PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

SISTEMA DE MONITORAMENTO DIGITAL DE GRANDEZAS ELÉTRICAS

ÍGOR ABRAHAO PARANHOS

Dissertação de Mestrado
Porto Alegre, Agosto de 2007.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ÍGOR ABRAHÃO PARANHOS

SISTEMA DE MONITORAMENTO DIGITAL DE GRANDEZAS ELÉTRICAS

Porto Alegre
2007

ÍGOR ABRAHÃO PARANHOS

SISTEMA DE MONITORAMENTO DIGITAL DE GRANDEZAS ELÉTRICAS

Dissertação submetida à Pontifícia Universidade Católica do Rio Grande do Sul como parte dos requisitos para a obtenção do grau de mestre em engenharia elétrica.

Orientador : Prof. Dr. Fausto Bastos Líbano

Porto Alegre
2007

ÍGOR ABRAHÃO PARANHOS

SISTEMA DE MONITORAMENTO DIGITAL DE GRANDEZAS ELÉTRICAS

Dissertação submetida à Pontifícia Universidade Católica do Rio Grande do Sul como parte dos requisitos para a obtenção do grau de mestre em engenharia elétrica.

Aprovado em _____ de _____ de _____

Banca Examinadora:

_____
Prof. Dr. Daniel Ferreira Coutinho – Pucrs

_____
Prof. Dr. Luis Alberto Pereira – Pucrs

_____
Prof. Dr. Samir Ahmad Mussa – Ufsc

AGRADECIMENTOS

Agradeço primeiramente à minha noiva, pelo apoio e companheirismo, aos meus pais, pelo contínuo incentivo ao crescimento intelectual, ao meu orientador, pela oportunidade, ensinamentos e amizade, aos professores colaboradores, que contribuíram com conhecimentos técnicos, e aos meus colegas de mestrado, que neste período foram meus irmãos, pelo suporte técnico e emocional.

RESUMO

A alta tecnologia encontrada nos equipamentos modernos tem colaborado no controle de processos industriais, aumentando a produtividade e a confiabilidade em diversos setores, facilitando a elaboração de produtos de melhor acabamento. Entretanto, a utilização destes equipamentos em linhas de produção aumenta consideravelmente a susceptibilidade do sistema aos distúrbios na qualidade de energia, devido à micro-eletrônica associada. O aparecimento de cargas não-lineares tem afetado diretamente os sinais de corrente e tensão, de forma a deixar o ambiente eletricamente poluído. A preocupação com os níveis da qualidade da energia elétrica vem aumentando bastante nos últimos anos, tanto para consumidores como para as concessionárias. Em meio a isto, sistemas de monitoramento da qualidade da energia foram surgindo, e de acordo com o desenvolvimento tecnológico, sendo acrescidos de novas funções. Porém, muitos se tornaram de custo elevado, de forma a não serem acessíveis a empresas de recursos limitados. Sistemas mais simples também passaram a ser comercializados, todavia, sem determinadas ferramentas, como banco de dados.

Este trabalho apresenta a implementação de um sistema de monitoramento digital de grandezas elétricas, que visa aliar o baixo custo à um recurso de banco de dados (para o armazenamento da informações), e à visualização remota dos mesmos, através de uma página da internet. Para isso, foi desenvolvido um hardware para fazer a leitura trifásica de tensão e corrente, baseado no micro-controlador MSP430f169; um software para a aquisição dos dados, apresentação de cálculos, gráficos e armazenagem, desenvolvido sob a plataforma Delphi 5; um banco de dados (em PostgreSQL) e uma interface WEB, fazendo-se uso da linguagem PHP.

Também, é proposto um estudo a respeito do assunto, de forma a caracterizar alguns sistemas e recursos encontrados nos produtos comerciais, assim como uma revisão bibliográfica, apontando as tendências para essa área.

ABSTRACT

The technology associated to modern electronic devices is helping the control over industrial processes, which has increased the productive and the reliability on many sectors, resulting in the production of high quality products. However, the usage of these equipments over production lines makes the system more susceptible to power quality events, because of the associated micro-electronics. The appearance of non-linear loads has been affected directly the voltage and current signals, making the environment electrically polluted. The concern with the power quality has significantly increased on the past few years, mainly due to the consumers aiming the avoidance of nonprogrammable stops in the production line (due to bad functioning of equipments), and also due to the concessionaires of electric energy desiring an adequate control over the system. Meanwhile, the power monitoring systems have appeared in the market with improved functions, mostly because of technology development. However, many of these monitoring systems are expensive, mainly in economies of developing countries, making harder for small companies to acquire one. Less expensive systems are available in the market, but with no capacity of database recourses.

This work proposes the implementation of low cost electrical digital monitoring systems with database resources. In addition, the proposed monitoring system is capable of remote access to the database through an internet web page. To this end, hardware to acquire three-phase voltage and current signals and software to download the data are developed, where the software is capable of data processing, mathematical calculus, producing data graphics for analysis and to manage the information for the WEB interface. In addition, the state of the art in electric monitoring systems is revised in order to characterize the functionalities and resources of available commercial products, as well as a bibliographic review to give a picture of the tendencies in this area.

LISTA DE ILUSTRAÇÕES

LISTAS DE TABELAS

LISTA DE SIGLAS

AD – Analógico – Digital

ASP - Active Server Pages

ARM - Advanced RISC Machines

ANSI - American National Standards Institute

ASCII - American Standard Code for Information Interchange

BD – Banco de Dados

bps – Bits por Segundo

CBEMA - Computer and Business Equipment Manufacturers Association

CGI - Commom Gateway Interface

COMTRADE - Common Format for Transient Data Exchange

CRC - Cyclic Redundancy Check

DB9 – Conector da porta serial

DER – Diagrama Entidade-Relacionamento

DFD – Diagrama de Fluxo de Dados

DHT – Distorção Harmônica Total

DLL - Dynamic-Link Library

ER – Entidade-Relacionamento

EN - European Committee for Standardization

FPGA - Field-Programmable Gate Array

FTP - File Transfer Protocol

GPS - Global Positioning System

HI-FI – High Fidelity

HTML - Hypertext Markup Language

HTTP - Hypertext Transfer Protocol

IEC - International Electrotechnical Commission

IEEE - Institute of Electrical and Electronics Engineers

I/O – Input/Output

IP – Internet Protocol

ISDN - Integrated Services Digital Network

ITI - Information Technology Industry Council

Kbps – Kilo Bits por Segundo

LAN - Local Area Network

Mbps – Mega Bits por Segundo

MIPS - Milhões de Instruções Por Segundo

MS – Microsoft

ODBC - Open Database Connectivity

PC – Personal Computer

PDA – Personal Digital Assistant

PHP – Personal Home Page

PLL - Phase-Locked Loop

PPP - Point-to-Point Protocol

QE – Qualidade de Energia

QEE – Qualidade de Enegia Elétrica

RAM – Random Access Memory

RISC - Reduced Instruction Set Computer

RMS – Root Mean Square

RS232 – Recomended Standard 232

RS485 – Recomended Standard 485

RTU – Remote Terminal Unit

RTS – Request to Send

RX – Recepção de dados pelo canal de comunicação

SFW – Select From Where

SGDB - Sistema Gerenciador de Banco de Dados

SQL – Structured Query Language

SRAM – Static Random Access Memory

STFT - Short Time Fourier Transform

TC – Transformador de Corrente

TCP – Transmission Control Protocol

THD – Total Harmonic Dirtortion

TI – Tecnologia da Informação

TP – Transformador de Potencial

TTL - Transistor–Transistor Logic

TX – Envio de dados pelo canal de comunicação

UART - Universal Asynchronous Receiver/Transmitter

URL - Uniform Resource Locator

VCC – Pólo positivo da alimentação

WWW – World Wide Web

LISTA DE SÍMBOLOS

A – Ampère

V – Volt

$ - Dolares

TA,TB,TC – Tensão nas fases “A”, “B” e “C”, respectivamente

VA,VB,VC – Tensão nas fases “A”, “B” e “C”, respectivamente

CA,CB,CC – Corrente nas fases “A”, “B” e “C”, respectivamente

pu – Por Unidade

GHz – Giga Hertz

Hz – Hertz

m – Metros

s - Segundos

SUMÁRIO

1 INTRODUÇÃO

1.1 Generalidades

O advento da energia elétrica foi o que propiciou o desenvolvimento humano se dar de forma tão impressionante nos últimos anos. A utilização da mesma possibilitou que processos industriais se dessem de forma mais rápida e precisa, que a informação percorresse distâncias mais longas, em uma velocidade extraordinária, que a noite tivesse luz e vida, dentre muitas outras facilidades.

Até pouco tempo atrás, definia-se o objetivo de um sistema elétrico como: “transportar a energia elétrica de uma fonte geradora até os terminais elétricos dos equipamentos, mantendo a tensão em certos limites” [8]. Por décadas esta foi a concepção para a utilização da energia elétrica. Confiabilidade e qualidade raramente eram assuntos relevantes. As máquinas em geral eram mais simples e robustas, sendo insensíveis à pequenas variações na tensão. Flutuações na tensão oriundas da rede pública não eram relevantes no objetivo funcional.

Este panorama vem mudando desde os anos 80, período em que começaram a aparecer máquinas e dispositivos micro-processados, controlados por semicondutores de potência, os quais possibilitaram a automatização dos processos e a confecção de produtos mais bem acabados, numa velocidade muito maior e com um custo menor.

Alguns efeitos nocivos à qualidade da energia começaram a surgir, como distorções nas formas de onda de tensão e corrente (que deveriam ser puramente senoidais), impulsos, interferências eletromagnéticas, sobre-tensão, sub-tensão, interrupções transitórias (outages), cintilação (flicker), variações na freqüência e harmônicas. A explicação de cada um destes distúrbios pode ser encontrada nas referências [8],[24],[27],[28],[48],[58],[60] e [84].

Estes fenômenos estão associados principalmente à cargas não lineares no sistema, como por exemplo circuitos retificadores (controlados ou não), lâmpadas de descarga, monitores de computador, aparelhos de solda, fornos a arco e muitas outras cargas eletrônicas que cada vez são mais comuns e necessárias às industrias e residências.

Entretanto, apesar destas máquinas modernas de alta tecnologia terem melhorado o desempenho de diversos setores produtivos, as linhas de produção ficaram mais susceptíveis a falhas e paradas, levando-se em conta que estes equipamentos são sensíveis a distúrbios na forma de onda dos sinais fornecidos, devido à microeletrônica associada.

Efeitos como a má interpretação dos dados ou a reinicialização indesejada de computadores industriais estão vinculados à poluição na rede elétrica, oriunda da presença de outros equipamentos com comportamento não linear, operando em paralelo com equipamentos susceptíveis a interferências. Os custos relacionados com a parada e/ou perda de produção de um produto/equipamento e com a reinicialização de uma linha de produção trazem prejuízos significativos para as partes envolvidas.

Em meio a este panorama, a “qualidade da energia” se tornou um assunto de extrema importância para a estratégia competitiva e lucratividade das empresas, pois estas passam a evitar gastos desnecessários ao não ter perda nenhuma no processo produtivo, ou ao não pagar multa pelo baixo fator de potência, por exemplo.

Por isso, as empresas estão começando a valorizar a utilização de sistemas de supervisão e suas implicações como um todo, e não apenas como um insumo isolado do processo produtivo. Neste nível de amadurecimento, o gerenciamento energético se aproximou bastante da “tecnologia da informação” (TI), deixando de ser focado e confinado exclusivamente aos equipamentos e a questões energéticas, como consideradas tradicionalmente. Atualmente, não se busca apenas o estado da arte em termos de algoritmos e técnicas de otimização. O foco principal é a integração das informações de produção, energia e gestão.

Quando se fala em “qualidade de energia” vale o mesmo conceito. Uma empresa moderna não aceita mais soluções isoladas, propostas por um ou outro departamento. Todos os setores devem compartilhar informações e interagir com elas de forma simples e rápida. Confirma-se cada vez mais a tendência de que os passos tecnológicos caminhem em boa parte para a área de software, a partir de uma plataforma de hardware multifuncional.

Na última década, com o surgimento dos medidores eletrônicos baseados em sistemas digitais, possibilitou-se que os equipamentos de gerenciamento passassem a oferecer a monitoração de todas as grandezas em tempo real para múltiplos usuários, permitindo generalizar ainda mais estas informações e associar à questão energética os fatores da qualidade de energia.

A tendência para os próximos anos é de que a evolução dos sistemas de gerenciamento de energia será tão grande e integradora quanto foi na última década. Por esta razão, as empresas hoje não pretendem apenas acompanhar tensão e corrente em tempo real, mas também o número de vezes e a freqüência com que ocorreram interrupções no fornecimento da energia, assim como analisar as tensões entregues pela concessionária e as correntes consumidas, em busca de transientes, harmônicas e variações na amplitude dos sinais elétricos.

De forma geral, o que um consumidor deseja é comprar energia barata e que atenda a seus requisitos mínimos em termos de oscilações e interrupções. Ainda mais atualmente, com o surgimento do “consumidor livre” e do interesse das concessionárias em manter os clientes, a questão da qualidade de energia vem sendo impulsionada fortemente.

Em meio a este ambiente, mais recentemente foram sendo desenvolvidos equipamentos digitais que são capazes de realizar a leitura dos sinais elétricos de interesse, com o objetivo de fornecer ao usuário informações das condições energéticas, como consumo e a qualidade da energia, por exemplo. E com a evolução da eletrônica e do poder computacional dos micro-controladores atuais, foram surgindo sistemas de monitoramento da energia cada vez mais sofisticados, os quais proporcionam informações e ferramentas avançadas.

Recursos como comunicação de dados sem fio, armazenamento de informações em banco de dados e visualização remota via internet, alerta de eventos por e-mail ou mensagem de telefone celular podem ser citados como aspectos modernos que auxiliam no gerenciamento do negócio. Entretanto, todos colaboram para o ajuste do custo do sistema.

Houve então uma divisão dos sistemas de monitoramento em termos de recursos e custos: os mais avançados, cujo custo é mais elevado, e os mais simples, com a proposta de serem distribuídos no mercado, sendo acessíveis a empresas de recursos limitados.

E é neste intervalo que este trabalho propõe se enquadrar. Assim, foi desenvolvido um sistema de monitoramento digital de grandezas elétricas, onde o diferencial é o serviço de banco de dados, para o armazenamento dos dados e consulta remota pela internet, aliado ao baixo custo do sistema, justificado pela utilização de alguns softwares livre para o desenvolvimento dos aplicativos, e pelo baixo custo dos componentes do hardware.

1.2 Estado da Arte

Observa-se no meio científico o crescente número de artigos vinculados ao assunto “qualidade de energia”, onde muitos são referentes a “medidores eletrônicos”, “qualímetros” e “sistemas de monitoramento da energia”. Entre eles, uma boa parte está vinculada a estratégias para monitoramento, métodos para os cálculos das grandezas elétricas, a vantagem da utilização de banco de dados com acesso via internet e a importância da centralização dos dados. Além destes, outros apresentam métodos para interpretação e análise das informações elétricas, e de forma mais desafiadora, a aplicação de inteligência artificial para a busca de distúrbios. Entre estes aspectos citados, outros também são encontrados em artigos deste assunto, mas observa-se que grande parte enfatiza o ganho material e financeiro com a monitoração da qualidade da energia, ressaltando a importância dessa atividade.

De forma mais específica, os artigos [3],[20],[22],[23],[38],[41],[52],[56],[60], [80],[84],[86],[87] e [89] apresentam um panorama atual do assunto “qualidade de energia” e dos “sistemas de monitoramento”, apresentando as vantagens da utilização deste último, como o ganho financeiro, o controle do sistema produtivo e a integração dos dados elétricos com outras informações de gestão.

Na referência [87], são encontrados métodos para a implementação do hardware, incluindo o tratamento dos sinais, os algoritmos para o cálculo das grandezas elétricas e a comunicação dos dados. O artigo [63] defende a utilização do “ModBus RTU” RS-485 para a

transferência das informações, devido larga utilização desse protocolo no meio industrial. No artigo [13], foi utilizado o “FieldBus”, também por utilizar um barramento industrial para a comunicação de dados. Por outro lado, os artigos [20] e [87] defendem a utilização da rede “Ethernet” para a transferência de dados entre o medidor e o software gerenciador, devido à confiabilidade e à taxa de transmissão muitas vezes maior que na comunicação serial.

Ainda a respeito da comunicação de dados, pode-se citar os artigos [15] e [40], que incluem tecnologias sem fio para essa tarefa, em um sistema de monitoramento.

Quanto aos algoritmos de cálculo, o artigo [57] propõe a utilização da “Short Time Fourier Transform” (STFT) para a detecção do início e do término de eventos como “sags” e “swells”, e do “Wavelet” para medição com alta resolução de variações na freqüência de sinais não-estacionários, dependentes do tempo. Em [47] é proposto um novo algoritmo, chamado de “Least Squares Algorithm”, que é um método para se calcular o valor RMS de tensão e corrente, e o ângulo de fase de cada componente harmônica diretamente de uma medida de tensão e corrente.

O artigo [87] em especial, defende que os medidores não precisam ter “display” nem algoritmos para os cálculos das grandezas elétricas, pois todas informações podem ser obtidas “online” no software.

Nos artigos [87] e [89] são encontrados assuntos referentes à utilização de banco de dados para o armazenamento das informações, e de página da internet para acesso a esses registros, enfatizando as vantagens da utilização destes recursos e apresentando estratégias de implementação. Tais artigos falam também sobre o protocolo TCP/IP.

Os artigos [3],[23],[38] e [39] abordam o fluxo das informações e métodos de integração do sistema, enfatizando estratégias de monitoramento para uma busca eficaz aos eventos de qualidade de energia, e a vantagem na correção dos mesmos.

Sobre o software, análise e interpretação de dados e busca por eventos, as referências [20],[41],[23],[39],[41],[52],[86],[57],[86] e [80] podem ser citadas. O artigo [39] ressalta os obstáculos e as estratégias para se obter uma precisa localização de distúrbios. Em [52] são encontrados parâmetros relevantes a serem medidos, assim como os gráficos a serem mostrados.

A respeito da organização dos arquivos de oscilografia, pode-se associar os artigos [37] e [86], que inclusive se reportam ao padrão “COMTRADE”.

Por final, são citadas as referências [30],[33],[34],[36] e [88] no que tange a inclusão de tecnologias de “inteligência artificial” para a análise e interpretação dos dados.

1.3 Organzacao da Dissertação

Esta dissertação tem como proposta o desenvolvimento de um “sistema de monitoramento digital de grandezas elétricas”, cujo objetivo é aliar o baixo custo dos módulos integrantes (hardware/software) aos recursos de armazenamento (banco de dados) e consulta dos dados através da internet.

Foi também realizada uma revisão bibliográfica do assunto “sistemas de medição e monitoramento da qualidade da energia elétrica”, levantando-se as vantagens da utilização de um sistema referente, assim como os recursos mais comumente encontrados. Serão também sinalizadas algumas tendências tecnológicas. A seguir, será explicado como os assuntos se dividem nos capítulos.

O capítulo 2 faz uma abordagem a respeito dos sistemas comerciais de monitoramento de energia, disponíveis no mercado, refletindo sobre os recursos e limitações que os mesmos podem apresentar, assim como uma breve descrição das normas existentes para a padronização dos métodos de medição e monitoramento.

O capítulo 3 mostra como o software deste o projeto foi desenvolvido, enfocando a importância da utilização de técnicas de engenharia de software para a elaboração de um aplicativo organizado e flexível.

No capítulo 4 são apresentados os desenvolvimentos e os resultados obtidos com o hardware do sistema (medidores), ressaltando-se como foram feitas as escolhas para os dispositivos componentes da estrutura, assim como a construção dos circuitos esquemáticos, placas de circuito impresso e o programa do micro-controlador.

O capítulo 5 descreve brevemente o projeto e o desenvolvimento do banco de dados e da interface de acesso remoto ao mesmo (página da internet), de forma a ilustrar os benefícios da utilização destes recursos.

Por último, no capítulo 6 são apresentadas as considerações finais a respeito do sistema desenvolvido, as possíveis implementações futuras, e também alguns comentários sobre as tendências dos sistemas de monitoramento de energia.

2 MONITORAMENTO DA QUALIDADE DA ENERGIA

Qualidade de energia não tem uma definição padrão. Se o usuário conseguir executar a tarefa desejada com a energia disponível, então pode-se considerar, genericamente, que a qualidade de energia é boa [38].

Mas a simples disponibilidade de energia é apenas um fator determinante da qualidade da mesma, porque por outro lado, quando algum problema ocorre impedindo o perfeito funcionamento de uma máquina, por exemplo, associa-se a isto um problema de qualidade de energia. [38].

Em geral, a energia recebida é bastante boa, de uma perspectiva elétrica. Com a exceção de problemas climáticos e acidentes eventuais, a maioria dos problemas de qualidade de energia são resultado da interação entre cargas conectadas à rede elétrica. A Figura 2.1 mostra a origem dos problemas de qualidade de energia, analisada sob a ótica do consumidor, segundo [27].

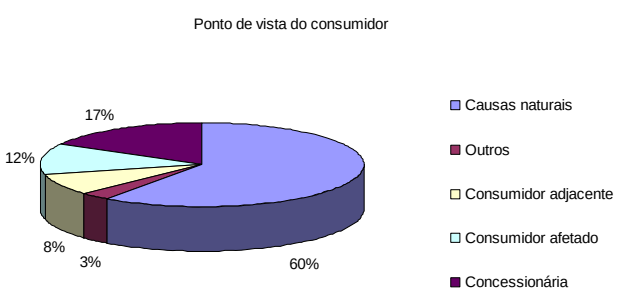


Figura 2.1 – Origem dos problemas de qualidade de energia.

O impacto adverso que algumas cargas causam na rede elétrica acarreta em muitos problemas na qualidade da energia, e para um melhor entendimento desses eventos, o monitoramento da qualidade da energia veio a ser uma atividade muito importante.

