140;// sensor 7
    FormGrafico.Series7.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[8,1]);
    FP:= StrToFloat(StringGrid1.Cells[8,Linha]);
    Resultado:= FA - FP + 160;// sensor 8
    FormGrafico.Series8.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[9,1]);
    FP:= StrToFloat(StringGrid1.Cells[9,Linha]);
    Resultado:= FA - FP + 180;// sensor 9
    FormGrafico.Series9.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[10,1]);
    FP:= StrToFloat(StringGrid1.Cells[10,Linha]);
    Resultado:= FA - FP + 200;// sensor 10
    FormGrafico.Series10.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[11,1]);
    FP:= StrToFloat(StringGrid1.Cells[11,Linha]);
    Resultado:= FA - FP + 220;// sensor 11
    FormGrafico.Series11.AddXY(Tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[12,1]);
    FP:= StrToFloat(StringGrid1.Cells[12,Linha]);
    Resultado:= FA - FP + 240;// sensor 12
    FormGrafico.Series12.AddXY(Tempo,Resultado);
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Procedimento utilizado para montar um grafico com os dados da StringGrid
procedure TFormPrincipal.MontaGraficoOffLine(linhas: Integer);
var
  i: integer;
  Resultado, FA, FP, tempo: Real;
```

```
begin
  for i:=1 to linhas do
  begin
    tempo:= StrToFloat(StringGrid1.Cells[0,i]);
    FA:= StrToFloat(StringGrid1.Cells[1,1]);
    FP:= StrToFloat(StringGrid1.Cells[1,i]);
    Resultado:= FA - FP + 20;// sensor 1
    FormGraficoOffLine.Series1.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[2,1]);
    FP:= StrToFloat(StringGrid1.Cells[2,i]);
    Resultado:= FA - FP + 40;// sensor 2
    FormGraficoOffLine.Series2.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[3,1]);
    FP:= StrToFloat(StringGrid1.Cells[3,i]);
    Resultado:= FA - FP + 60;// sensor 3
    FormGraficoOffLine.Series3.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[4,1]);
    FP:= StrToFloat(StringGrid1.Cells[4,i]);
    Resultado:= FA - FP + 80;// sensor 4
    FormGraficoOffLine.Series4.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[5,1]);
    FP:= StrToFloat(StringGrid1.Cells[5,i]);
    Resultado:= FA - FP + 100;// sensor 5
    FormGraficoOffLine.Series5.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[6,1]);
    FP:= StrToFloat(StringGrid1.Cells[6,i]);
    Resultado:= FA - FP + 120;// sensor 6
    FormGraficoOffLine.Series6.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[7,1]);
    FP:= StrToFloat(StringGrid1.Cells[7,i]);
    Resultado:= FA - FP + 140;// sensor 7
    FormGraficoOffLine.Series7.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[8,1]);
    FP:= StrToFloat(StringGrid1.Cells[8,i]);
    Resultado:= FA - FP + 160;// sensor 8
    FormGraficoOffLine.Series8.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[9,1]);
    FP:= StrToFloat(StringGrid1.Cells[9,i]);
    Resultado:= FA - FP + 180;// sensor 9
    FormGraficoOffLine.Series9.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[10,1]);
    FP:= StrToFloat(StringGrid1.Cells[10,i]);
    Resultado:= FA - FP + 200;// sensor 10
    FormGraficoOffLine.Series10.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[11,1]);
    FP:= StrToFloat(StringGrid1.Cells[11,i]);
    Resultado:= FA - FP + 220;// sensor 11
    FormGraficoOffLine.Series11.AddXY(tempo,Resultado);
    FA:= StrToFloat(StringGrid1.Cells[12,1]);
    FP:= StrToFloat(StringGrid1.Cells[12,i]);
    Resultado:= FA - FP + 240;// sensor 12
```

```
    FormGraficoOffLine.Series12.AddXY(tempo,Resultado);
  end;
end;
//---------------------------------------------------------------------------


//                Funcoes utilizados pelo programa                //

//---------------------------------------------------------------------------
// Funcao utilizada para converter uma string para um numero binario
function TFormPrincipal.Conversao(palavra: String): String;
var
  palavra_convertida, aux: string;
  i: integer;
begin
  palavra_convertida:= '';// palavra_convertida esta recebendo vazio

  for i:= 7 to 10 do // inicio do loop para converter a string para binario
  begin
    if palavra[i] = '0' then begin
      aux:='0000'
    end// if
    else if palavra[i] = '1' then begin
      aux:='0001'
    end// else if
    else if palavra[i] = '2' then begin
      aux:='0010'
    end// else if
    else if palavra[i] = '3' then begin
      aux:='0011'
    end// else if
    else if palavra[i] = '4' then begin
      aux:='0100'
    end// else if
    else if palavra[i] = '5' then begin
      aux:='0101'
    end// else if
    else if palavra[i] = '6' then begin
      aux:='0110'
    end// else if
    else if palavra[i] = '7' then begin
      aux:='0111'
    end// else if
    else if palavra[i] = '8' then begin
      aux:='1000'
    end// else if
    else if palavra[i] = '9' then begin
      aux:='1001'
    end// else if
    else if palavra[i] = 'A' then begin
      aux:='1010'
```