Os sistemas de energia estão em constante mudança. As cargas dos usuários de energia vão desde simples dispositivos lineares, como cargas resistivas ou lâmpadas incandescentes, até cargas não lineares, vinculadas a dispositivos mais sofisticados, como por exemplo, computadores industriais, sistemas de acionamento elétrico ou lâmpadas fluorescentes compactas, o que tem criado a necessidade de se acompanhar de perto a energia da rede.

Consumidores de energia elétrica, comerciais e industriais vêm, cada vez mais, necessitando de um elevado grau de desempenho do fornecimento da energia por eles consumida. A crescente sofisticação dos processos de controle e a utilização de computadores e equipamentos sensíveis a variações na qualidade da energia manifestam essa necessidade. Fenômenos tais como harmônicos, transitórios impulsivos, variações de tensão de curta duração, entre outros, precisam ser monitorados quantitativamente para o conhecimento da qualidade da energia utilizada [3]. Para garantir uma produtividade e crescimento contínuos, o ambiente elétrico precisa ser puro e limpo.

A Tabela 2.1 mostra exemplos do nível de confiabilidade aceitável para alguns clientes de energia elétrica. Pode-se perceber que há situações onde a falta de energia é bastante crítica, e a qualidade da energia deve ser bastante alta, neste caso, falando-se da confiabilidade do fornecimento.

Tabela 2.1 – Exemplo de confiabilidade para alguns setores.

Número de 9´s	Confiabilidade	Quantidade de falta de energia, por ano	Aceitável para
3	99,9%	9 horas	Residências
4	99,99%	59 minutos	Fábricas
5	99,999%	5 minutos	Hospitais e aeroportos
6	99,99999%	32 segundos	Bancos
7	99,999999%	30 milisegundos	Mercados "online"

FONTE: KHAN, A. K. “Monitoring Power for the Future”.

A filosofia do monitoramento da qualidade da energia tem evoluído do simples objetivo de encontrar defeitos nos sinais elétricos, para um grande número de finalidades. Como exemplo, pode ser citado: a garantia de conformidade com o contrato ou com as normas;

determinação de práticas de prevenção de falhas; estabelecer estratégias para manutenção preditiva; obtenção de informações estatísticas para diversos fins; e garantia de operação de equipamentos dentro dos limites especificados pelos fabricantes.

Segundo [3], “a monitoração da qualidade de energia elétrica seguida da organização sistemática e automática de toda a informação adquirida ao longo de um período de monitoração, provê uma poderosa ferramenta de análise para auxiliar no planejamento, na operação e no projeto de sistemas de transmissão e distribuição de energia elétrica”.

Junto com a evolução do objetivo do monitoramento da energia, a tecnologia agregada ao sistema também o fez. Os equipamentos incorporaram avanços na área da comunicação, aquisição de dados, tamanho físico e custo. Com as taxas de transmissão de dados aumentando consideravelmente, o desejo pelo refinamento na precisão das medições ocorreu naturalmente.

A Figura 2.2 busca caracterizar de uma forma esquemática a evolução (nos últimos anos) dos serviços e recursos para o monitoramento elétrico.

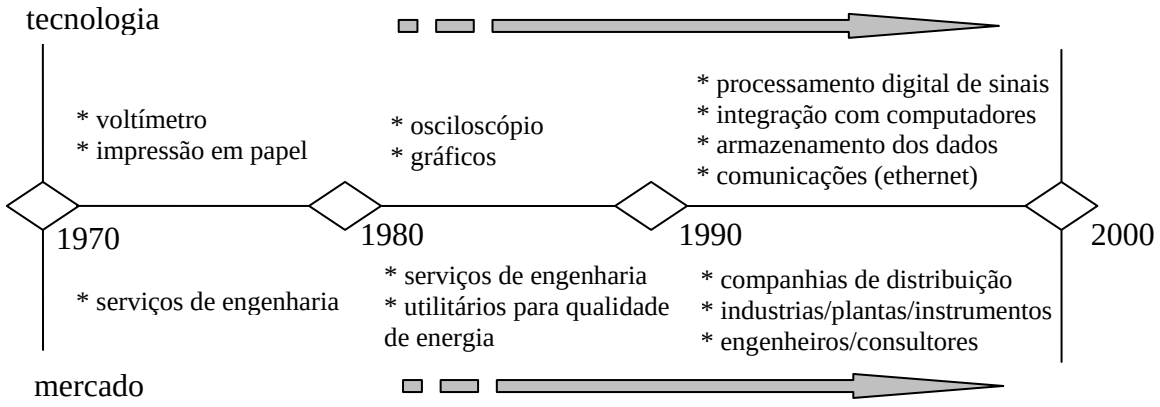


Figura 2.2 – Evolução dos serviços e equipamentos para monitoramento de energia.

Como se pode observar na Figura 2.2, o avanço no processamento digital de sinais, a evolução dos micro-controladores e o armazenamento de dados tem mudado a forma de se analisar a qualidade da energia, assim como a forma de apresentação dos resultados. Tecnologias mais antigas não tinham recursos computacionais para a realização de cálculos “online” e classificação dos distúrbios elétricos.

No princípio, o número de amostras por ciclo que se conseguia era de 16, o que dificultava a análise da forma de onda e a busca por distúrbios (principalmente os de pequeno período, como impulso ou “notch”), devido a este valor representar uma resolução muito baixa.

Hoje em dia, taxas de amostragem comumente utilizadas são 64, 128, 256, 512 e 1024 pontos por ciclo. A taxa de amostragem adequada em combinação com uma resolução de AD conveniente permite uma visualização fiel da forma de onda dos sinais elétricos. A Tabela 2.2 exemplifica as modificações dos sistemas antigos aos atuais.

Tabela 2.2 – Evolução dos recursos computacionais.

	Passado		Presente
Resolução do AD	10 bit AD		12 bit AD
Taxa de amostragem	14 amostras/ciclo		256 amostras/ciclo
Memória	papel / disquete		16Mb(RAM) / 40 Gb (HD)
Taxa de transmissão	1200 bps		100Mbps
Custo	$$$$		$$

FONTE: KHAN, A. K. “Monitoring Power for the Future”.

Devido à evolução das ferramentas componentes dos sistemas, o monitoramento contínuo dos sinais elétricos se tornou possível. E estes dados estrategicamente adquiridos são muito importantes para a determinação de onde e quando um evento ocorreu, conduzindo o usuário de energia a entender como eles podem ser evitados e/ou mitigados no futuro, atividade que certamente acarreta em diversos benefícios.

O monitoramento contínuo das harmônicas e o devido tratamento, por exemplo, evitam o aquecimento em transformadores, condutores ou disjuntores, levando a uma falha prematura.

Também, o contínuo conhecimento da potência utilizada e dos valores eficazes provém informações importantes para o planejamento de expansões futuras da planta industrial, de modo a saber se as sub-estações e os alimentadores estão adequados. Essas informações também são muito úteis quando há a necessidade da adição de novas cargas ou sistemas de potência, de forma a se fazer investimentos apropriados.

2.1 Elementos de um sistema de monitoramento

Dados são informações sem contexto. Para que os pontos representativos dos sinais elétricos possam ser utilizados para a tomada de conclusões, eles devem ser tratados e compartilhados. Muitos aparelhos medidores de energia coletam dados continuamente a altas taxas de amostragem e os armazenam em memórias. Entretanto, apenas um pequeno percentual é utilizado para análise, e é essencial saber destacar o que é relevante para a interpretação e tomada de decisão [38].

Para a adequada integração e fácil acesso às informações de um sistema de monitoramento de energia, quatro elementos são desejáveis: hardware, software, banco de dados e página da internet. Destes, o primeiro é sempre indispensável. Os outros podem não existir, apesar de serem cada vez mais importante para a manipulação e integração dos dados.

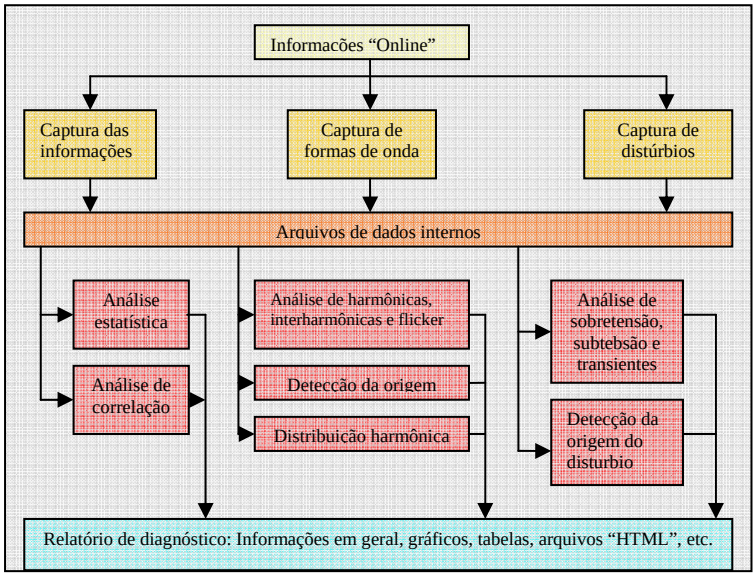


Figura 2.3 - Características desejáveis para um sistema diagnosticador da qualidade da energia.

A interação organizada dos elementos acima citados pode gerar informações bastante valiosas para análise da qualidade da energia e, se integradas com outros dados de produção e custos, podem colaborar bastante para o sucesso do negócio. A Figura 2.3, retirada da referência [19], mostra as características desejáveis para um sistema diagnosticador da qualidade da energia.

2.1.1 Hardware

Os instrumentos para a verificação da qualidade da energia tem tido grandes avanços nos últimos anos. Conforme comentado anteriormente, características como a taxa de amostragem, taxa de transmissão e resolução do “AD” evoluíram consideravelmente. Entretanto, isto não significa que todos medidores digitais modernos existentes utilizam o máximo destes recursos, pois existem diversas classes: de medidores mais básicos aos mais elaborados.

Mas, de qualquer forma, a grande maioria dos equipamentos tem o mesmo princípio de funcionamento: um micro-controlador programado para realizar aquisições digitais dos sinais (os conversores analógico-digital podem ser internos ou externos), estruturando os pontos em vetores, representativos das formas de onda dos sinais elétricos. E como os “ADs” lêem sinais de baixa tensão/corrente, existem “TPs” e “TCs”, que significam transformador de potencial e de corrente, respectivamente, assim como os circuitos para condicionar o sinal (atenuação e “offset”) e para proteção.

Segundo [38], quando se deseja escolher um medidor digital para o monitoramento de energia, deve-se considerar os seguintes itens:

- Armazenamento das informações elétricas adquiridas, mesmo sob a falta de energia;
- Taxa de amostragem – deve ser suficiente de acordo com a análise que se deseja fazer (valor RMS, energias, nível de harmônicas);
- Fácil utilização, configuração e programação;
- Comunicação – de acordo com a aplicação. Atualmente é comum a rede “ethernet” por “TCP/IP”.
- Encapsulamento físico – dependendo da aplicação: portátil ou fixo.

Para a maioria dos usuários, o mais importante é que o medidor deva capturar os eventos de qualidade de energia com fidelidade, e sem que a memória ultrapasse a sua capacidade no período de monitoramento.

A seguir, será apresentada uma classificação dos medidores, de acordo com os recursos por ele oferecidos.

2.1.1.1 Medidores Básicos

Também conhecidos como de 1° geração, são os medidores de mais baixo custo. As informações oferecidas por ele são simples, como a leitura dos sinais de tensão, corrente, potências, energias e fator de potência. A precisão das respostas gira em torno de 0,5%, quando da utilização de “TCs” e “TPs” de igual ou maior precisão.

Os medidores básicos são tipicamente portáteis, e possuem um painel simples. Também economizam nas opções de comunicação (muitas vezes não tem porta de comunicação) e portas de entrada/saída. Normalmente são empregados no setor industrial para mostrar as medidas elétricas elementares [63],[87].

2.1.1.2 Medidores Intermediários

Estes medidores, também chamados de 2° geração, têm os mesmos recursos dos medidores básicos, e ainda a capacidade de medir a energia real e aparente demandada, podendo possuir também alguns recursos para a captação de eventos. Os recursos de comunicação e portas de entrada e saída também são mais avançados, apesar de a comunicação ser serial.

Basicamente, estes medidores servem para monitorar alimentadores de sub-estações, quando nenhum distúrbio elétrico é buscado.

2.1.1.3 Medidores Avançados

Os medidores avançados herdaram as características dos básicos e intermediários, somando as funções de captura dos eventos de qualidade de energia. São também conhecidos como medidores de 3° geração.

Em outras palavras, esses medidores têm, além de adquirir as formas de ondas dos sinais elétricos e realizar os cálculos dos parâmetros de performance, a capacidade de monitorar a rede e iniciar o armazenamento dos dados, a partir de um distúrbio elétrico. Os parâmetros para o disparo são inseridos pelo usuário. Por exemplo, os limites para o valor RMS de tensão podem estar configurados para mais ou menos 10% do valor nominal, o que significa que acima de 139.7V ou abaixo de 114.3V, para um valor nominal de 127V, o disparo ocorreria, e alguns ciclos antes e depois do evento seriam armazenados.

Esse tipo de medidor comumente oferece também recursos como alarmes e alerta de distúrbios, assim como entradas e saídas analógicas e digitais, memória de massa integrada, displays modernos e diversos tipos de comunicação (logo a frente será comentado com mais detalhes).

Ainda, esta classe de medidores pode hospedar uma página da internet (HTML). Para se visualizar os dados é preciso somente uma conexão com a internet, um “IP” fixo e um navegador [63]. Este recurso torna o medidor uma boa ferramenta para o monitoramento da qualidade da energia, entretanto o torna de custo elevado.

2.1.1.3.1 Memória de Massa

Os medidores mais avançados possuem a capacidade de armazenar grandes volumes de dados, que podem representar formas de onda, informações sobre distúrbios captados ou outras informações.

Existem diferenças entre os fabricantes no que se refere ao modo de armazenamento dos dados. Alguns medidores enviam os dados a um computador assim que possível. Outros, diferentemente, possuem uma grande capacidade de armazenamento de dados, e armazenam informações por um longo período de tempo (um mês, por exemplo), antes de transferi-las a outro armazenador de dados (outro computador, possivelmente com um software de análise dos dados).

A combinação da memória de massa com um software analisador das informações oferece maiores recursos ao sistema para não perder dados ou eventos, como paradas.

2.1.1.3.2 Comunicação de Dados

A escolha da forma de comunicação de dados do sistema é um fator muito importante para o projeto e desempenho do hardware e do software. O tipo de comunicação escolhida vai influenciar na necessidade do hardware ter ou não uma grande memória de massa, na quantidade de ciclos e pontos por ciclo transferida do medidor para o software, no número de medidores ligados à rede de comunicação e, no custo do sistema.

Atualmente, com o avanço tecnológico existente, diversas são as formas para se comunicar dados, podendo-se citar:

- Comunicação serial ponto a ponto – Também conhecido por padrão RS232, permite, como o nome já sugere, a comunicação serial entre dois dispositivos. A taxa de transmissão mais comum é de 19200 bps, e a distância (para essa taxa) é de 10m. A vantagem encontrada nela é a simplicidade de utilização, mas por outro lado, limitações como a distância máxima e o número de dispositivos são notáveis, assim como a velocidade de transmissão, que é bem inferior à de outras comunicações, como “Ethernet” [85].

- Comunicação serial multiponto – Entre outros existentes, cita-se o padrão RS485, onde, como o item anterior, a comunicação se dá de forma serial, com a diferença que o número máximo de dispositivos que podem se comunicar é de 32, e a extensão do barramento pode ser de 1200m (para um taxa de transmissão

de 100Kbps). Esta forma de comunicação é mais confiável que a do item anterior, devido à transmissão ser balanceada (ou diferencial), o que ajuda a anular os efeitos do deslocamento do nível do terra e dos ruídos que podem aparecer nas vias de transmissão. De acordo com a configuração física, é possível tornar a comunicação “half-duplex” ou “full-duplex” [77].

- Comunicação via infravermelho – Esta comunicação oferece a facilidade de não ser necessário fio para a transferência de dados. Isto é feito através da luz infra-vermelho que é emitida pelo transmissor, e recebida pelo receptor, sendo que a comunicação pode ser bi-direcional. E com a difusão dos computadores de mão (“PDAs”), os quais os mais avançados possuem comunicação infravermelho, assim como alguns “notebooks”, a coleta local de dados dos medidores através desta comunicação se torna prática. Entretanto, parâmetros como a taxa de transferência e distância entre os dispositivos podem limitar a utilização desta comunicação.

- Comunicação via Bluetooth - A tecnologia “Bluetooth” é, basicamente, um padrão para comunicação sem fio, porém de curto alcance. Ela funciona na freqüência de 2,45GHz, e permite que até 8 elementos formem uma rede, onde um é o mestre. Outra rede pode ser formada entre mais de uma dessas redes de 8 elementos, formando assim uma rede maior. A taxa de transmissão é de 723Kbps, o que oferece uma maior vazão de dados do que as comunicações citadas anteriormente. A grande vantagem dessa comunicação está na eliminação da utilização de cabeamento físico, possibilitando a comunicação do hardware com outro dispositivo “Bluetooth”, como um “palmtop” ou “notebook”, sem cabo nenhum, o que é muito vantajoso se o local é de difícil acesso. A desvantagem está no custo (mais elevado que as outras comunicações) e na utilização do protocolo, que é restrito às empresas detentoras da tecnologia. [1].

- Comunicação por Ethernet – Atualmente, este é o padrão mais comum para a formação de uma “LAN” (Local Area Network) entre dispositivos. Cada componente deve ter um hardware dedicado para a interface de rede, e a rede deve possuir um “HUB”, um “Switch” ou um “Roteador” para possibilitar a comunicação. A vantagem desse tipo de comunicação está na taxa de

transmissão (10 ou 100Mbps), que é bem superior às anteriores, e também pelo fato de ser uma tecnologia bem conhecida, atualmente. A desvantagem, em relação à comunicação RS485, é o custo, levando-se em conta o hardware integrado para a interface de rede, e também do concentrador de rede (citados acima).

Ainda poder-se-ia citar outras formas de comunicação possíveis, como “WI-FI”, “Wireless”, “Profibus”, “USB” entre outros.

2.1.1.4 Medidores Detectores de Transientes

Estes medidores de último tipo oferecem, além das características dos “avançados”, a detecção de transientes na faixa de “megahertz”. Eles capturam a forma de onda dos sinais elétricos e tem a capacidade de fazer a análise harmônica, muitas vezes até uma ordem elevada, como a 255°. Para oferecer essa precisão, taxas de amostragem da ordem de 512 amostras por ciclo são comuns. Taxas de amostragem altas são necessárias para a captação de eventos extremamente rápidos, de forma que se possa caracterizar a magnitude, o período e a polaridade, permitindo o diagnóstico da origem do evento [63].

Assim como os medidores avançados, essa classe possui diversos tipos de comunicação e de entradas / saídas, e é classificada como de 3° geração.

A Tabela 2.3 mostra as características das famílias de medidores acima descritas.

Tabela 2.3 – Características das famílias de medidores

Tipo de Medidor	**Básico**	**Intermediário**	**Avançado**	**Detectores de transientes**
Aplicação	Simples alimentadores	Alimentadores de sub-estação	Sub-estações principais	Entrada de alimentação
Medições				
Corrente, Tensão, Potências	X	X	X	X
Fator de Potência	X	X	X	X
Frequência	X	X	X	X
Energias	X	X	X	X
Harmônica de tensão e corrente		X	X	X
Fator de Deslocamento		X	X	X
Demanda de corrente e potência		X	X	X
Predição da demanda de potência		X	X	X
Qualidade de energia				
THD de tensão e corrente			X	X
Detecção de sobre/subtensão			X	X
Captura "onboard" da forma de onda			X	X
Captura da forma de onda para o PC			X	X
Alarme da distorção na forma de onda			X	X
Detecção de distúrbio na direção		X	X	X
Detecção de transientes				X
Entradas / Saídas				
Nenhum	X			
Saída KYZ		X	X	X
Menos de 3		X		
10 ou mais			X	X
Comunicação				
Nenhuma	X			
Proprietária		X		
ModBus RTU / RS232 / RS485		X	X	X
Ethernet par trançado		X	X	X
Ethernet fibra ótica			X	X
Página da WEB / Alerta por email			X	X
Recursos Avançados				
Relógio sincronizado por GPS			X	X
Funções programáveis / Lógica			X	X
Memória de massa "onboard"		X	X	X
Alta velocidade de gravação de eventos			X	X
Registro das informações		X	X	X

Fonte: PAULI, S. KRAHL, B.O. LEUSCHNER, B. “Know Your Power”.

2.1.2 Software

O software de um sistema de monitoramento pode ter diversas interfaces e recursos. Isto depende da proposta do sistema, ou seja, mais completo ou não, o que influencia diretamente nos custos.

Entretanto, independentemente da complexidade do software, uma boa organização estrutural do mesmo é sempre desejável. Isto ajuda o aplicativo a ser flexível e adaptável à modificações estruturais, e inclusive no processamento dos diferentes tipos de eventos de qualidade de energia [18]. Também, o software deve oferecer uma interface amigável, ser de fácil manipulação e confiável. [63].

Basicamente, o software deve se comunicar com o hardware e com o banco de dados (se presente), de forma a gerenciar o fluxo de informações do sistema, e apresentar resultados em forma de tabelas, relatórios e gráficos, assim como também pode enviar alertas de eventos por e-mail ou mensagens de telefone celular para os usuários.

Muitos softwares têm a capacidade de configurar os parâmetros dos medidores, como horário, seleção de grandezas elétricas, desabilitação do teclado, passo de integração, relação de “TP” e “TC”, etc. A Figura 2.4 mostra exemplos de janelas (de softwares comerciais) para a configuração do hardware.

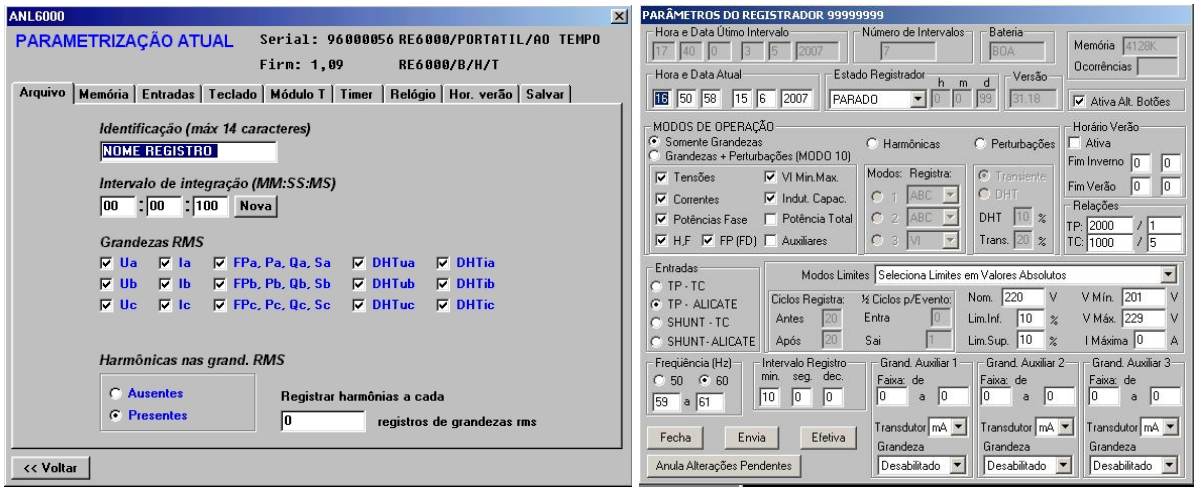


Figura 2.4 – Exemplo de configuração do hardware via software.

Após todo sistema corretamente instalado e configurado, a função do software é coletar as informação elétricas adquiridas pelos medidores, e processar os dados de forma oferecer informações úteis para os usuários, além de armazená-las para posteriores consultas e análises.

As informações disponibilizadas aos usuários são diversas, dependendo da filosofia e complexidade do software. Entretanto, comumente são encontradas funções para a visualização “online” das formas de onda dos sinais elétricos, como mostrado na Figura 2.5.

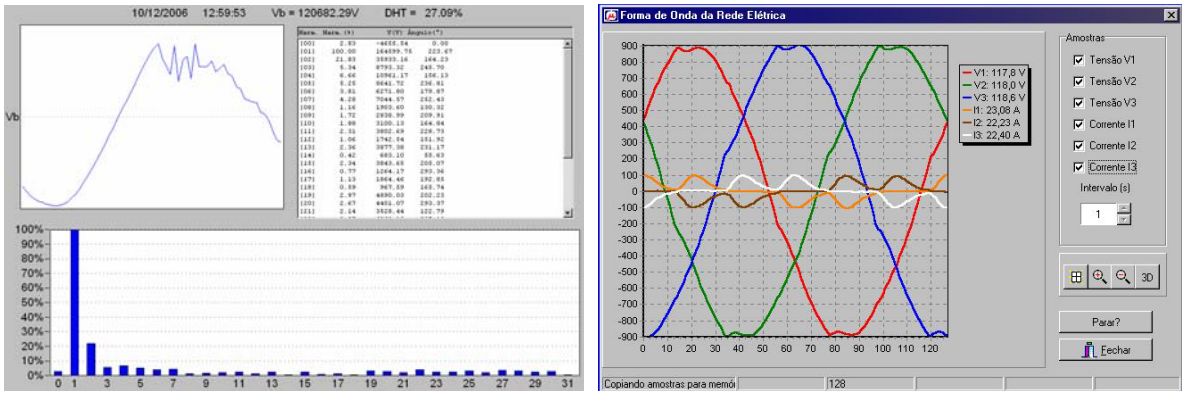


Figura 2.5 – Formas de onda instantâneas.

Em tempo real, também podem ser mostradas as grandezas elétricas, calculadas pelos medidores, ou pelo software. Abaixo são citadas algumas grandezas.

- Potência ativa das fases A, B, C e total;
- Potência reativa das fases A, B, C e total;
- Potência reativa indutiva das fases A, B, C e total;
- Potência reativa capacitiva das fases A, B, C e total;
- Potência aparente das fases A, B, C e total;
- Fator de potência das fases A, B, C e total;
- Corrente das fases A, B, C e de neutro;
- Tensão das fases A, B e C;

- Tensão entre Fases;
- Valores de tensão em “pu”;
- Distorção harmônica total de tensão das fases A, B e C;
- Distorção harmônica total de corrente das fases A, B e C;
- Freqüência;

Outras informações que o software pode disponibilizar são relatórios de distúrbios elétricos ocorridos, análises estatísticas, predição e identificação de eventos, além de informações elétricas em geral. Abaixo estão descritas algumas informações típicas [20],[71].

- Valor eficaz máximo e mínimo de tensão e de corrente, para cada intervalo de registro (ciclo);
- Tempo em que a tensão ou a corrente estiveram fora dos limites programados, para cada intervalo;
- Banco de capacitores necessário inserir ou retirar;
- Postos diários (Ponta, Fora-ponta e Reservado);
- Grau de desequilíbrio de fases;
- Tensões: mínimas, máximas e média (programável de 1 a 10);
- Correntes: mínimas, máximas e média (programável de 1 a 10);
- Demandas: ponta, fora-ponta e reservado (programável de 1 a 20);
- Consumo/Fator de Potência: por fase e total;
- DHTs;
- Relatórios de Falta de Energia e Alterações;
- Demandas e consumos por postos horários, através da programação dos horários de "Fora-ponta", "Ponta" e "Reservado".

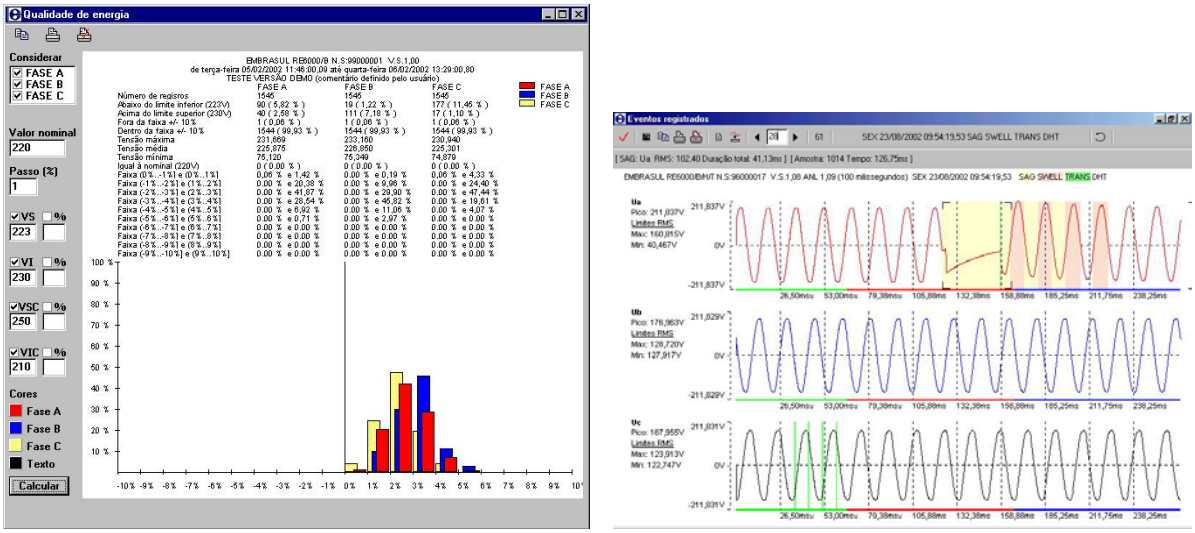


Figura 2.6 – Exemplos de relatórios e informações elétricas.

De acordo com a referência [20], a função de identificação de eventos pode ser dividida em três categorias: identificação do local, identificação da causa e sugestão para mitigar o evento.

A identificação do local do evento consiste na busca pelo local de origem do mesmo, o que é bastante importante para a resolução rápida de problemas, através da remoção do dispositivo gerador do distúrbio, e da procura pelo método de compensação mais adequado. Um número maior de medidores instalados no sistema oferece uma exatidão maior para a localização do evento.

A referência [20] também indica que a tendência estimada para os parâmetros de qualidade da energia é feita para um período de tempo relativamente longo, devido ao cálculo ser feito pela média, de meia em meia hora, de valores tomados a cada meio segundo. Estas informações de tendência são valores RMS de tensão e corrente, potências, energias, freqüência, fator de potência, distorção harmônica ímpar, par e total, fator de distorção e fator de desbalanceamento.

Quanto aos algoritmos utilizados pelos softwares, os mesmos podem ser de diferentes tipos. A referência [20], por exemplo, cita os algoritmos “Transformada Discreta de Fourier (DFT)”, “Transformada Rápida de Fourier”(FFT), “Método de Prony” e “Transformada Discreta Esperta de Fourier”(SDFT), e utiliza este último método por ser mais preciso, quando a freqüência desvia do valor nominal, e também pelo cálculo ser mais rápido e imune

à harmônicas. “Wavelet” e outras formas de quantificação vêm também sendo estudadas nos últimos tempos para testar a performance sob o ponto de vista de quantidade de energia dos sistemas elétricos.

Além do armazenamento em banco de dados, geralmente o software também possibilita o armazenamento das informações em arquivos, que em alguns casos está em formato de texto, em outros é um formato próprio do fabricante do software, mas a tendência é que todos os sistemas de monitoramento de energia utilizem um padrão para o gerenciamento de arquivos de informações elétricas, que é o formato “COMTRADE” [29],[68].

Ainda, os softwares podem permitir a exportação dos dados para outros aplicativos, como o “Microsoft Excel”, “Lotus”, “Microsoft Access” e outros.

2.1.3 Banco de Dados

A função principal de um banco de dados em um sistema de monitoramento de energia é o armazenamento das informações que o software coletou dos medidores da rede. Mas o fato de os dados terem sido guardados no banco não é importante somente para futuras consultas e análises. Esse armazenamento permite que os medidores possuam menos memória de massa para guardar as informações, possibilitando uma redução do custo do medidor [61].

Ainda, devido à organização que as tabelas do banco de dados oferecem, e também à potencialidade da linguagem SQL, torna-se fácil realizar consultas das informações pelos parâmetros desejados, como localidade, data, hora ou grandezas elétricas, permitindo que se filtre os dados de interesse.

A partir desta seleção dos dados, é possível ao sistema de gerenciamento da energia gerar relatórios de demanda, consumo, eventos ocorridos, entre outros, que se processados juntamente com outras informações de custos e produção, formam uma poderosa ferramenta para a gestão do negócio.

A utilização do banco de dados também oferece a possibilidade de se inserir processamentos com inteligência artificial para a busca de distúrbios, auxiliando na automação para a análise da qualidade da energia. Isto também é muito útil para a

comparação das condições elétricas existentes com as condições históricas, de forma a ser possível prever estatisticamente quando pode ocorrer alguma falha. Esse conhecimento é bastante importante para se evitar que algum dano maior aconteça, fazendo-se antes disso a manutenção preditiva e preventiva [38].

Além das vantagens acima descritas, a utilização de um banco de dados torna possível o fácil acesso às informações elétricas através da internet, o que possibilita às indústrias uma integração das informações operacionais e econômicas de forma fácil e a um baixo custo, possibilitando que um especialista realize uma análise de qualquer lugar do mundo, e não somente de dentro da planta industrial [89]. Isto para empresas globalmente distribuídas é uma grande vantagem, pois não necessita uma pessoa altamente qualificada em cada unidade, e também podem analisar os dados da empresa como um todo.

2.1.4 Interface de Consulta Remota ao Banco

Devido ao fenomenal progresso da internet nos últimos anos, a comunicação de dados por meio dela vem se tornando cada vez mais comum e conveniente. Com o amadurecimento desta grande rede de computadores, três classes principais de aplicação se desenvolveram. Primeiro, somente informações estáticas, como arquivos de texto e figuras estavam disponíveis na WEB. Esta tecnologia foi desenvolvida inicialmente para permitir o compartilhamento de documentos complexos, embora estáticos, através da internet. A linguagem HTML combinada com o protocolo HTTP ofereceram um mecanismo para a visualização remota para esse tipo de documentos.

Em seguida, devido à falta de interatividade dos formulários HTML, foi desenvolvida uma tecnologia chamada CGI (do inglês, “Commom Gateway Interface”), que permitiu a geração de páginas dinâmicas, facultando a um navegador passar parâmetros para um programa alojado em um servidor WEB (pequenos programas conhecidos como scripts CGI interpretam esses parâmetros e geram a página depois de os processar). Ou seja, o usuário informa os parâmetros desejados e os envia ao servidor, que realiza o processamento e apresenta o resultado.

E por final, nos ultimo anos, uma outra tecnologia tem surgido na internet: as páginas ASP (do inglês, “Active Server Pages”) e Java. Assim como a CGI, ASP também é executada no servidor, e envia os resultados para o cliente em forma de documento HTML. Mas a ASP oferece funções mais poderosas e com interface de mais fácil implementação.

Hoje em dia é comum que pessoas acessem a dados administrativos através de consultas pelo navegador da internet, inclusive informações sobre grandezas elétricas, disturbios ocorridos e consumo. E essa tentência torna-se cada vez mais marcante, devido às seguintes vantagens [42]:

- Custo e disponibilidade: os usuários não precisam comprar nenhum dispositivo de hardware ou software para poder utilizar o acesso à rede mundial de computadores. Simplesmente com um computador com placa de rede ou placa de modem e uma linha telefônica, linha ISDN ou modem a cabo pode-se ter obter as informações desejadas ou atualizar dados remotamente;

- Suporte a diversas plataformas: em um meio padronizado do navegador da internet com os protocolos HTML e TCP/IP, os usuário podem visualizar as mesmas informações em diferentes plataformas. Assim, a interface gráfica ao usuário é desenvolvida uma vez, e pode ser apresentada em diversas versões do Microsoft Windows, no UNIX, no LINUX, etc.;

- Compatibilidade com sistemas de arquitetura aberta: As aplicações da internet seguem sistemas de padrão aberto, ou seja, SQL, HTML, HTTP, FTP, TCP/IP e PPP. Intercâmbio de dados e expansão do sistema são feitos com mínimo esforço. A interface das aplicações e do banco de dados são elaboradas consistentemente de acordo com as diretrizes industriais, ou seja, através de uma tecnologia padrão de programação para o acesso a banco de dados (ODBC);

- Interface homem-máquina amigável: os usuários não precisam aprender comandos ou operações difíceis para utilizar uma nova aplicação. Assim, os esforços com treinamento podem ser minimizados;

- Minimização dos esforços para a instalação e manutenção: novos aplicativos podem ser carregados diretamente da internet aos computadores clientes.

Atualizações dos softwares podem ser feitas automaticamente quando novas versões estão disponíveis, de forma rápida e fácil;

- Acesso a banco de dados relacional: as aplicações da internet podem trabalhar com banco de dados, de forma a ler dinamicamente os dados nele inseridos instantaneamente, a também atualizá-los.

Segundo [42], um sistema de monitoramento da qualidade da energia com acesso aos dados via WEB deve conter as seguintes propriedades:

- Monitoramento: o sistema deve monitorar as informações todo o tempo;
- Compartilhamento de dados: o sistema deve ser capaz de compartilhar dados com outro sistema relacional de dados, através do ODBC, e também com outros arquivos simples unidimensionais, por meio do padrão SQL;
- Utilização de funções via internet: um sistema avançado de monitoramento da qualidade da energia deve oferecer ferramentas para o supervisionamento do sistema pela internet;
- Segurança: por um fator de segurança, os usuários devem ser divididos em diferentes grupos, com as respectivas permissões.

Também, segundo [42], o sistema acessível via internet deve conter os seguintes elementos:

- Clientes: são os computadores que acessam aos dados do sistema através de um navegador da internet;
- Servidor WEB: é o computador que armazena a programação que serve de interface para o usuário cliente. Ele que recebe os pedidos, faz o processamento (interface com o banco de dados) e envia a resposta, em forma de página HTML.
- Banco de dados: este é o dispositivo utilizado para armazenar as informações do sistema de monitoramento da qualidade da energia. Quando solicitada alguma consulta, ele realiza o processamento, filtrando os dados de interesse, e fornece a resposta.

A Figura 2.7 mostra o esquema de conexões sugerido.

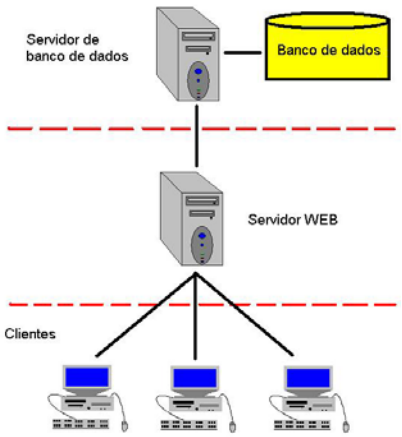


Figura 2.7 – Esquema de ligação entre clientes, servidor e banco de dados.

2.2 Normas

As normas internacionais são normas técnicas estabelecidas por organismos internacionais de normalização para aplicação em âmbito mundial, propondo regulamentos técnicos e procedimentos. Estas normas internacionais não constituem barreiras técnicas, bem pelo contrário, elas são elaboradas para que sejam usadas global e nacionalmente como referência.

Por esta razão assiste-se a uma forte tendência de os organismos nacionais de normalização adotarem as normas internacionais integralmente como normas nacionais [21].

As normas mais importantes para o assunto de qualidade de energia estão citadas abaixo [27].

- EN50160: é uma nova norma que cobre flicker, interharmônicas, desvios/variações de tensão, e muito mais;
- IEC 61000-4-15: é uma norma de medição de flicker que inclui especificações para medidores;
- IEC 61000-4-7: descreve uma técnica de medição padrão para harmônicas;

- IEEE 519 (1992): é uma prática recomendada pela IEEE, utilizada principalmente por concessionárias de energia nos EUA. Descreve níveis aceitáveis de harmônicas para o ponto de entrega de energia pela concessionária;

- IEEE 1159 (1995): é uma prática recomendada pela IEEE para monitoração e interpretação apropriada dos fenômenos que causam problemas de qualidade de energia;

- ANEEL 505 (2001): estabelece, de forma atualizada e consolidada, as disposições relativas à conformidade dos níveis de tensão de energia elétrica em regime permanente.

- CBEMA (Computer and Business Equipment Manufacturers Association): A curva CBEMA define os níveis de suportabilidade de equipamentos, em função da magnitude da tensão e da duração do distúrbio. Distúrbios que caiam fora da curva podem causar danos aos equipamentos;

- ITI: Information Technology Industry Council. Grupo trabalha para defender os interesses da indústria de informática.

3 DESENVOLVIMENTO DO SOFTWARE

Durante as três primeiras décadas da era do computador, o principal desafio era desenvolver um hardware que reduzisse o custo de processamento e armazenagem de dados. Ao longo da década de 1980, avanços na microeletrônica resultaram em maior poder de computação a um custo cada vez mais baixo. Atualmente, o problema é diferente. O software superou o hardware como chave para o sucesso do sistema, tanto em termos de custos, como estratégia competitiva aos sistemas concorrentes, por meio das informações e interface oferecidos [69].

Assim, o software gerenciador tem uma função muito importante para o funcionamento do sistema de monitoramento da energia, pois ele é o dispositivo mestre na rede RS485, que deve fazer os pedidos dos vetores representativos das tensões e/ou correntes de um determinado medidor, para então apresentar as formas de onda, os cálculos dos parâmetros de performance e realizar o armazenamento das informações no banco de dados. Ou seja, ele deve gerenciar o fluxo de dados, oferecendo vazão ao processo. A Figura 3.1 mostra a estrutura do sistema.

Como o sistema pode ser composto por uma quantidade relativamente grande de medidores (até 32, segunda a referência [77]), e o software deve gerenciar todas as características de cada um, incluindo a temporização para a consulta e as informações adquiridas, o software deve ter um fluxo de dados muito bem organizado e estruturado, de forma a este não trancar ou interpretar os dados de forma errônea.

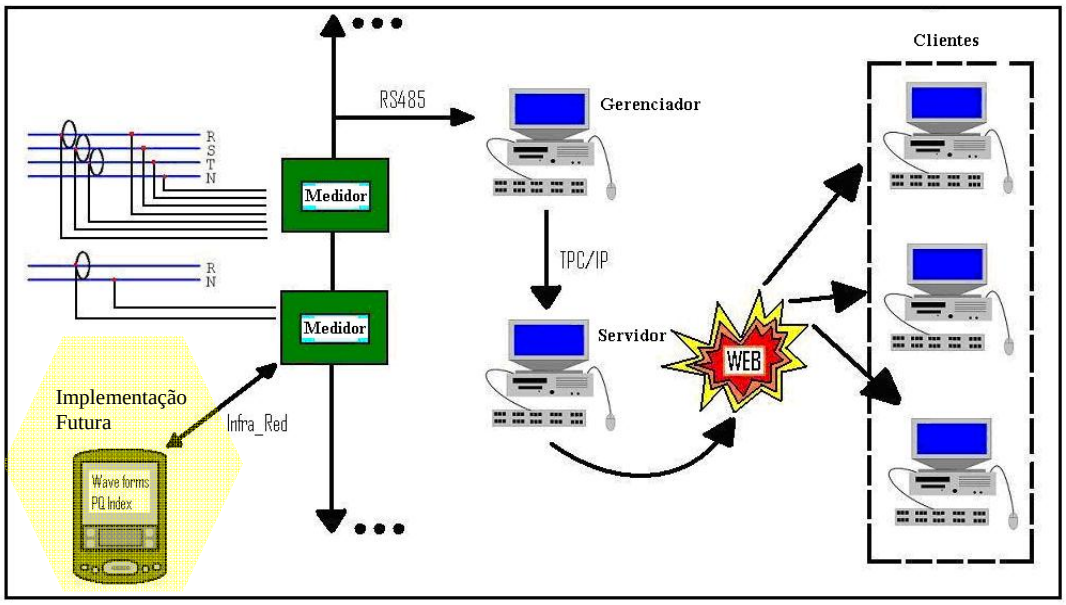


Figura 3.1 – Topologia do sistema proposto.