```
    end// else if
    else if palavra[i] = 'B' then begin
      aux:='1011'
    end// else if
    else if palavra[i] = 'C' then begin
      aux:='1100'
    end// else if
    else if palavra[i] = 'D' then begin
      aux:='1101'
    end// else if
    else if palavra[i] = 'E' then begin
      aux:='1110'
    end// else if
    else if palavra[i] = 'F' then begin
      aux:='1111'
    end; // fim do ultimo else if

    // if utilizado para realizar a concatenacao dos numeros binarios
    if palavra_convertida = '' then
      palavra_convertida:= aux
    else
      palavra_convertida:= palavra_convertida + aux
  end; // fim do loop de conversao para binario

  Result:= palavra_convertida;
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Funcao utilizada para somar os caracteres do numero binario
function TFormPrincipal.Somador(palavra: String): Integer;
var
  i, somador, aux: integer;
begin
  somador:= 0;
  for i:=1 to 16 do
  begin
    aux:= StrToInt(palavra[i]);
    somador:=somador + aux;
  end; // fim do for
  Result:= somador;
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Funcao utilizada para somar o tempo de chegada dos dados dos sensores
function TFormPrincipal.SomaTempo(tempo: Real): Real;
var
  aux : string;
  intervalo: real;
begin
```

```
  if StringGrid1.RowCount = 2 then
    Result:= tempo
  else begin
    aux:= StringGrid1.Cells[0,StringGrid1.RowCount - 2];
    intervalo:= StrToFloat(aux);
    Result:= intervalo + tempo;
  end;// else
end;
//---------------------------------------------------------------------------
end.
```

```
unit UnitConfiguracao;

interface

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

type
  TFormConfiguracao = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    ComboFluxo: TComboBox;
    ComboBitsParada: TComboBox;
    ComboNumeroBits: TComboBox;
    ComboTransmissao: TComboBox;
    ComboComunicacao: TComboBox;
    ComboParidade: TComboBox;
    BotaoEstabelcerConexao: TButton;
    VaComm1: TVaComm;

    procedure FormCreate(Sender: TObject);
    procedure BotaoEstabelcerConexaoClick(Sender: TObject);

  private
    { Private declarations }
  public
    procedure FluxoControle(chave: boolean);
    function AquisicaoDados(mensagem: string): string;
  end;

var
  FormConfiguracao: TFormConfiguracao;

implementation
{$R *.DFM}

//---------------------------------------------------------------------------
procedure TFormConfiguracao.FormCreate(Sender: TObject);
begin
   with ComboComunicacao do
      ItemIndex:= Items.IndexOF('2');
   with ComboTransmissao do
      ItemIndex:= Items.IndexOF('br9600');
   with ComboNumeroBits do
```

```
    ItemIndex:= Items.IndexOF('db8');
  with ComboBitsParada do
    ItemIndex:= Items.IndexOF('sb1');
  with ComboParidade do
    ItemIndex:= Items.IndexOF('paNone');
  with ComboFluxo do
    ItemIndex:= Items.IndexOF('fcRtsCts');
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Procedimento utilizado para realizar a conexao da Microbalanca com o
// computador
procedure TFormConfiguracao.BotaoEstabelcerConexaoClick(Sender: TObject);
begin
  VaComm1.PortNum:= (ComboComunicacao.ItemIndex + 1);
  VaComm1.Baudrate:= TVaBaudrate(ComboTransmissao.ItemIndex);
  VaComm1.DataBits:= TVaDataBits(ComboNumeroBits.ItemIndex);
  VaComm1.StopBits:= TVaStopBits(ComboBitsParada.ItemIndex);
  VaComm1.Parity:= TVaParity(ComboParidade.ItemIndex);
  VaComm1.FlowControl:= TVaFlowControl(ComboFluxo.ItemIndex);

  VaComm1.Open;
  close;
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Procedimemto para realizar o chaveamento na porta serial da Microbalanca
//para que possa ser possivel realizar a aquisicao de todos os dados medidos
procedure TFormConfiguracao.FluxoControle(chave: boolean);
var
  aux: string;
begin
  if chave = true then begin
    VaComm1.RTSControl:= rtsEnable;
    sleep(150);
    aux:= VaComm1.ReadText;
  end
  else begin
    VaComm1.RTSControl:= rtsDisable;
    sleep(150);
    aux:= VaComm1.ReadText;
  end;
end;
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
// Funcao utilizada para realizar a aquisicao dos dados medidos pela Microbalanca
function TFormConfiguracao.AquisicaoDados(mensagem: string): string;
begin
```

```
  // Envia uma mensagem para a Microbalanca requisitanto seus dados
  VaComm1.WriteText(mensagem);
  sleep(150);
  // Retorna os dados requisitos
  Result:= VaComm1.ReadText;
end;
//---------------------------------------------------------------------------
end.
```