Também, o software foi construído pensando-se em possíveis atualizações ou modificações futuras, de forma a que este não estivesse limitado aos recursos iniciais, e sim oferecesse flexibilidade e possibilitasse reutilização de software. Por exemplo, outras formas de comunicação com os medidores poderiam ser utilizadas, ao invés do Modbus/RTU, assim como diferentes formas de armazenamento dos dados, ou formas de apresentação dos mesmos, entre muitos outros pontos passíveis de mudanças.

Por isso, a implementação do software foi feita baseando-se nas técnicas de orientação a objetos, criando-se uma estruturação de classes e subclasses.

3.1 Programação Estruturada e Orientação a Objetos

À medida que o número de sistemas baseados em computador começou a crescer, o mesmo aconteceu com bibliotecas de software. Os projetos de software passaram a ter um número bastante grande de linhas de código. Softwares vindo de outros países muitas vezes precisavam de ajustes ou manutenção, onde podiam ser inseridas milhares de novas

instruções. A complexidade foi tomando uma intensidade tão grande, que o problema se tornava uma “caixa preta”, onde ficava muito difícil de obter o referencial. A adaptação do software à correção de uma falha, ou à utilização de um novo hardware não era nada trivial. O esforço despendido na manutenção do software começou a absorver recursos em índices alarmantes [69].

Em meio a esta necessidade, técnicas para o planejamento e construção de um software começaram a ser elaboradas. Primeiramente, três estruturas simples foram propostas: seqüência, decisão (condição) e iteração, representando a programação estruturada, onde a organização se baseava na utilização de sub-rotinas e funções.

Em seguida, a técnica de programação orientada a objetos apareceu, não para substituir a programação estruturada, mas sim como uma atualização desta, onde houve a formalização de práticas como o uso de variáveis locais e a visibilidade dos dados. Também alterou o foco principal de funções e procedimentos, para os objetos e seus relacionamentos.

A programação orientada a objetos tem como alicerce os conceitos de encapsulamento, classe, herança e polimorfismo. A utilização deste método permite a criação de bibliotecas que facilitam o compartilhamento e reutilização de código, o que reduz o tempo de desenvolvimento e, principalmente, a simplificação da manutenção das aplicações [45].

A construção de software orientada à objetos ajuda ao projetista estruturar o aplicativo sustentado no conceito da “modularidade”, que é um método bastante interessante de se aplicar por ajudar na decomposição do problema. A modularidade é efetivada pela ocultação de informações, ou seja, os módulos comunicam entre si somente as informações que são necessárias para a obtenção da função do software. O uso da ocultação de informações como critério de projeto para sistemas modulares oferece maiores benefícios quando são exigidas modificações durante as fases de teste e manutenção [69].

O conceito de independência funcional é o produto direto da modularidade e dos conceitos de abstração e ocultação de informações, e é conseguida desenvolvendo-se módulos com funções de “um único propósito”, evitando iteração excessiva com outros módulos. Isto facilita o desenvolvimento do software, porque a função pode ser dividida em compartimentos e as interfaces são simplificadas, além de que a propagação de erros é reduzida e a reutilização de módulos se torna possível. Em fim, a independência funcional é bastante importante para um bom projeto, e o projeto é a chave para a qualidade do software [69].

A “independência” é medida usando-se dois critérios qualitativos: coesão e acoplamento. A coesão é uma extensão natural do conceito de ocultação de informações, enquanto o acoplamento é uma medida da interconexão entre os módulos de uma estrutura de software. No projeto de software, é vantajoso buscar uma coesão moderada ou alta, e um baixo acoplamento [69].

Todos esses conceitos e estratégias para a elaboração de software têm o objetivo de agregar ao aplicativo as seguintes caraterísticas:

- Manutenibilidade: o software deve evoluir de modo a atender alterações das necessidades do cliente;
- Confiável: o software deve apresentar estabilidade;
- Eficiente: o software não deve desperdiçar recursos computacionais;
- Aceitável: o software deve ser facilmente entendido pelo usuário, através de uma interface simples e compatível com o propósito.

Ainda relacionado com a estruturação do aplicativo e orientação à objetos está a “reutilização de software”, que traz consigo as seguintes vantagens:

- Diminuição do custo de desenvolvimento;
- Aumento da produtividade;
- Atividade de manutenção facilitada;
- Aumento na qualidade;
- Redução dos riscos.

Por todas estas vantagens acima relatadas, o software gerenciador e o banco de dados do sistema de monitoramento de energia foram estruturados tendo como base as técnicas de engenharia de software.

3.2 Estruturação do Software

A primeira tarefa para o começo da estruturação do software foi a identificação de quais funções o aplicativo deveria desempenhar (diretrizes), quais eram as possibilidades e quais os riscos.

Assim, foi elaborada uma descrição do processamento para o software gerenciador do sistema, mostrada a seguir:

> O software gerenciador permite ao usuário fazer a configuração do banco de dados (conexão) e do protocolo de comunicação, onde a primeira é indispensável para a aquisição das informações dos medidores, e a segunda é necessária somente se os dados forem ser escritos no banco. Também é possível, e necessário, o cadastramento dos medidores no software, através dos parâmetros endereço, fases (TA, TB, TC, CA, CB, CC), número de pontos por ciclo, número de ciclos, temporização (timer) e código do medidor no banco de dados (se for o caso). Também é possível escolher se o software deve apresentar gráficos, cálculos e/ou armazenar as informações no banco de dados. Essa lista de medidores cadastrados pode ser salva em arquivo de texto, assim como pode ser posteriormente aberta.
>
> Após todos medidores estarem cadastrados, o usuário pode escolher de quais quer começar a adquirir os dados, e interromper, recomeçar, cadastrar e descadastrar a qualquer momento.
>
> Os vetores são adquiridos dos medidores através de pedidos, utilizando-se o protocolo ModBus/RTU. As informações pedidas a um medidor são sempre iguais, de acordo com as fases, número de ciclos e pontos por ciclo cadastrados.
>
> Tendo as informações em memória, o software deve realizar as operações indicadas (gráficos, cálculos e armazenamento em banco de dados), de acordo com o cadastramento do medidor.

Essa descrição ajuda a identificar os agentes e os processamentos do sistema, auxiliando no desenvolvimento do primeiro diagrama utilizado no projeto do software , o diagrama de fluxo de dados (DFD), que auxilia no desenvolvimento dos domínios da informação e funcional, simultaneamente. A Figura 3.2 mostra o DFD obtido.

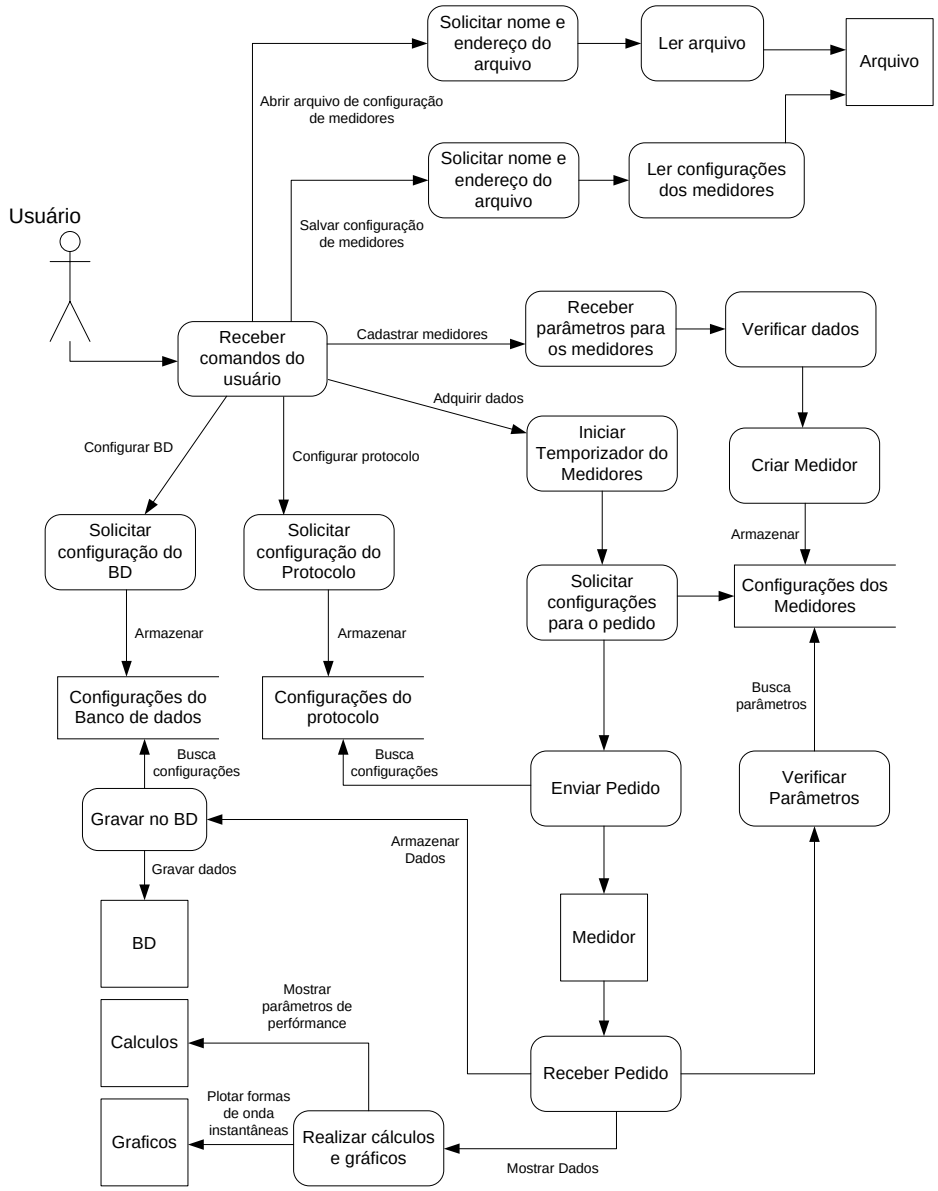


Figura 3.2 – Diagrama de fluxo de dados.

Este diagrama é bastante interessante de se observar no início do planejamento do software, pois ele é uma técnica gráfica que descreve o fluxo de informações e as transformações que são aplicadas à medida que os dados se movimentam da entrada para a saída.

Pode-se observar que bloco “Receber comandos do usuário” representa uma parte central do software, o qual direciona o processamento para as sub-funções (ramificações) do sistema, como a abertura (ou gravação) de um arquivo de configurações, a configuração do

protocolo de comunicação ou do banco de dados, o cadastramento/retirada dos medidores ou o controle das aquisições de dados.

Também se pode verificar que em níveis mais abaixo o processamento é automático, não mais dependendo da ação do usuário. Por exemplo, quando o temporizador de um medidor cadastrado termina a contagem, é enviado um pedido pela porta serial com as configurações armazenadas na instância desse medidor. E quando chega a resposta à esse pedido, o software verifica os de novo os parâmetros para decidir se deve gerar os gráficos, realizar os cálculos ou gravar no banco de dados.

Outra ferramenta utilizada, cujas finalidades são a delimitação do contexto do sistema, documentação e entendimento dos requisitos, servindo também como entrada para a etapa de análise, é a descrição e o diagrama dos casos de uso.

A descrição dos casos de uso está no Apêndice A, e o diagrama de casos de uso está mostrado na Figura 3.3.

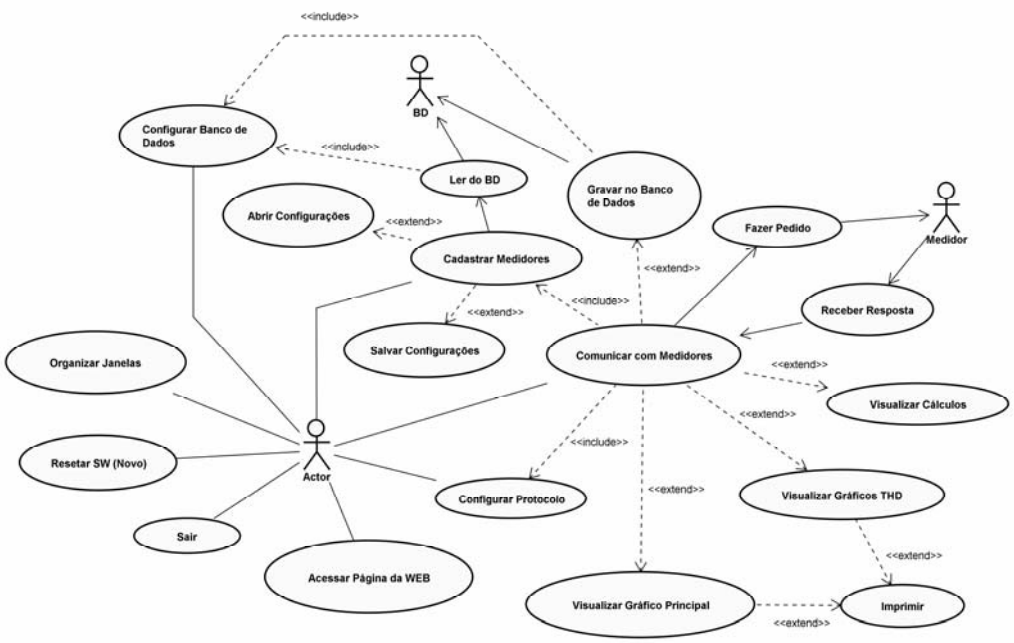


Figura 3.3 – Diagrama de casos de uso.

No diagrama da Figura 3.3 busca-se caracterizar de forma gráfica a visualização de como o sistema a ser desenvolvido vai interagir com o ambiente (usuários e outros sistemas), o que serve de base do processo de desenvolvimento do sistema [4].

Outro diagrama interessante de se utilizar no desenvolvimento de um software é o diagrama de classes, que é um grafo que descreve as estruturas de classe do aplicativo, seus atributos, operações e relacionamentos presentes no sistema. Este diagrama é utilizado para a modelagem estática, dando suporte às necessidades funcionais (serviços que o sistema deve providenciar aos usuários finais), e auxiliar nas seguintes estruturações:

- Vocabulário do sistema: especificação das abstrações contidas dentro do domínio do sistema, identificando as suas responsabilidades;

- Colaboração: envolve o trabalho conjunto entre os objetos do sistema, visando um comportamento cooperativo.

O diagrama de classes desenvolvido está mostrado na Figura 3.4.

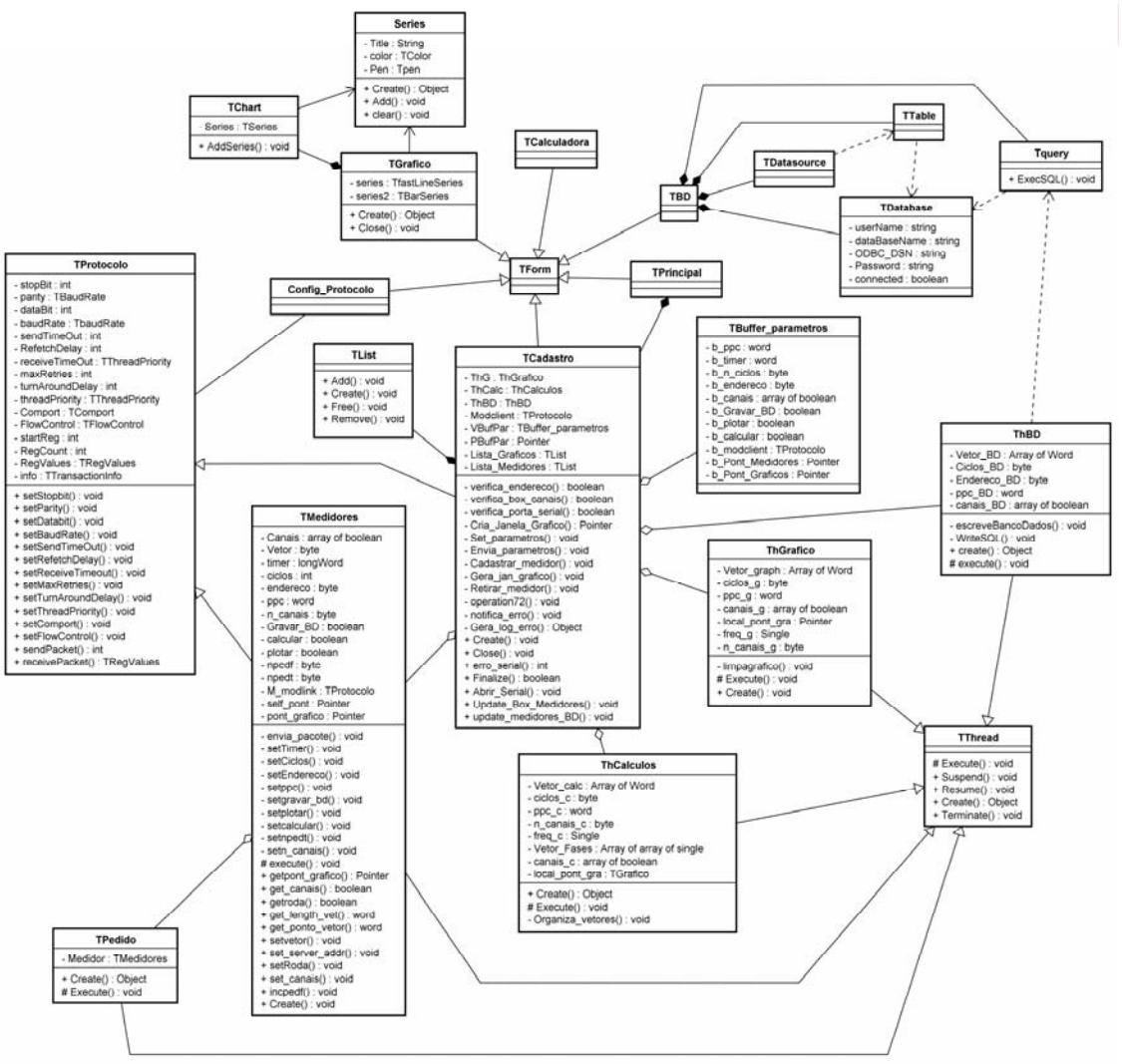


Figura 3.4 – Diagrama de classes.

O último diagrama utilizado para a estruturação do software foi o diagrama de seqüência, que tem como objetivo apresentar as interações entre objetos, enfatizando a seqüência temporal em que as mensagens são trocadas.

Esta é outra forma de se visualizar graficamente o fluxo de informações e dados, através das mensagens dos objetos do software, e está mostrado no Apêndice B.

3.3 Interface Visual

Todo usuário que vá utilizar algum software não deseja se deparar com uma interface muito difícil de entender, confusa ou frustrante. Por isso, a interface visual do software foi planejada visando oferecer ao usuário facilidade de manipulação e visualização das informações.

Na tela inicial, o usuário tem acesso a botões e à barra de menus, que levam à configuração do banco de dados, configuração do protocolo de comunicação, ao cadastramento dos medidores e à página do sistema na internet. Também é possível abrir um cadastro de medidores.

Ainda na tela inicial são encontradas mais duas informações: a data e a hora da última entrada e saída do software (para o usuário monitorar a utilização do aplicativo), e o status do protocolo ModBus (ativado ou não, taxa de transmissão, pedidos pendentes...).

A Figura 3.5 mostra a tela inicial do software.

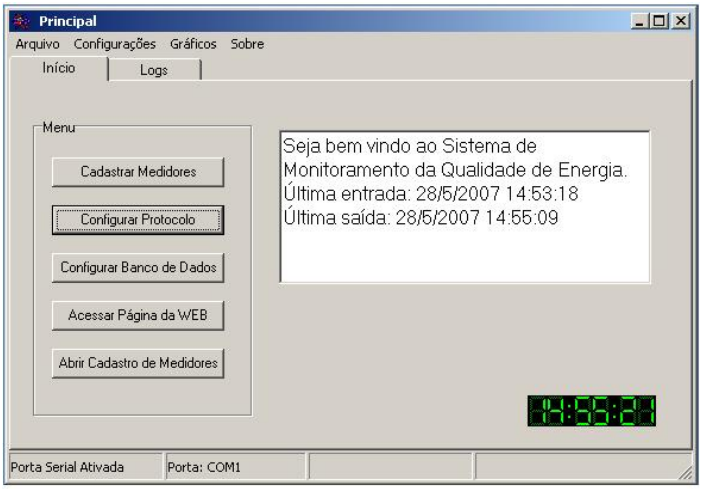


Figura 3.5 – Janela inicial do software.

A Figura 3.6 mostra a janela para a conexão com o banco de dados, onde pode-se observar que o usuário pode escolher o “driver” do windows para fazer o “link” com o banco, assim como um nome de usuário, a senha e a empresa. Caso esteja tudo correto, o software se conecta ao banco de dados, e já procura todos medidores cadastrados para esta empresa, assim como a posição da última medição realizada.

Figura 3.6 – Janela de conexão com o banco de dados

A janela de configuração do protocolo de comunicação (ModBus) possibilita ao usuário configurar os parâmetros referentes, como a taxa de transmissão, número de bits de dados, paridade, controle de fluxo de dados (utilizado “RTS toggle” para a rede RS485), número de tentativas, tempo de espera entre um pacote e outro, etc. Toda a comunicação realizada entre o software e os medidores estará baseada nestes parâmetros de configuração.

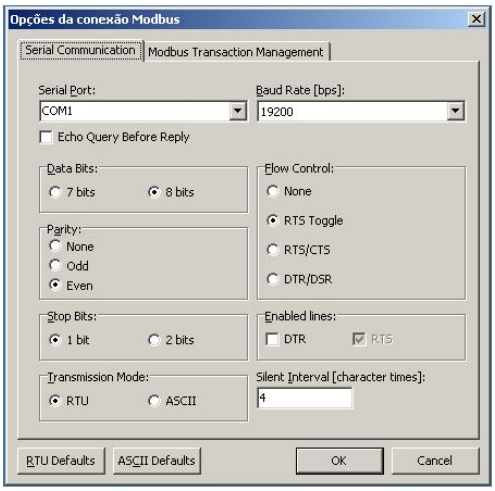


Figura 3.7 – Janela de configuração do protocolo de comunicação.

Outra janela de muita importância para o software é a do cadastramento dos medidores (Figura 3.8), onde se pode cadastrar/descadastrá-los da lista, iniciar a consulta, interromper ou parar a mesma, verificar quais os medidores cadastrados, e quais os seus parâmetros, entre outras funções.

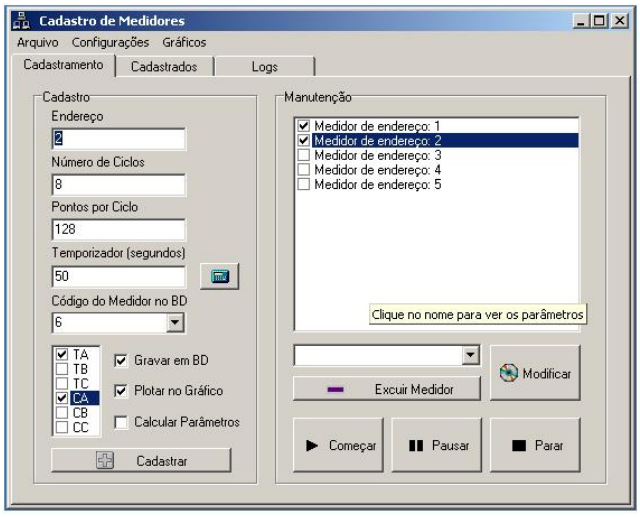


Figura 3.8 – Janela de cadastramento dos medidores.

Pode-se observar que a forma de entrada de dados para os parâmetros “endereço”, “pontos por ciclo”, “número de ciclos” e “timer” é feito por meio de “caixas de texto” (EditBox), onde o usuário poderia colocar manualmente um valor errado, ou fora dos limites. Por isso, antes de realizar o cadastramento do medidor, o aplicativo faz uma varredura nestes

dados de entrada em busca de erros de escrita ou limite, e se encontra algum, um aviso é emitido.

Já o código do medidor no banco de dados aparece em uma “combo box” (caixa onde as opções são listadas), a partir das informações coletadas pelo software na conexão do banco de dados. Esse item só é obrigatório estar configurado se as informações do medidor deverão ser armazenadas no banco.

Há também, além da escrita no banco, mais duas opções referentes à visualização dos dados, que são a visualização do gráfico e a dos cálculos dos parâmetros de performance. Quando alguma destas opções estiver selecionada, o software gera uma janela exclusiva do medidor recém cadastrado. Ou seja, não importa se existam 5 ou 10 medidores cadastrados, é possível ver as formas de onda de tensão e corrente trifásica de cada medidor ao mesmo tempo, através das 5 ou 10 janelas de gráficos. O mesmo ocorre com os cálculos, como é mostrado na Figura 3.9.

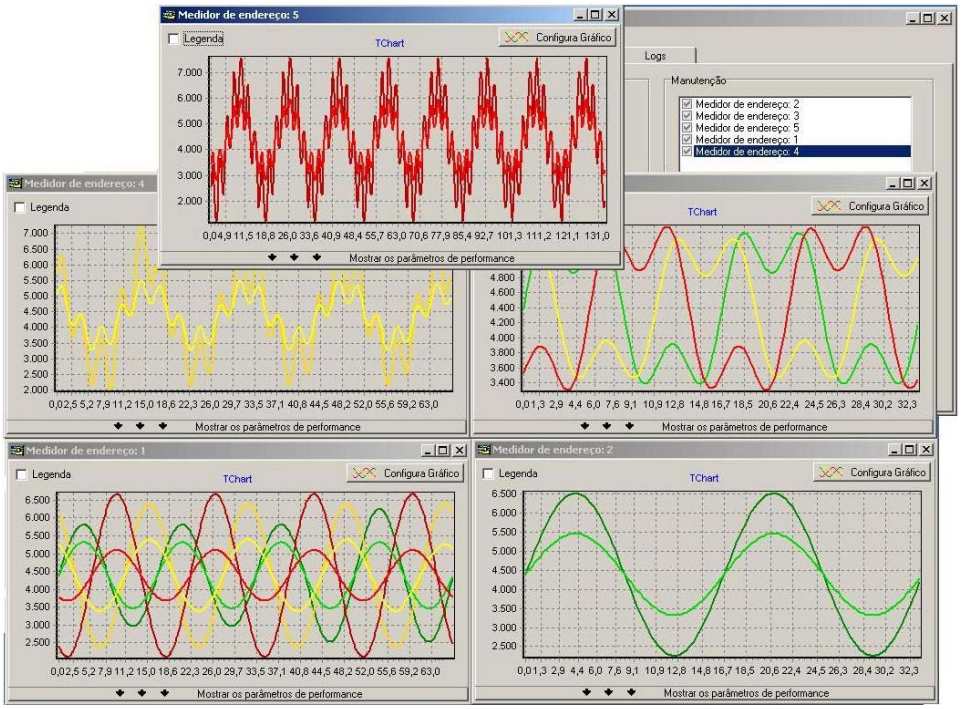


Figura 3.9 – Visualização gráfica das formas de onda dos medidores.

É possível verificar na Figura 3.9 que o usuário pode escolher entre ver ou não os cálculos de parâmetros de performance. Isto foi elaborado para facilitar a visualização das

informações dos medidores quando há muitos cadastrados, e assim, muitas janelas para serem vistas.

Por fim, se o usuário tenta sair do software, e existem medidores cadastrados (adquirindo dados ou não), o aplicativo avisa e pede confirmação para sair do software, mesmo perdendo a lista de medidores cadastrados e as informações instantâneas.

3.4 Implementação do Código Fonte

A implementação do programa foi feita baseada em métodos modernos de engenharia de software, utilizando a programação estruturada e a orientação à objetos, de forma que a estrutura do aplicativo ficasse modularizada.

Como se pode observar na Figura 3.4, diversas classes compõem o aplicativo. Entretanto, as mais importantes de serem explicadas são as desenvolvidas pelo projetista. São elas: TPrincipal, TCadastro, TBD, TGrafico, TRelacao_TPTC, TCalculadora, TBuffer_parametros, TPedidos, TMedidore, ThGrafico, ThCalculos, ThBancoDados. As 6 primeiras classes são derivadas de TForm, ou seja, são janelas visíveis. As 4 últimas, são derivadas da classe “TThread”, que será explicada a seguir. Ainda há classes muito importantes, referentes ao protocolo de comunicação (ModBus), porém não foram desenvolvidas neste trabalho. Elas fazem parte de uma biblioteca chamada “ModLink”, que é um componente comercial, e está disponível em [7] .

Embora todas classes sejam relevantes, somente as mais importantes serão explanadas.

3.4.1 TCadastro

A TCadastro é uma classe derivada de TForm, e por isso é uma janela visível. Esta é uma janela de cadastramento e manutenção dos medidores do sistema, e por isso deve oferecer todos os recursos para tal.

Como se pode observar na Figura 3.8, os parâmetros para o cadastramento do medidor são o número na rede ModBus (endereço), o número de ciclos e pontos por ciclo, as fases a serem adquiridas, a periodicidade (timer) e as funções “gravar em banco de dados”, “plotar em gráfico” e “calcular parâmetros de performance”.

E levando-se em conta que podem ser muitos medidores cadastrados, a classe TCadastro deve gerenciar esta lista de medidores. Para isto, ela usa uma outra classe, chamada TList, que vem com o Delphi, e que permite a criação de listas de ponteiros de objetos (através de um vetor dinâmico), assim como a fácil manipulação dos itens componentes (inserção, remoção...).

Para a armazenagem dos parâmetros na classe TMedidor, que será explicada adiante, é utilizada outra classe como uma etapa intermediária, a TBuffer_parametros. Isto é feito para haver um desacoplamento das informações lidas ou escritas pela classe TCadastro na classe TMedidor. Abaixo está mostrado o código fonte que armazena os parâmetros inseridos pelo usuário na classe TBuffer_parametros, assim como a sua declaração e inicialização.

```
type
  TBuffer_parametros = class
    b_ppc, b_timer, b_cod_BD : word;
    b_n_ciclos, b_endereco : byte;
    b_canais : array[0..5] of boolean;
    b_Gravar_BD, b_plotar, b_calcular : boolean;
    b_modclient : TModbusClient;
    b_Pont_Medidores : ^TMedidor;
    b_Pont_Graficos : ^TGrafico;
  end;
//=============================================//
procedure TCadastro.FormCreate(Sender: TObject);
begin
 Lista_Medidores := Tlist.Create;
 Lista_Graficos := Tlist.Create;
 VBufPar := TBuffer_parametros.Create;
 PBufPar := @VBufPar;
 Combo_medidores.Clear;
end;
```

```
function TCadastro.set_parametros(janela : boolean) : boolean;
var
 b : byte;
begin
 if CB_bd.Checked and (Combo_codBD.ItemIndex = -1) then
 begin
  showMessage('Código do medidor no BD não selecionado !');
  if not(BD.Database1.Connected) then
  begin
   if application.MessageBox('Banco de dados não está conectado, conectar ?',
                             'Aviso',MB_YESNO) = IDYES then
    BD.show;
  end;
  result := false;
  exit;
 end;
//=============================================//
 PBufPar^.b_endereco := strtoint(box_endereco.text);
 PBufPar^.b_n_ciclos := strtoint(box_n_ciclos.text);
 PBufPar^.b_ppc := strtoint(box_ppc.text);
 PBufPar^.b_timer := strtoint(box_timer.text);
 PBufPar^.b_Gravar_BD := CB_bd.Checked;
 PBufPar^.b_plotar := CB_plotar.Checked;
 PBufPar^.b_calcular := CB_calculos.Checked;
 if (Combo_codBD.ItemIndex = -1) or
      (Combo_CodBD.Items[Combo_CodBD.ItemIndex] = '') then
  PBufPar^.b_cod_BD := 0
 else
  PBufPar^.b_cod_BD := strtoint(Combo_CodBD.Items[Combo_CodBD.ItemIndex]);
 PBufPar^.b_Pont_Graficos := nil;
 for b := 0 to 5 do
  PBufPar^.b_canais[b] := box_canais.Checked[b];
 gera_jan_grafico(janela);
 result := true;
end;
```

Pode-se verificar que tanto a lista auxiliar de parâmetros (VBufPar) como a lista de medidores (Lista_Medidores) utilizam ponteiros para o seu funcionamento. Por isso aparece, por exemplo, uma expressão como “PBufPar^.b_n_ciclos := strtoint(box_n_ciclos.text);”, significa o ponteiro “PBufPar” apontando para o parâmetro “n_ciclos” da lista “VBufPar”.

A seguir está mostrado o código para a armazenagem dos parâmetros na classe TMedidor, o que é feito através do método construtor “Create”, que aloca memória para a nova instância da classe.

```
procedure TCadastro.cadastrar_medidor;
begin
 Modclient := Tmodbusclient.Create(modbusclient1);
 Modclient.OnHoldingRegistersRead := Modbusclient1.OnHoldingRegistersRead;
 Modclient.Connection := Modbusconnection1;
 PBufPar^.b_modclient := Modclient;
 New(PBufPar^.b_Pont_Medidores);
 PBufPar^.b_Pont_Medidores^ := TMedidor.create(PBufPar^.b_endereco,
  PBufPar^.b_n_ciclos,PBufPar^.b_ppc,PBufPar^.b_timer,PBufPar^.b_cod_BD,
   PBufPar^.b_plotar,PBufPar^.b_Gravar_BD,PBufPar^.b_calcular,
    PBufPar^.b_modclient,PBufPar^.b_Pont_Graficos,
     PBufPar^.b_Pont_Medidores,PBufPar^.b_canais);
 Lista_Medidores.Add(PBufPar^.b_Pont_Medidores);
 Update_Box_Medidores;
 Envia_parametros;
end;
```

Logo após a criação da nova instância de TMedidor, a classe TCadastro formula uma mensagem de configuração para ser enviada ao medidor (comando “Envia_parametros”, acima mostrado), que irá se auto-configurar com os parâmetros determinados pelo usuário.

O medidor, assim que receber o pacote de configuração deve responder afirmando que a escrita ocorreu normalmente. Caso tenha havido algum erro ou falha (e o medidor não tiver enviado a resposta ao pacote de configuração), o software irá emitir um aviso de falha, requisitando uma opção das seguintes alternativas: Abortar o cadastramento do medidor, tentar configura-lo de novo ou ignorar o erro de comunicação. A Figura 3.10 mostra essa janela de opções.

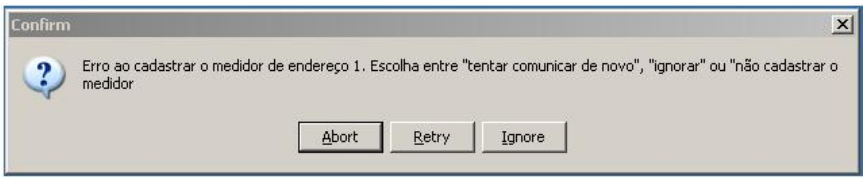


Figura 3.10 – Erro no cadastramento de um medidor.

Ainda, esta classe deve gerenciar o começo, interrupção e parada das aquisições de cada medidor. Isto é feito através dos métodos de sincronismo público “Resume” e “Suspend” da classe TThread, da qual a classe TMedidor faz parte.

3.4.2 TMedidor

A classe TMedidor é derivada da classe abstrata TThread, a qual possui um método virtual abstrato chamado “Execute”, que permite a função a ser executada seja processada em paralelo com o programa principal.

Isto foi necessário devido a cada instância da classe TMedidor ter um temporizador operando em paralelo com o programa, e também pela facilidade de iniciar e interromper esse processamento, através dos métodos “Resume” e “Suspend”.

Os parâmetros dessa classe são ajustados com o construtor “Create”, conforme mostrado acima no procedimento “procedure TCadastro.cadastrar_medidor”, e como está mostrado no código fonte abaixo, retirado de dentro da classe TMedidor.

```
constructor TMedidor.create(p_endereco, p_ciclos : byte;
  p_ppc, p_timer, p_cod_BD : word;
    p_graph, p_BD, p_calc: boolean; p_M_modclient : Tmodbusclient;
      p_Pont_grafico, p_self_pont : pointer; p_canais : array of boolean);
begin
 inherited create(true);
 ppc := p_ppc;
 ciclos := p_ciclos;
 endereco := p_endereco;
 Timer := 1000*p_timer;
 cod_BD := p_cod_BD;
 Gravar_BD := p_BD;
 Plotar := p_graph;
 Calcular := p_calc;
...
end;
```

Uma outra classe derivada de TThread, que é chamada de dentro da classe TMedidor, é a TPedido, que é a classe que vai efetivar o envio do pedido dos medidores pela rede RS485, com protocolo ModBus. O código desta última classe está mostrado a seguir.

```
constructor TPedido.create(pont_med : pointer);
begin
 inherited create(true);
 Medidor := pont_med;
end;
//=================================================//
…
  for j := 0 to Medidor.npedt-1 do
    Medidor.M_modclient.ReadHoldingRegisters(64*j,64, Medidor);
...
end;
```

Pode-se observar que na função do de envio do pedido componente “Modlink”, “Medidor.M_modclient.ReadHoldingRegisters(64*j,64, Medidor);”, o “M_modclient” é uma instância da classe TModBusClient, a qual cada medidor tem uma, devido à identificação do remetente no momento da recepção da resposta. Ou seja, quando a resposta chegar, a função de recepção (que se encontra na classe TCadastro) vai saber qual o medidor que enviou.

Outra observação relevante é o ponteiro que é passado como parâmetro na função acima mostrada (Medidor). Ele é utilizado na recepção da resposta para indicar qual medidor deve receber os dados. O código abaixo mostrado, que na verdade é retirado da classe TCadastro, demonstra a associação do ponteiro. A variável local “local_pont_med”, que é um ponteiro para a classe TMedidor, recebe o valor que está em “info.UserData”, ou seja, o endereço do medidor (instância de TMedidor) o qual deve receber os dados.

```
procedure TCadastro.ModbusClient1HoldingRegistersRead(
  Sender: TModbusClient; const Info: TTransactionInfo; StartReg,
  RegCount: Word; const RegValues: TRegValues);
var
 local_pont_med : ^TMedidor; i, length_temp : word;
 Vetor_Temp : array of word;  Canais_Temp : array[0..5] of boolean;
begin
 local_pont_med := info.UserData;
 if Info.Reply = srNormalReply then
 begin
  local_pont_med^.setvetor(startreg,regvalues);
  local_pont_med^.incnpedf;
  if local_pont_med^.p_npedt <> (local_pont_med^.p_npedf) then exit;
...
end;
```

Observa-se que os dados recebidos são escritos no medidor apontado por “local_pont_med”, através da chamada de método da classe TMedidor “local_pont_med^.setvetor(startreg,regvalues);”, sendo o “regvalues” o vetor de dados.

Mas ainda há um detalhe muito importante de ser explicado, referente à comunicação de dados. Sabendo-se que o número máximo de pontos que um medidor pode enviar em um pacote (uma transação ModBus) é de 125 pontos (limite estipulado pelo protocolo [54]), e que o número total de pontos que um medidor deve enviar é calculado pelo número de fases, vezes o número de ciclos, vezes o número de pontos por ciclos, pode-se perceber que provavelmente um pacote não suportaria enviar toda informação necessária.

Para ser mais claro, supõe-se que o software tenha cadastrado um medidor, que tem que enviar as informações das três fases de corrente e tensão, fazendo uma leitura de dois ciclos,

com 128 pontos por ciclo. Segundo a fórmula abaixo, o número de pontos é muito maior que os 125 permitidos em um único pacote.

$$6 \times 2 \times 128 = 1536\, pontos \tag{3.1}$$

Por isso, esses 1536 pontos devem ser enviados por etapas, ou seja, o software deverá fazer “n” pedidos para conseguir obter todos os pontos necessários.

Levando-se em conta que o número de pontos por ciclo comumente utilizado é múltiplo de 64 (64,128,256,512...), estipulou-se que em cada pacote seriam trafegados 64 pontos (e não os 125). Assim, no caso acima exemplificado, seriam necessários 24 pedidos para a obtenção dos 1536 pontos.

$$\frac{1536(pontos)}{64(pontos\_por\_pacote)} = 24(pacotes) \tag{3.2}$$

É por isso que no procedimento “procedure TPedido.Execute;” o número de pedidos que serão feitos é igual ao valor da variável “npedt” (parâmetro da classe TMedidor), que significa o número total de pedidos que devem ser feitos.

Da mesma forma como houve uma contagem para a realização dos pedidos, também há uma contagem para a recepção deles, pois o software deve monitorar até que momento deve esperar por respostas.

Por isso, a cada resposta recebida o software incrementa uma variável chamada “npedf” (também parâmetro da classe TMedidor), que significa o número de pedidos feitos, e monitora se o número de pedidos feitos é igual ao número de pedidos totais. Caso seja, o aplicativo saberá que a resposta recebida foi a última e, assim, pode seguir para as rotinas de gráfico, cálculo e armazenamento em banco de dados.

```
procedure TMedidor.setnpedt;
begin
 npedt := (n_canais*ciclos*(100*ppc div 64))div 100;
end;
-----------------------------------------------
procedure TMedidor.incnpedf;
begin
 inc(npedf);
end;
```

É interessante observar que a classe TMedidor é que centraliza todas as informações a seu respeito (como o número de pedidos que deve fazer, em qual pedido está, o vetor de dados e todos demais parâmetros) e que, como forma de proteger essas informações, as variáveis não são públicas, e sim privadas. E para uma outra classe consultar ou alterar os dados, deve fazê-lo através dos métodos de classe. O trecho abaixo mostra um exemplo de uso de "property", que é um recurso utilizado para proteger o conteúdo de uma variável, e uma função pública para ler um valor do vetor de dados.

```
Private
   procedure setciclos(p_ciclos : byte);
published
   property p_ciclos : byte
     read ciclos       write setciclos;
public
   function get_ponto_vetor(posicao : word) : word;
 …
procedure TMedidor.setciclos(p_ciclos: byte);
begin
    ciclos := p_ciclos;
end;
function TMedidor.get_ponto_vetor(posicao : word) : word;
begin
    result := Vetor[posicao];
end;
```

3.4.3 TGrafico

Esta classe, derivada de Tform, é instanciada para cada medidor cadastrado no sistema, e apontada por um ponteiro que é guardado como parâmetro na classe TMedidor. Ou seja, sempre que um medidor é cadastrado, uma nova janela (instância de TGrafico) é criada e associada a um medidor através de um ponteiro.

A janela de gráfico é composta por dois gráficos: um das formas de onda instantâneas trifásicas de tensão e corrente, e outro do espectro harmônico (até a 12° harmônica) de tensão ou corrente. Ainda nesta janela estão mostrados os parâmetros de performance calculados. Tanto os gráficos como os cálculos não são gerados pela classe TGrafico, mas sim pelas classes ThGrafico e ThCalculos, explicados no próximo ítem.

Pelo fato de poder haver muitas janelas de gráficos no sistema, dois recursos foram elaborados para facilitar a manipulação com elas: no menu das janelas “principal” e de “cadastro dos medidores” tem uma guia que possibilita maximizar / minimizar todas janelas, assim como escolher uma para trazer para frente, como mostrado na Figura 3.11; e também, como alternativa para distribuição das janelas e economia de espaço, é possível ao usuário escolher mostrar ou não os cálculos e o gráfico de harmônicas, como pode-se ver na Figura 3.12.

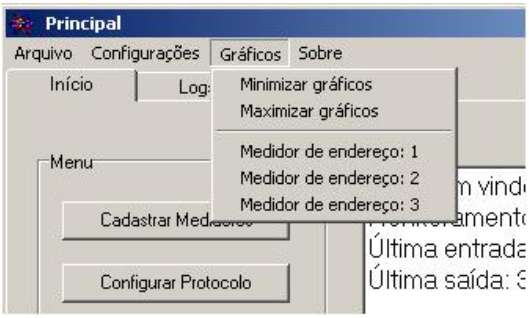


Figura 3.11 – Manipulação de janelas de gráficos.

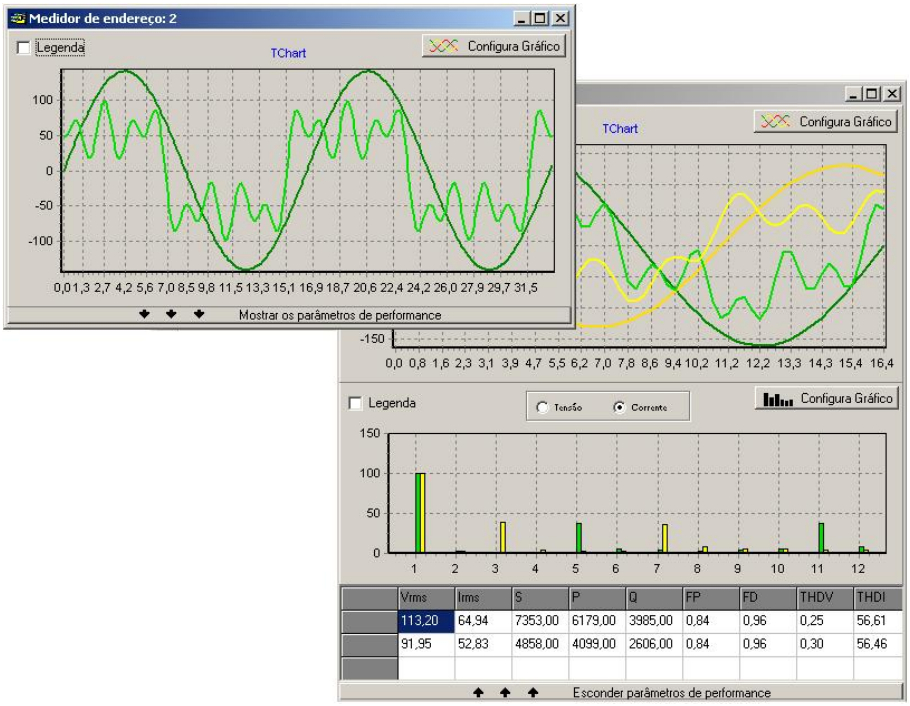


Figura 3.12 – Distribuição das janelas de gráficos

3.4.4 ThBancoDados, ThGrafico e ThCalculos

Essas três classes são derivadas da TThread, por um motivo conveniente: elas precisam executar as respectivas tarefas em paralelo com o restante do processamento, oferecendo vazão ao fluxo de dados.

O código construtor que cria uma instância de cada classe está dentro do procedimento "procedure TCadastro.ModbusClient1HoldingRegistersRead", mostrado acima. Pode-se observar que cada instância da classe está associada uma variável do seu tipo (ThBD, ThG, ThCalc), e que neste construtor "Create" são passados todos parâmetros necessários para a execução da tarefa. O código fonte que executa a função para plotar as formas de onda instantâneas está mostrado abaixo.

```
procedure ThGrafico.Execute;
var
 f,g : word;
 tm : single;
 W : double;
 cont : byte;
 fator : single;
begin
 limpagrafico;
 flag_corrente_g := n_ten_g*ppc_g*ciclos_g;
 tm := 0;
 for g := 0 to (ciclos_g*ppc_g)-1 do
 begin
  cont := 0;
  for f := 0 to 5 do
   if canais_g[f] then
   begin
    if (ciclos_g*ppc_g*cont+g) >= flag_corrente_g then
     fator := relacao_TPTC.relacao_TC*619.3
    else
     fator := relacao_TPTC.relacao_TP*6.266;
    W := (Vetor_Graph[(ciclos_g*ppc_g*cont+g)]-2047)/fator;
    local_pont_gra^.series[f].Add(W,floattostrf(tm,fffixed,4,1));
    inc(cont);
   end;
  tm := tm+(1/60)/ppc_g*1000;
 end;
end;
```

Deve-se observar que há uma variável "fator" que aparece dividindo o valor do ponto (armazenado em "W"). Também se pode ver que ela assume um valor, dependendo do ponto que está vigente. Para ser mais claro, a comparação "if(ciclos_g*ppc_g*cont+g)>=flag_corrente_g" verifica se o atual ponto é referente à corrente

ou tensão. Se for o primeiro caso, a variável fator será o valor da “relação TC” vezes 619,3, e no outro caso será a “relação TP” vezes 6,266. E isso é feito para o ajuste do sinal para a representação real, pois o valor que o micro-controlador envia é o valor do “AD”, como explicado no capítulo 4 (Desenvolvimento do Hardware). Por isso aparece uma subtração de 2047 do valor do ponto, e depois a divisão pelo “fator”. Mas ainda deve-se deixar claro as variáveis “relação_TP” e “relação TC” não fazem parte desse retorno do tratamento de sinais, e sim uma ferramenta para o usuário conseguir ver as formas de onda de tensão e corrente de forma mais cômoda.

Isso é feito porque, na maioria das vezes, a tensão terá um valor (na escala do gráfico) muito maior que a corrente. Por exemplo, se um sinal de tensão com valor de pico de 181 volts fosse plotado junto com um de corrente, com valor de pico de 1 ampère, a forma de onda deste último seria imperceptível (na verdade, iria parecer uma reta). Por isso, o software oferece a possibilidade de multiplicação ou divisão aos sinais de corrente, na magnitude escolhida pelo usuário. A seguir, está mostrada a janela de configuração das relações de TP e TC. Outra explicação para a utilização deste recurso é a utilização de “TPs” e “TCs” externos ao hardware, podendo-se assim reajustar os parâmetros via software.

Figura 3.13 – Janela de configuração das relações “TP” e “TC”.

A classe ThCalculos, que é a que realiza os cálculos dos parâmetros de performance, utiliza uma DLL para carregar as rotinas de cálculo de um arquivo externo, não estando junto ao código fonte do programa. Isto foi feito pelo seguinte motivo: se os algoritmos dos cálculos forem atualizados (outros métodos de cálculos podem ser empregados), o usuário do sistema não precisa receber e reinstalar todo software. Basta instalar a nova DLL (que é um arquivo muito mais leve que o programa inteiro) para a atualização dos algoritmos.

A declaração das funções da DLL e um exemplo de chamada de uma função (cálculo do valor RMS de tensão e corrente, no caso) estão mostrados a seguir. As rotinas para os cálculos do parâmetros de performance (que se encontram nesse arquivo externo) e a representação dos algoritmos utilizados estão apresentadas no APÊNDICE C, e a teoria dos algoritmos de cálculo utilizados pode ser encontrada em [48].

A Figura 3.14 mostra alguns valores de cálculos obtidos.

```
function Calcula_rms(vet : array of single; ppc : word;
 ciclos : byte; freq : single) : single;
    stdcall; external 'Calc_parametros.DLL'
function Calcula_PAparente(Trms, Crms : single) : single;
    stdcall; external 'Calc_parametros.DLL'
function Calcula_PAtiva(ten, corr : array of single; ppc : word;
 ciclos : byte; freq : single) : single;
    stdcall; external 'Calc_parametros.DLL'
function Calcula_PReativa(Paparente, Pativa : single) : single;
    stdcall; external 'Calc_parametros.DLL'
function Calcula_FP(Pativa, PAparente : single) : single;
    stdcall; external 'Calc_parametros.DLL'
function Calcula_FD(ten, corr : array of single; ppc : word;
 ciclos : byte; freq : single) : single;
    stdcall; external 'Calc_parametros.DLL'
function Calcula_Distorcao(vet : array of single; ppc : word;
 ciclos : byte; freq, VRms : single) : single;
    stdcall; external 'Calc_parametros.DLL'
function Calcula_harmonicas(vet : array of single; ppc : word;
 ordem_harmo, ciclos  : byte; freq: single) : single;
    stdcall; external 'Calc_parametros.DLL'
//=============================================//
if canais_c[b] then
  begin
   temp_rms[b] := Calcula_rms(Vetor_fases[b],ppc_c,ciclos_c,freq_c);
   local_pont_gra^.Tab_Params.Cells[1,(b+1)] :=
                floattostrf(temp_rms[b],fffixed,4,1);
  end;
 for b := 3 to 5 do
  if canais_c[b] then
  begin
   temp_rms[b] := Calcula_rms(Vetor_fases[b],ppc_c,ciclos_c,freq_c);
   local_pont_gra^.Tab_Params.Cells[2,(b-2)] :=
                floattostrf(temp_rms[b],fffixed,4,1);
  end;
```

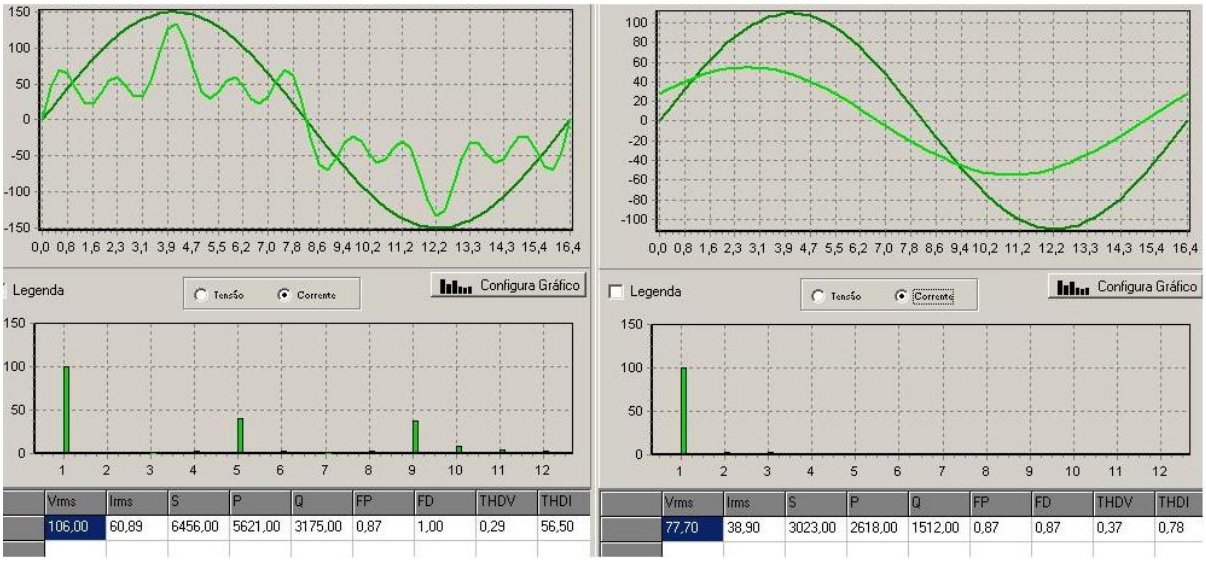


Figura 3.14 – Janelas apresentando os parâmetros de performance.

3.5 Comunicação com o Banco de Dados

A comunicação com o banco de dados se dá somente para a escrita de dados, ficando a parte de cadastros de empresas, equipamentos e pontos para a página da internet, assim como a consulta aos dados. Isto foi feito para economizar esforço de implementação no software, já que a página da internet teria que ter essa interface de cadastro dos medidores.

A parte operacional para a escrita no banco é relativamente simples, e se dá da seguinte forma: quando o medidor recebe um pacote de resposta (pela rede RS485), ele verifica o parâmetro “Gravar em banco de dados”. Se este for verdadeiro o software chama a função para gravar no banco, mas se for falso, não o faz.

A função para gravar no banco de dados é também uma Thread, onde os dados são convenientemente separados em vetores e escritos no banco, através de comandos SQL. Os parâmetros para a escrita no banco são: o código do ponto no banco (cada medidor cadastrado tem um código do banco de dados como parâmetro), a fase a que pertence esse ponto (A, B ou C), a data e a hora da medição, e o valor do ponto de tensão e/ou corrente. Ou seja, a cada ponto deve ser gerado um comando SQL.

Entretanto, se esse comando for executado a cada ponto dos vetores, o processo torna-se lento, por não facilitar o fluxo de dados. Para a resolução deste problema, a cada ponto é gerado um comando SQL, mas não executado, e sim anexado a um conjunto de comandos SQL do componente “query” (componente do delphi destinado a fazer escritas e leituras do banco), para depois serem executadas ao mesmo tempo. Para ser mais claro, em vez do software escrever e esperar confirmação de escrita 64 vezes no banco de dados (valor fictício utilizado como exemplo), ele vai enviar um conjunto de comandos SQL e esperar uma confirmação apenas. A diferença de performance de um método para o outro é considerável. Abaixo está mostrado o código utilizado para o processo acima relatado.

```
procedure ThBancoDados.escrevebancodados;
var
 f : word;
begin
 BD.Query1.sql.Clear;
 if (Canais_BD[0] or Canais_BD[3]) then
  for f := 0 to Tamanho_vet-1 do
   writesql(...)
   ...
 BD.Query1.ExecSQL;
end;
//=============================================//
procedure ThBancoDados.writesql(v_Fase, v_DataHora : string;
                v_ponto : word; v_tensao, v_corrente : single);
begin
 BD.query1.sql.Add(…);
end;
```

3.6 - Medidor Virtual

O medidor virtual tem esse nome porque foi implementado em software (Delphi 5), com o intuito de simular a operacionalidade do medidor real. Ou seja, todas formas de onda adquiridas por ele são provenientes de funções matemáticas.

Essa idéia surgiu pelo fato de a programação de software ser de mais fácil de implementação, comparada com a do hardware, e principalmente, de muito mais fácil correção de erros. Também, como o hardware teria uma inércia inicial até o funcionamento da comunicação serial (o que realmente não foi trivial de dominar) e da instrumentação para a

aquisição dos sinais elétricos, o medidor virtual veio à tona, permitindo que software e hardware estivessem paralelamente em desenvolvimento.

Ainda, a possibilidade de se obter senóides puras, com harmônicos e/ou com defasagem entre tensão e corrente permitiram que os algoritmos para o cálculo dos parâmetros de performance fossem validados.

3.6.1 - Código Fonte

De modo a oferecer a mesma funcionalidade do medidor real, o virtual é configurado por meio de um pacote de configuração enviado pelo software. Esse pacote contém informações de quais fases adquirir, com quantos ciclos e o número de pontos por ciclo. Mas além da configuração via pacote de comunicação, o aplicativo permite ainda que as configurações sejam realizadas localmente.

O trecho do programa abaixo mostra uma função que é chamada para cada registrador que tenha vindo no pacote de comunicação. Pode-se observar que a variável “RegAddr” indica a posição deste. Então, como se pode verificar, o primeiro registrador contém o valor do número de pontos por ciclo (p_Buf^.b_ppc), o segundo o número de ciclos (p_Buf^.b_ciclos) e os demais, de 2 a 7, informam por meio do valor 1 ou 0 (sim ou não, respectivamente) quais fases devem ser adquiridas.

```
procedure TPrincipal.ModbusServer1SetHoldingRegisterValue(
  Sender: TModbusServer; RegAddr, RegValue: Word);
begin
 case regaddr of
  0 : pBuf^.b_ppc := regvalue;
  1 : pBuf^.b_ciclos := regvalue;
  2 : if regvalue = 1 then pBuf^.b_canais[0] := true else pBuf^.b_canais[0] := false;
  …
end;
 if regaddr = 7 then atualiza_med;
end:
```

A expressão “p_Buf^.” que aparece antes das variáveis dos parâmetros é um ponteiro, que aponta para as variáveis da classe TBuffer, como mostrado a seguir. Observa-se que “vBuf” é uma variável instância da classe TBuffer, e “p_Buf” recebe o endereço de “v_Buf”, através do operador @.

```
TBuffer = class
  b_ppc : word;
  b_ciclos : byte;
  b_canais : array[0..5] of boolean;
end;
-----------------------------------------------------------
Pbuf : ^TBuffer;
vBuf : TBuffer;
-----------------------------------------------------
vBuf := TBuffer.Create;
pBuf := @vBuf;
```

Após o 7° registrador ter sido recebido, uma função “atualiza_med” é chamada, a qual irá, entre outras funções, chamar uma outra função, a “cria_AD”.

```
procedure TPrincipal.atualiza_med;
var
 b : byte;
begin
 cria_AD;
 edit2.Text := inttostr(pBuf^.b_ppc);
 edit1.text := inttostr(pBuf^.b_ciclos);
 for b := 0 to 5 do
  box_canais.Checked[b] := pBuf^.b_canais[b];
end;
```

Essa função “cria_AD” na verdade cria uma instância da classe derivada de TThread chamada de “sinais”, sendo que as configurações necessárias para o funcionamento dela são passados como parâmetros, como o número de ciclos e pontos por ciclo, o vetor boleano contendo a informação das fases, a freqüência dos sinais, o endereço do medidor e demais informações para a inserção de harmônicos e defasagens.

```
procedure TPrincipal.cria_AD;
begin
 if AD <> nil then
 begin
  AD.Suspend;
  AD.Terminate;
  AD.Free;
 end;
 AD := sinais.create(…);
 checkbox1.Checked := true;
 AD.Resume;
end;
```

A seguir, será mostrada a classe “Sinais”, a qual gera os pontos das senoides por meio de cálculos matemáticos e os organiza em um vetor único, que será mapeado na hora de um pedido de dados.

```
for i := 0 to (pontos_ciclo*n_ciclos)-1 do
  begin
   c := 0;
   if canais[0] then
   begin
    vetor[...] := trata_sinal(10*rand1*
     (sin(pi*120*(t+Def_T_A)+0)+
      0.4*sin(120*pi*(t+Def_T_A)*Vet_harmo_A[0]*3)+
       0.4*sin(120*pi*(t+Def_T_A)*Vet_harmo_A[1]*5)+
        0.4*sin(120*pi*(t+Def_T_A)*Vet_harmo_A[2]*7)+
         0.4*sin(120*pi*(t+Def_T_A)*Vet_harmo_A[3]*9)+
          0.4*sin(120*pi*(t+Def_T_A)*Vet_harmo_A[4]*11)+
           0.4*sin(120*pi*(t+Def_T_A)*Vet_harmo_A[5]*13)));
    inc(c);
   end;
....
-----------------------------------------------------------------
function trata_sinal(ponto_flut : extended): integer;
var
 S : string;
begin
 S := FloatToStrF(ponto_flut*6.266,fffixed,6,0);
 result := (strtoint(S)+2047);
end;
```

Pode-se reparar que a função matemática que gera os pontos é um somatório de senos, sendo estes de diferentes freqüências (a fundamental, e as harmônicas de 3°, 5°, 7°, 9°, 11° e 13° ordens), multiplicados por uma amplitude, que tem na sua composição uma variável chamada “rand1”. Esta, é um número que fica variando randomicamente, de acordo com a expressão “rand1 := 10+random(8);”. Assim, o valor da amplitude pode variar de 120 a 180 (Volts). Também deve ser dito que esse processo é refeito a cada 4 segundos, que foi um valor considerado adequado. Abaixo é mostrado o método “Execute” da classe “Sinais”, onde aparece o temporizador para o processo.

```
procedure Sinais.Execute;
begin
 while not(suspended) do
 begin
  synchronize(constroi_sinais);
  sleep(4000);    // 4 segundos
 end;
end;
```

Ainda a respeito da geração dos pontos, pode-se observar que existem duas variáveis que não foram explicadas. Uma é “Def_T_A”, que é um valor somado à variável da função, ou seja, ao incremento do tempo. Essa variável somada ao “passo” de incremento causa uma defasagem entre esse sinal e o outros.

A segunda variável é a “Vet_harmo_A”, que é um vetor de 12 posições, contendo os valores um ou zero. Assim, se o valor for zero, a harmônica correspondente não existirá (pois seno (0) = 0). E também deve-se atentar para que as posições de zero a cinco são referentes à tensão, e de seis à onze são utilizadas para a corrente.

O código fonte mostrado foi ilustrativo somente da fase “A” de tensão. Entretanto, o mesmo ocorre com as outras fases de tensão e corrente, com as suas respectivas variáveis (“Vet_harmo_B”, “Vet_harmo_C”, “Def_T_B”, “Def_T_C”, “Def_C_A”...).

Pode-se ainda observar que todos os pontos antes de serem armazenados recebem um tratamento (através da função “trata_sinal”), para que os pontos não tivessem nem casas decimais, nem valores negativos, de forma que pudessem ser enviados pela porta serial.

Focando-se agora nas harmônicas e na defasagem dos sinais, como se pôde perceber o aplicativo permite ao usuário escolher quais sinais terão quais harmônicas e quais defasagens. Isso se deve à classe “config” (derivada de TForm), onde se pode visualmente realizar essas configurações. A Figura 3.15 ilustra o citado.

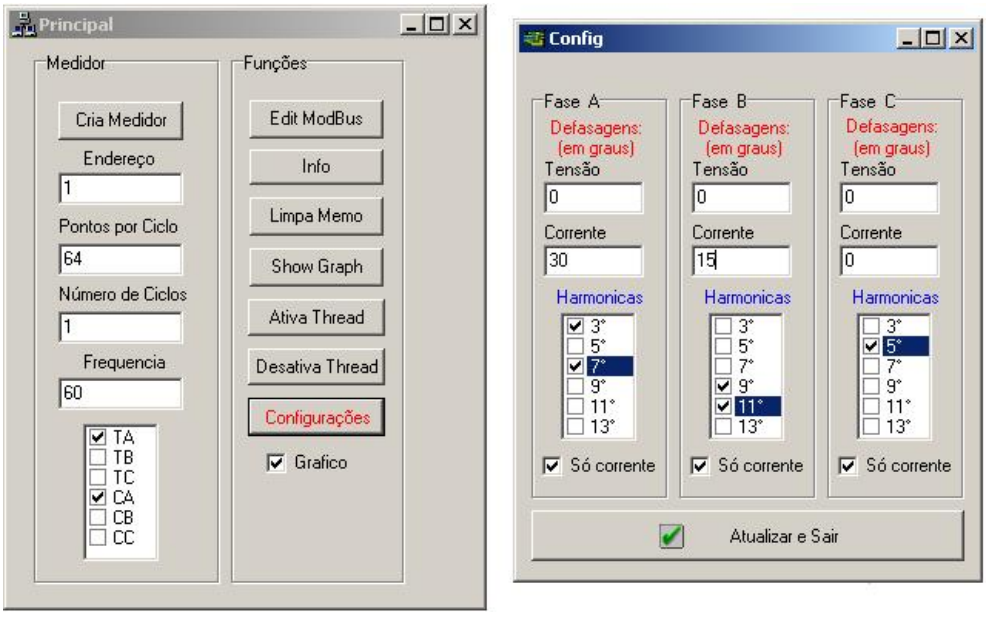


Figura 3.15 – Configurações do medidor virtual.

Desta forma pode-se visualizar as formas de onda localmente, o aplicativo também mostra em gráfico as formas de onda, como mostrado abaixo.

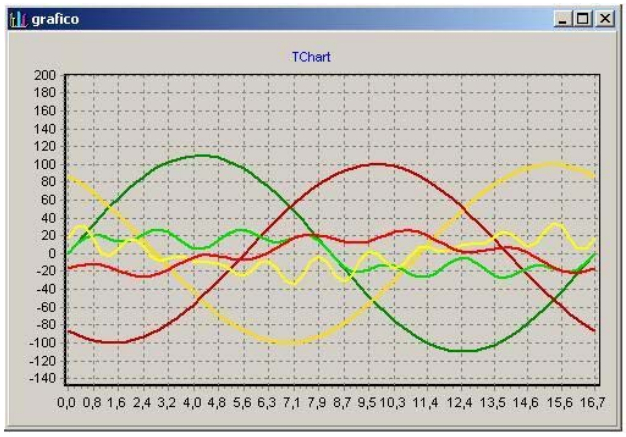


Figura 3.16 – Gráfico do medidor virtual.

3.7 Resultados

A funcionalidade do aplicativo final ofereceu resultados muito bons. O usuário do software pôde realizar o cadastramento dos medidores com flexibilidade no sistema, ou seja, de acordo com a necessidade, em termos de número de ciclos e pontos por ciclo, assim como a periodicidade das aquisições.

A forma “multi-janelas” de se visualizar os gráficos e cálculos permitiu ao usuário ter acesso às informações elétricas em diversos pontos de uma planta industrial ao mesmo tempo, como mostrado na Figura 3.9, possibilitando uma tentativa de entendimento do efeito de cada máquina na rede elétrica, por exemplo.

Entretanto, olhando pelo lado das limitações, é claro que quanto maior o número de medidores cadastrados no sistema, levando-se em conta também a periodicidade das aquisições e o número de pontos a adquirir, o software pode apresentar deficiências no fluxo de dados devido à velocidade de comunicação utilizada ser limitada a valores relativamente baixos (para os dias de hoje). Porém, este ponto deve ser considerado como uma oportunidade para o contínuo desenvolvimento do sistema.

3.8 Conclusões

Ao final das etapas de implementação, testes, mudanças e manutenções do aplicativo, pode-se concluir que todas as técnicas de “engenharia de software” utilizadas foram válidas e muito importantes nas fases de projeto, implementação e manutenção.

Entretanto, percebeu-se que na fase de implementação e testes do código fonte foram feitos muitos ajustes nos diagramas desenvolvidos na fase de projeto, o que leva à conclusão de que a etapa de implementação deve começar antes do término do planejamento.

Observou-se que a forma mais inteligente de se criar um aplicativo é seguindo o método espiral, que consiste em realizar as etapas “planejamento”, “análise de riscos”, “engenharia” e “avaliação do cliente”, nesta ordem, e recomeçar o espiral pelo planejamento, até que o projeto esteja em um nível satisfatório.

Ainda sobre o projeto do sistema baseado nas técnicas de engenharia de software, deve-se ressaltar que foram utilizados somente alguns recursos dos muitos existentes, pelo fato deste sistema não ser tão complexo, e assim, não necessitar muita documentação.

Desta forma, a estruturação feita proporcionou um desenvolvimento modularizado e organizado da programação, permitindo posteriores ajustes e incremento de recursos, assim como ofereceu, durante a fase de testes, facilidade na resolução de eventuais problemas. Outra vantagem dessa organização é a possibilidade de reutilização de código para implementações futuras.

Como alteração necessária para o incremento de desempenho do sistema está a forma de comunicação de dados, que atualmente é feita por rede RS485 com protocolo “ModBus”, e que oferece taxas de transmissão de 19200 bps, mas pode-se utilizar este mesmo protocolo em uma rede “Ethernet”, sob o protocolo “TCP/IP”.

4 DESENVOLVIMENTO DO HARDWARE

De acordo como foi apresentado no item 2.1.1 (referente às possibilidades de hardware), muitos são os recursos possíveis de se agregar aos medidores digitais de um sistema de monitoramento de energia. Como foi visto, recursos como a forma de comunicação de dados, a forma de processar as informações e apresentar resultados, a quantidade de dados suportada, enfim, os recursos possíveis podem ser mais complexos ou mais simples, dependendo da filosofia do sistema, o que afeta diretamente no custo do dispositivo.

O medidor digital de um sistema de monitoramento de energia deve ser projetado de forma a corresponder ao nível tecnológico proposto. Ou seja, se este tiver recursos avançados, como analisar continuamente as formas de onda dos sinais elétricos em busca de algum evento ocorrido, memória de massa com capacidade de armazenar informações elétricas por dias ou meses, displays sofisticados que permitem a reprodução das formas de onda, comunicação sem fio, entre muitos outros recursos possíveis, os medidores digitais terão certamente um custo maior.

Para efeitos de comparação, foram feitas quatro cotações de medidores digitais de sistemas de monitoramento de empresas reconhecidas no mercado nacional, mostradas a seguir.

1. Exemplo 1 – Analisador para medição de distúrbios em média e baixa frequências em tensão e corrente, cálculos vetoriais, aritméticos, demanda, harmônicos e flicker.
   Valor: R$5500,00
   Software: R$1200,00

2. Exemplo 2 – Medidor de energia para registrar tensões e correntes, VA, W(hora, total, ponta e fora ponta), VAR-hora, fator de pontência, demanda, THDs, freqüência, tensão e corrente máximas e tensão mínima.
   Valor: R$4700,00
   Software: R$1000,00
   Treinamento: R$500,00
3. Exemplo 3 – Analisador para levantamento de curvas de carga, diagnóstico energético, qualidade de energia (Sag, Swell), rateio de custos, diagnóstico de distorções harmônicas, correção do fator de potência, levantamento de demandas e conservação de energia.
   Valor: R$4419,00
   Sensores de corrente opcionais: R$2688,00
4. Exemplo 4 – Medidor de painel das grandezas elétricas mais elementares.
   Valor: R$ 700,00

É possível concluir, a partir dos dados acima apresentados, que o custo para distribuir analisadores de energia em uma planta industrial é considerável, envolvendo um investimento relativamente alto, o qual pode ser inacessível para empresas de recursos limitados.

A partir do preço médio dos medidores mais avançados exemplificados (por volta dos R$5000,00) e do mais básico, pôde-se dimensionar um hardware de custo intermediário.

4.1 Objetivo do Medidor

Como foi proposto na introdução, o sistema em desenvolvimento tem como objetivo aliar um baixo custo do sistema a um adequado nível tecnológico e a um razoável número de recursos, de forma a que o usuário possa visualizar as formas de onda “online” de diversos pontos de medição de uma planta industrial, acompanhando os parâmetros de performance, e armazená-los em um banco de dados, com a possibilidade de visualização das informações de qualquer lugar geográfico, através de um navegador da internet. Assim, o importante neste ponto do trabalho é entender quais dispositivos do hardware poderiam colaborar para o baixo custo do medidor digital.

De forma mais específica, o medidor digital tem como objetivo fornecer as informações que lhe são pedidas pelo software gerenciador. Ou seja, ele deve ser capaz de ler

continuamente as tensões e correntes trifásicas, mantê-las em memória e, quando requisitado, enviar os vetores representativos das formas de onda pela rede de comunicação, utilizando o protocolo escolhido.

Como já explicado no item 3.4.1, assim que ligado, o medidor aguarda um pacote de configuração enviado pelo software, o qual conterá as informações de quantos ciclos devem ser adquiridos, quantos pontos por ciclo a capturar e quais as fases de tensão e corrente devem ser armazenadas. A partir dessas configurações, o medidor se auto-parametriza, de forma a sempre fazer a leitura dos sinais elétricos nestes moldes. E se outro pacote de configuração for recebido, o medidor se auto-configura novamente.

Após configurado, o medidor deve continuamente ler as fases de interesse, de acordo com o número de ciclos e pontos por ciclo configurados, e armazená-las em memória, sempre monitorando a rede de comunicação para verificar se nenhum pedido foi realizado, devendo ser atendido.

A leitura dos sinais de tensão e corrente se dá por meio da conversão do sinal analógico em digital, processo conhecido como digitalização do sinal, e que é efetuada por meio de amostragens, onde a taxa de amostragem está diretamente vinculada ao número de pontos capturados em um ciclo. Quanto maior o número de pontos amostrados, maior é a resolução do sinal, possibilitando com maior precisão a busca de distúrbios de curta duração, como “spikes” e outros transientes.

A Figura 4.1 mostra o efeito da resolução na leitura de um sinal elétrico. Pode-se observar que quanto maior é o número de pontos por ciclo, maior é a visibilidade de um distúrbio de alta freqüência. Entretanto, de forma oposta, quanto menor o número de pontos por ciclo, menores são as chances de um evento ser bem representado, e ainda pode nem ser captado, passando despercebido pela amostragem do sinal. O segundo transiente que aparece no primeiro gráfico, por exemplo, não aparece nos outros gráficos, por ter um período muito pequeno (ocorre entre dois pontos em uma amostragem de 512 pontos por ciclo, tendo assim uma freqüência de 30,720 KHz).

Por outro lado, um evento de duração maior, como a falta de tensão (ou corrente) mostrada nos gráficos, é perceptível para todas taxas de amostragem exemplificadas. E ainda assim, a menor resolução (32 pontos por ciclo) não mostra que é uma interrupção abrupta, escondendo o que realmente está acontecendo no sinal elétrico.

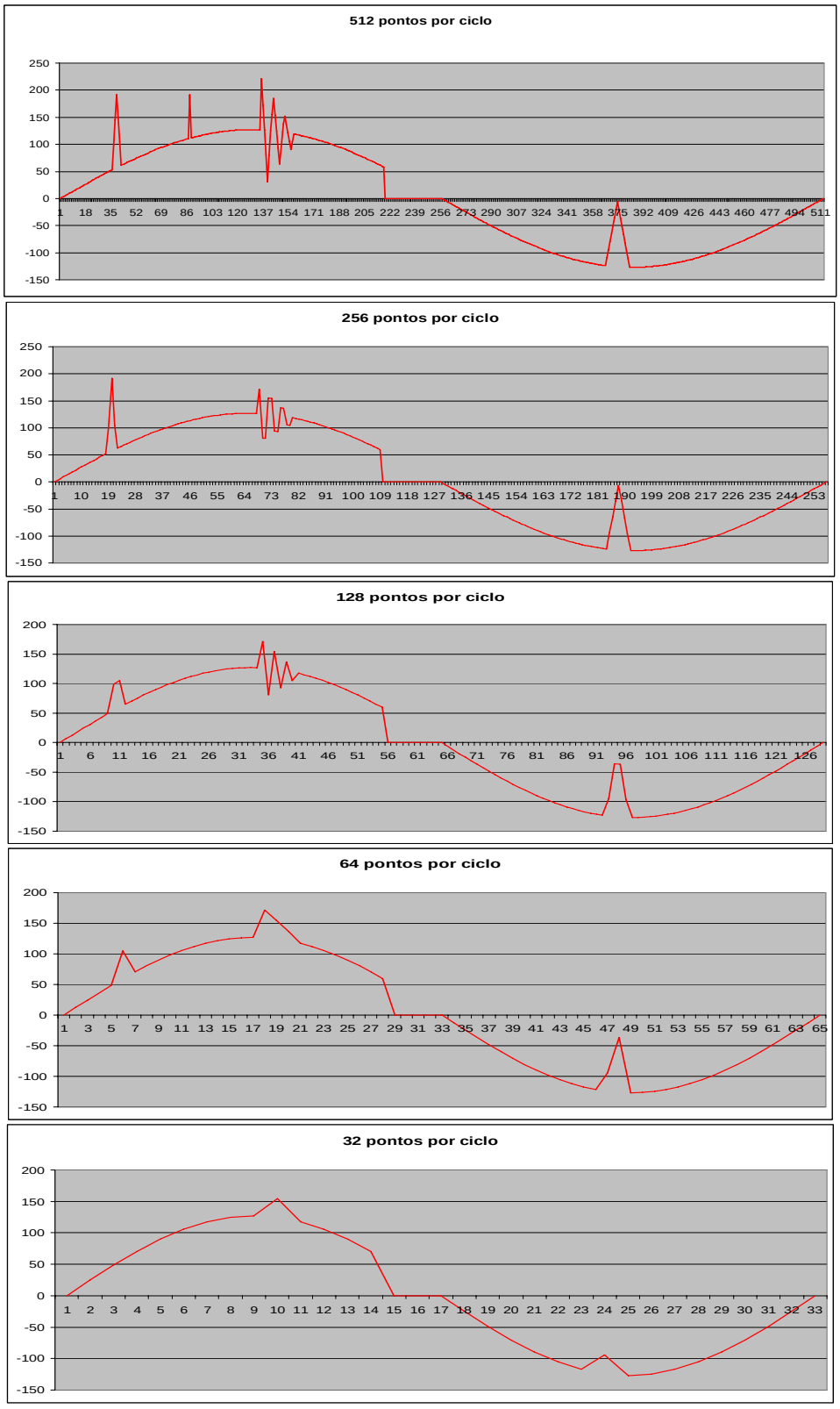


Figura 4.1 – Efeito da taxa de amostragem do sinal elétrico.

A partir desse pequeno estudo sobre a importância da taxa de amostragem para uma adequada representação de um sinal elétrico, pode-se concluir que quanto maior o número de pontos por ciclo melhor. E desconsiderando o incremento no processamento e a quantidade de dados que deve ser transferida do medidor para o software, realmente é mais vantajoso.

Porém, esses dois fatores citados não podem ser desconsiderados. A Tabela 4.1 mostra a relação entre o número de pontos por ciclo com o tempo para a transferência das informações do medidor para o software, utilizando uma taxa de transmissão de 19200bps.

Tabela 4.1 – Relação do número de pontos por ciclo com o período para a transferência dos dados para o software.

Número de fases	Número de pontos por ciclo	Número de ciclos	Número total de pontos	Número total de Bytes	Número de pedidos necessários	Tempo para a transferência de todos dados, em segundos (19200 bps)
6	32	1	192	384	6	0,4846875
6	64	1	384	768	12	0,969375
6	128	1	768	1536	24	1,93875
6	256	1	1536	3072	48	3,8775
6	512	1	3072	6144	96	7,755
6	1024	1	6144	12288	192	15,51
Observações:	Pacote de pergunta	Pacote de resposta	Bits por Byte	Bits por transação pergunta / resposta		
Modbus RTU	8 bytes	133 bytes	11 bits	141 bytes * 11 bits/byte = 1551 bits		

Na Tabela 4.1, pode-se verificar que a partir de 256 pontos por ciclo o período para a transferência dos dados começa a ficar crítico, podendo oferecer complicações para o fluxo de dados e desempenho do sistema.

Pelo fato de a taxa de amostragem não ser a única variável, a Tabela 4.2 mostra a relação do período para transferência dos dados com o número de ciclos a serem capturados, para um número fixo de 64 pontos por ciclo.

Considerando a comparação entre o número de pontos por ciclo, o número de ciclos e o período para a transferência dos dados, fez-se uma projeção de como seria se a transmissão se desse por “Ethernet”, a 10Mbps. Isto está mostrado na Tabela 4.3, onde se pode observar que a redução drástica no período para transferência. Por exemplo, o período necessário para a

comunicação 1 ciclo, 6 fases e 256 pontos por ciclo a 19200 bps, era de 3,87 segundos. A 10Mbps, com a mesma taxa de amostragem e no mesmo período, seriam possíveis 520 ciclos.

Tabela 4.2 - Relação do número de ciclo com o período para a transferência dos dados.

Número de fases	Número de pontos por ciclo	Número de ciclos	Número total de pontos	Número total de Bytes	Número de pedidos necessários	Tempo para a transferência de todos dados (19200 bps)
6	64	1	384	768	12	0,969375
6	64	2	768	1536	24	1,93875
6	64	4	1536	3072	48	3,8775
6	64	8	3072	6144	96	7,755
6	64	12	4608	9216	144	11,6325
6	64	16	6144	12288	192	15,51

Como será visto mais a frente, a comunicação escolhida para o sistema foi a rede RS485. Esta comparação é feita pois a migração para a comunicação via “Ethernet” é um dos desenvolvimentos futuros para o medidor digital, não tendo sido adotada agora pelo fato de o projeto visar o baixo custo e uma análise periódica dos sinais elétricos, e não tão intensiva, como esta comunicação permitiria.

Tabela 4.3 – Projeção do período de comunicação para uma rede Ethernet.

Número de fases	Número de pontos por ciclo	Número de ciclos	Número total de pontos	Número total de Bytes	Número de pedidos necessários	Tempo para a transferência de todos dados (10 Mbps)
6	32	1	192	384	6	0,0009306
6	64	1	384	768	12	0,0018612
6	128	1	768	1536	24	0,0037224
6	256	1	1536	3072	48	0,0074448
6	512	1	3072	6144	96	0,0148896
6	1024	1	6144	12288	192	0,0297792
6	256	520	798720	1597440	24960	3,871296

4.2 Análise dos Recursos

De modo a se obter um hardware de custo adequado com a proposta do sistema, alguns módulos integrantes do mesmo foram analisados, sendo citados a seguir.

Quanto à comunicação, conforme mostrado no item 2.1.1.3.2, muitas formas são possíveis para a transferência de dados do medidor para o software. E por isso, deve-se analisar qual tecnologia é suficiente para a necessidade, sem haver um super-

dimensionamento. Para tal, uma tabela comparativa das características das comunicações de possível escolha está mostrada a seguir.

Tabela 4.4 – Comparativo das comunicações mais prováveis.

	Comunicação		
Característica	Serial ponto a ponto	RS485	Ethernet
Custo	Baixo	Moderado	Alto
Possibilidade de formação de rede	Não	Sim	Sim
Taxa de transmissão	Razoável	Razoável	Muito boa
Facilidade de implementação na industria	Fácil	Moderada	Difícil
Confiabilidade	Confiável	Confiável	Bastante confiável

Pode-se perceber que em termos de custo e facilidade de implementação na indústria, a comunicação serial ponto a ponto seria a mais conveniente. Entretanto, é desejado ao sistema a formação de uma rede de medidores para se comunicarem com o software gerenciador, eliminando-se assim esta opção.

Em termos de taxa de transmissão e confiabilidade, a comunicação via Ethernet é bastante atraente, com capacidade de manipular uma grande quantidade de dados de forma ágil (10 ou 100 Mbps). Mas por outro lado, o custo e a facilidade de implementação na indústria não são os mais adequados. Como o objetivo do sistema é adquirir as informações de um ponto de medição periodicamente, em forma de amostras dos sinais elétricos, sendo assim um volume moderado de dados, mas não um grande volume, o meio físico para a transmissão de dados escolhido para o sistema foi o RS485.

Outros pontos integrantes do hardware que colaboram diretamente para o ajuste do custo do medidor são o display, o teclado, a alimentação, os circuitos para o tratamento de sinais e os sensores de tensão e corrente. Todos esses dispositivos foram dimensionados de forma a oferecer as funcionalidades desejadas, mas visando o baixo custo. Eles não serão explicados aqui, pois serão mostrados mais adiante.

Contudo, o ponto mais importante de se analisar é a escolha do micro-controlador a ser utilizado. Tem-se como referência para o projeto o baixo custo. Entretanto, o medidor deve ser capaz de realizar as tarefas e os processamentos necessários, como a leitura de tensões e

correntes trifásicas, a comunicação dos dados, alguns cálculos de parâmetros de performance, etc., assim como possuir recursos como memória “RAM”, “AD” de 12 bits, “Timers”, porta serial, interrupções, entre outros, sem haver deficiência tecnológica.

Assim, foi realizado um levantamento de quais micro-controladores poderiam ser utilizados, e a escolha do mais conveniente, mostrados a seguir.

4.3 Escolha do Microcontrolador

Existem diversos fabricantes de micro-controladores que poderiam ser utilizados no projeto. E por isso, a dificuldade estava em escolher qual componente tinha as melhores características, visando o custo/benefício, de acordo com o que se necessita no sistema.

Os principais pontos a serem analisados para a convergência das necessidades são:

- Arquitetura: teve-se preferência por arquiteturas que utilizassem a forma de processamento “RISC”, como o “ARM7” ou “Von Neumann”, por serem mais simples e por poderem utilizar uma freqüência de operação mais alta, além de serem mais baratos.

- Conversor “Analógico-Digital”: seguiu-se a norma “Aneel 505”, a qual indica que o conversor “AD” deve ser de no mínimo 12 bits.

- Freqüência de conversão do “AD”: devido à leitura trifásica, onde os pontos das diferentes fases devem ser adquiridos rapidamente, buscou-se por freqüência do “AD” bem maiores que a da rede elétrica.

- Milhões de instruções por segundo (MIPS): pelo fato de o micro-controlador dever oferecer um bom fluxo de dados (processamento), procurou-se por dispositivos com elavada relação “CLOCK” X “PLL”.

- Memória “flash”: de modo a se poder programar em “linguagem C”, buscou-se por uma memória “flash” que suportasse o tamanho do programa desenvolvido.

- Memória “RAM”: Pelo elevado número de variáveis utilizadas, e também pela quantidade de dados necessárias de se manipular, buscou-se por uma memória da ordem de 2Kb. Este cálculo é uma estimativa para o seguinte caso: 6 fases, 1 ciclo por fase, 64 pontos por ciclo e 2 bytes por ponto, resultando em 768 bytes.

- Interface Serial: devido à comunicação de dados se dar por uma porta serial, o micro-controlador deveria possuir essa interface.

- Portas de entrada e saída: Pelo fato de o hardware ter display, botões e comunicação com memória externa, o número de portas de I/O deveria suportar toda essa interface.

- Custo: Por se buscar o menor custo para o hardware, o preço de mercado do micro-controlador foi levado bastante em consideração.

A Tabela 4.5 mostra as características de alguns micro-controladores pesquisados, para a realização da escolha do melhor custo/benefício.

Tabela 4.5 – Escolha do micro-controlador do hardware.

	Micro-controladores				
Característica	LPC2212FBD144	AT91SAM7A3	ADUC7026	STR736FV1T7	MSP430F169
Fabricante	Philips	Atmel	Analog Devices	STMicroeletronics	Texas Instruments
Arquitetura	ARM7TDMI / RISC	ARM7TDMI / RISC	ARM7TDMI / RISC	ARM7TDMI / RISC	Von Neumann / RISC
Memória RAM	16k	32k	8k	16k	2k
Memória Flash	128k	256k	62k	256k	64k
Resolução do AD	10 bits	10 bits	12 bits	10 bits	12 btis
Frequencia de amostragem do AD	410k amostras por segundo	384k amostras por segundo	1M amostras por segundo	10M amostras por segundo	200k amostras por segundo
Frequencia máxima de operação	60MHz	60MHz	44MHz	36MHz	8MHz
Interface Serial	Sim	Sim	Sim	Sim	Sim
Custo (1000 unidades)	$8,18	$9,70	$8,80	$8,45	$6,72
Familiaridade com o componente	Baixa	Rasoável	Rasoável	Baixa	Boa

A Tabela 4.5 apresentada mostra somente alguns dispositivos disponíveis no mercado. Cada fabricante citado, e inclusive outros não mencionados, possuem diversos modelos e variações de micro-controladores.

Pode-se reparar que somente dois modelos apresentados têm 12 bits como resolução do "AD", o que já restringe consideravelmente a escolha. A arquitetura, freqüência do AD, porta serial e memória "Flash" eram satisfatórios para todos dispositivos, ficando a escolha em função da memória "RAM", do custo e da familiaridade com o componente.

Tomando-se como exemplo um caso de leitura de 6 fases, de 256 pontos por ciclo e 12 ciclos, ter-se-ia o seguinte volume de dados:

$$6(\textit{fases}) * 256(\textit{ppc}) * 12(\textit{ciclos}) * 2(\textit{bytes} / \textit{ponto}) = 36864(\textit{bytes}) \qquad (4.1)$$

Nesta configuração, dentre os micro-controladores citados, mesmo o dispositivo com maior quantidade de memória "RAM" (32k) não suportaria esse volume, fazendo-se necessário a utilização de uma memória externa para o funcionamento. Dessa forma, esse quesito foi assumido como menos prioritário.

Quanto à familiaridade com o componente, o grupo de pesquisa onde este trabalho foi desenvolvido já havia utilizado micro-controladores da Atmel, embora não "ARM7", da Texas e da Analog Devices.

Por meio desta filtragem dos possíveis micro-controladores a se utilizar, optou-se pelo MSP430F169, da Texas Instruments.

Ao final da escolha dos componentes para a montagem do medidor digital, estima-se que o custo do mesmo tenho ficar na ordem de R$ 600,00.

4.4 Esquema Elétrico do Hardware

O hardware inicialmente foi dividido em alguns módulos, os quais foram testados em "protoboard" para a validação dos circuitos. Cada um tem uma função muito importante para o adequado funcionamento do medidor, por isso foram analisados separadamente, para após serem integrados.

A seguir são identificados e explicados os circuitos de alimentação, proteção do AD, tratamento do sinal, gravação e operação do micro-controlador, conversor RS232 – RS485 e o barramento RS485.

4.4.1 Alimentação

A alimentação desenvolvida é uma fonte linear, construída com um transformador, diodos, capacitores e reguladores de tensão. Pode-se reparar que após a retificação, 4 tensões são obtidas: +15, -15, +5 e +3,3 volts. Essas tensões serão utilizadas pelos outros circuitos que compõe o hardware do medidor.

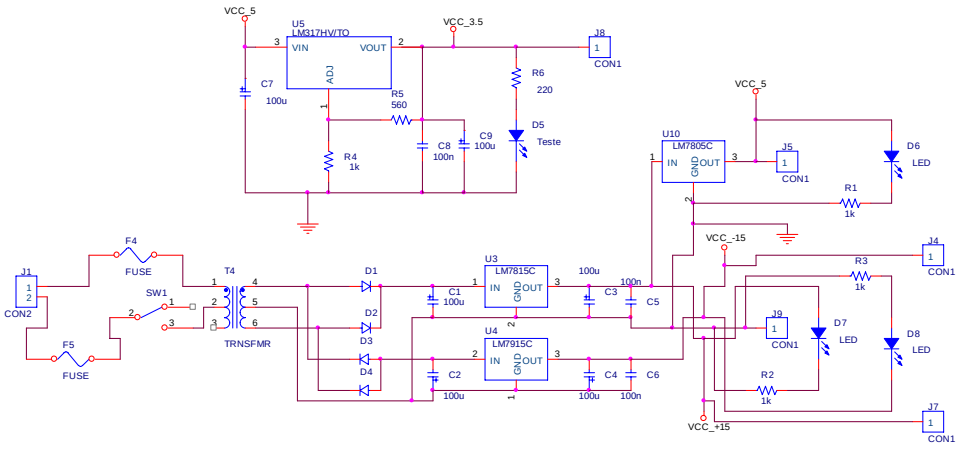


Figura 4.2 – Alimentação dos circuitos.

4.4.2 Circuito de Proteção e Sensoriamento de Tensão e Corrente

Como os sinais de interesse para o monitoramento são, em geral, de grande amplitude, como 127 ou 220 Volts para a tensão, e 5 ou 10 Ampères de corrente (exemplos fictícios, podendo estes valores serem maiores ou menores), estes devem ser tratados, de forma a poderem ser lidos pelo conversor “Analógico-Digital” do micro-controlador. Este último aceita a leitura de valores de zero até 3,3 Volts (que é a tensão de alimentação), sendo assim necessário adequar o sinal para estas faixas.

Isto é feito através da redução na amplitude, por meio da utilização de transformadores e divisores resistivos. Também, através da utilização de amplificadores operacionais, é aplicado um “off-set” aos sinais, para a eliminação da parte negativa, já que o “AD” só tem capacidade de ler valores positivos.

Mesmo depois de terem passado pelo tratamento, de forma que os sinais estivessem num nível adequado para serem lidos pelo “AD”, foi elaborado um circuito de proteção que garantisse que estes não fossem, em hipótese alguma (como surtos de tensão ou correntes), nocivos ao micro-controlador.

O circuito tem o seguinte funcionamento: em primeiro lugar, um diodo conectado entre o sinal e o terra garante que nenhum nível de tensão menor que o “terra” passe, acontecendo o mesmo para níveis de tensão maiores que o da alimentação, pela proteção do diodo entre o sinal e o “Vcc”. Após isto, o sinal passa por um amplificador operacional (para garantir potência ao sinal lido), tendo um diodo “Zenner” em paralelo, o qual limita a tensão do sinal no nível de avalanche do mesmo.

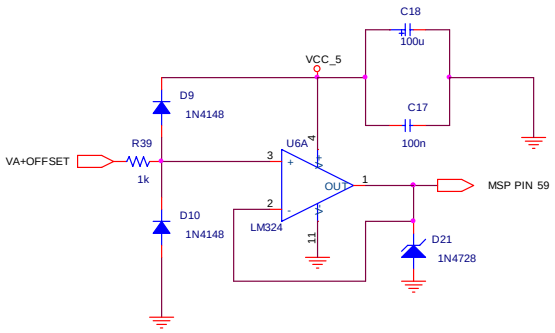


Figura 4.3 - Circuito de proteção do “AD”.

Como forma ilustrativa de mostrar o tratamento dos sinais, foram feitos dois gráficos simulados, um com o sinal de tensão lido da rede elétrica (168 Volts de pico), e o sinal lido após os ajustes necessários. Observa-se que este último foi atenuado em 188,5 vezes, e deslocado 1,7 volts para cima (off-set), atendendo aos limites do “AD” com certa folga.

Embora o sinal lido pelo “AD” esteja na faixa de 0 a 3,48V de amplitude (valores extremos), os pontos dos vetores enviados ao software gerenciador são o valor correspondente da tensão em valores de “AD”. Ou seja, o nível de tensão de “zero” Volts é representado pelo

valor “zero” do “AD”, e o nível máximo de tensão (ou corrente) pelo valor “4096”, como mostra a Tabela 4.6.

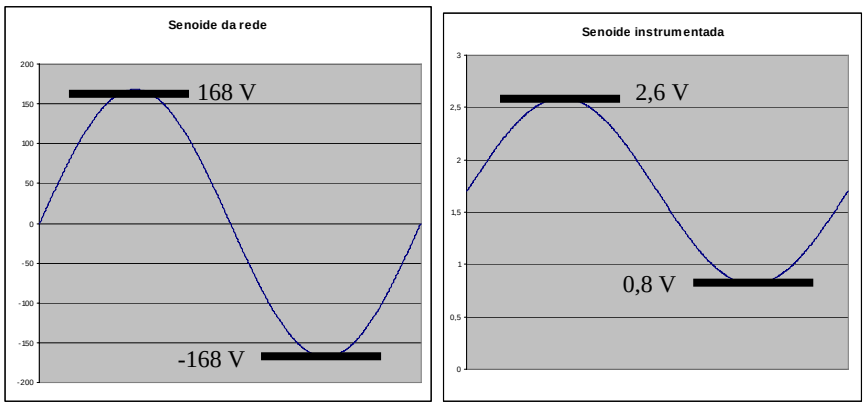


Figura 4.4 – Simulação do tratamento de sinal.

Tabela 4.6 – Relação entre nível de tensão e valor de “AD”.

Nível de tensão na rede elétrica	Valor após o tratamento de sinal	Valor correspondente de "AD"
168	2,6	3050
90	2,2	2563
0	1,7	2001
-90	1,2	1439
-168	0,8	952

4.4.3 MSP, Circuito de Programação e Transmissão de Dados

As figuras abaixo mostram os circuitos desenvolvidos para a operação do micro-controlador e da comunicação de dados.

Pode-se verificar na Figura 4.5 que os pinos 22 e 13 são, respectivamente, o “RX” e o “TX” para a programação. Já os pinos 33 e 32 são destinados à entrada e saída de dados pela porta serial. Também, deve-se reparar que existe um cristal externo, para a utilização de uma freqüência mais alta.

Na Figura 4.6, observa-se que foram utilizados dois conectores “DB9”, um para a programação e o outro para a comunicação de dados. Entretanto, foi preciso somente um componente MAX232, pois haviam entradas e saídas suficientes para tal.

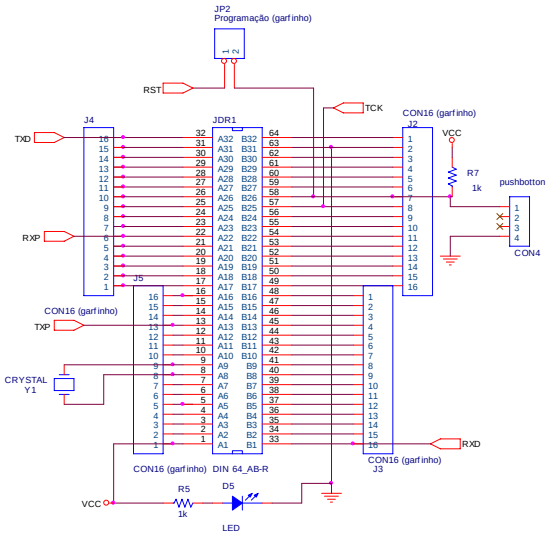


Figura 4.5 – Circuito do micro-controlador.

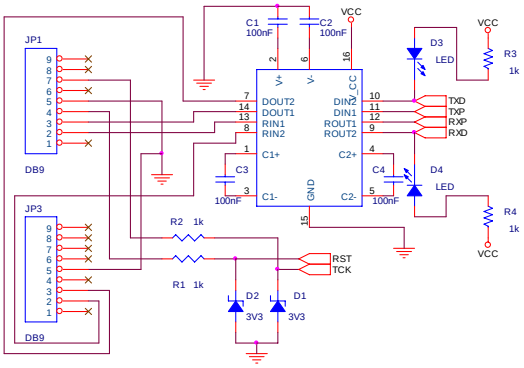


Figura 4.6 – Circuito de gravação e comunicação de dados.

4.4.4 Circuito da Memória Externa

Para a resolução da questão da memória RAM necessária para a operação do medidor, algumas soluções eram possíveis. Em primeiro lugar, se poderia ter optado pela escolha de outro micro-controlador com mais memória “RAM”. Mas, como foi visto, ainda assim não se obteria o tamanho para armazenamento de dados suficiente. Por isso, recorreu-se a uma memória “SRAM” externa que, a princípio, iria ser utilizada com mais 2 “latches” e 2

multiplexadores. Entretanto, este circuito iria demandar 24 pinos do micro-controlador, e também seria um pouco complexa a utilização.

Por isso, se resolveu utilizar uma “FPGA” para auxiliar no mapeamento da memória (endereçamento), e na entrada e saída de dados. Isto ofereceu simplificação em termos de tamanho de placa e complexidade da programação, já que se passou a utilizar somente 14 pinos do micro-controlador para essa função.

4.4.5 Circuito Conversor RS232 – RS485

A função do conversor do padrão RS232 para o RS485 é, principalmente, permitir que mais de um computador se comuniquem, de forma a organizar uma rede, onde existe um dispositivo mestre, que é o único que inicia comunicações, e escravos, que respondem aos pedidos dos mestres.

Deve-se observar no circuito que, além da “MAX232”, há um outro circuito integrado chave para a operação, que é o “MAX485”. Este último converte os sinais “TTL” em sinais para transmissão balanceada de dados (ou diferencial), o que ajuda a anular os efeitos do deslocamento do nível do terra e dos ruídos que podem aparecer nas vias de transmissão.

O barramento pelo qual os dados trafegam está esquematizado na Figura 4.7. Pode-se verificar que nesse barramento tem uma alimentação de 5 volts, que é utilizado para a alimentação dos conversores RS232 – RS485 que estão conectados aos dispositivos da rede.

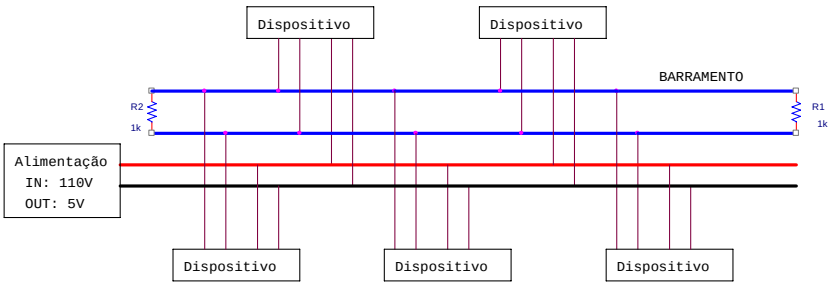


Figura 4.7 – Esquema do barramento RS485.

4.5 Placas de Circuito Impresso

Como já dito acima, o circuito do micro-controlador e da comunicação não ficaram na mesma placa da alimentação e tratamento dos sinais, assim como a memória externa, pois seria bastante complexo integrar tudo em uma mesma plataforma, e também pelo tamanho físico. Por isso, estes circuitos foram desenvolvidos em placas diferentes, e depois conectados por meio de barras de pinos.

As Figura 4.8 e Figura 4.9 mostram em fotos o hardware montado.

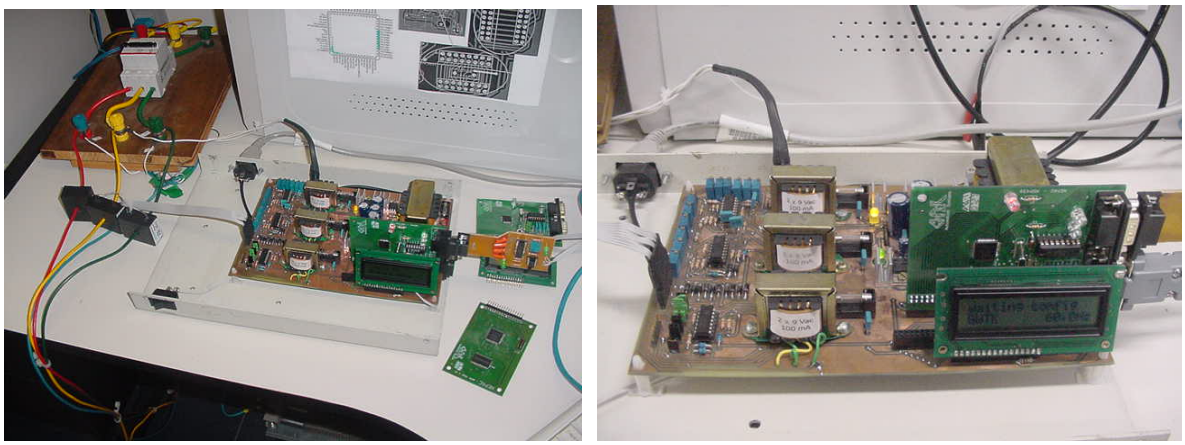


Figura 4.8 – Foto do Hardware desenvolvido.

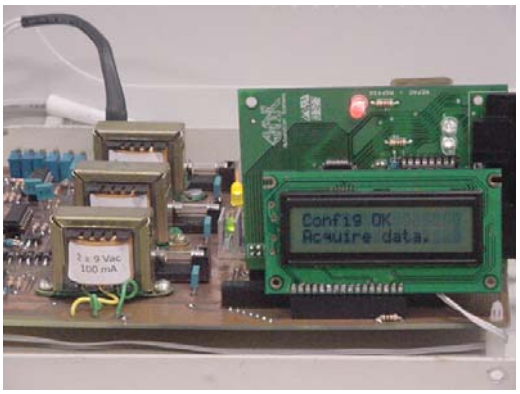


Figura 4.9 – Display mostrando que a configuração foi feita, e que os sinais estão sendo adquridos.

4.6 Estruturação e Desenvolvimento do Programa do Micro-Controlador

Após as etapas de escolha do micro-controlador, escolha dos outros componentes que integram o hardware, elaboração dos circuitos esquemáticos integrantes do hardware e a confecção das placas de circuito impresso, o desafio foi elaborar e colocar em prática o programa do micro-controlador, também conhecido como “firmware”, de forma a oferecer funcionalidade ao medidor, fazendo-lhe com que realizasse todas as tarefas propostas.

Assim como no desenvolvimento do software, três elementos são muito importantes para um bom desenvolvimento de um programa de micro-controlador, são eles: organização, modularização e documentação. Por isso, o primeiro passo foi descrever em texto como o hardware funcionaria.

> O medidor, quando ligado, deve aguardar até receber pela rede de comunicação o pacote de configuração, o qual conterá quais fases, quantos ciclos e quantos pontos por ciclo ele deve adquirir. A partir desse momento, o medidor deve começar a capturar continuamente os sinais elétricos, de acordo com a configuração estabelecida, armazenando-os em memória, e reescrevendo nesta última a cada nova leitura. Em paralelo, o micro-controlador deve monitorar a porta serial, para verificar se há algum pedido novo, e que deve ser interpretado e atendido.

Outra forma de se enxergar o funcionamento, ou os processos que o hardware deveria desempenhar, é através do diagrama de fluxo de dados. Da mesma forma como o texto descritivo acima mostrado, a Figura 4.10 identifica as tarefas que o hardware deve controlar, ou seja, o processamento que o micro-controlador deve realizar. Pode-se verificar pelo diagrama que este dispositivo deve organizar os eventos por meio de prioridades.

Sempre que chegar um pacote de comunicação pela porta serial, o micro-controlador deve interromper qualquer processamento para atender ao pedido. Ou seja, sempre que o dispositivo estiver adquirindo algum sinal, ou estiver realizando algum cálculo, o mesmo deve interromper esse processamento e atender ao evento da porta de comunicação (pedido), retornando ao processamento anterior após ter enviado a resposta.

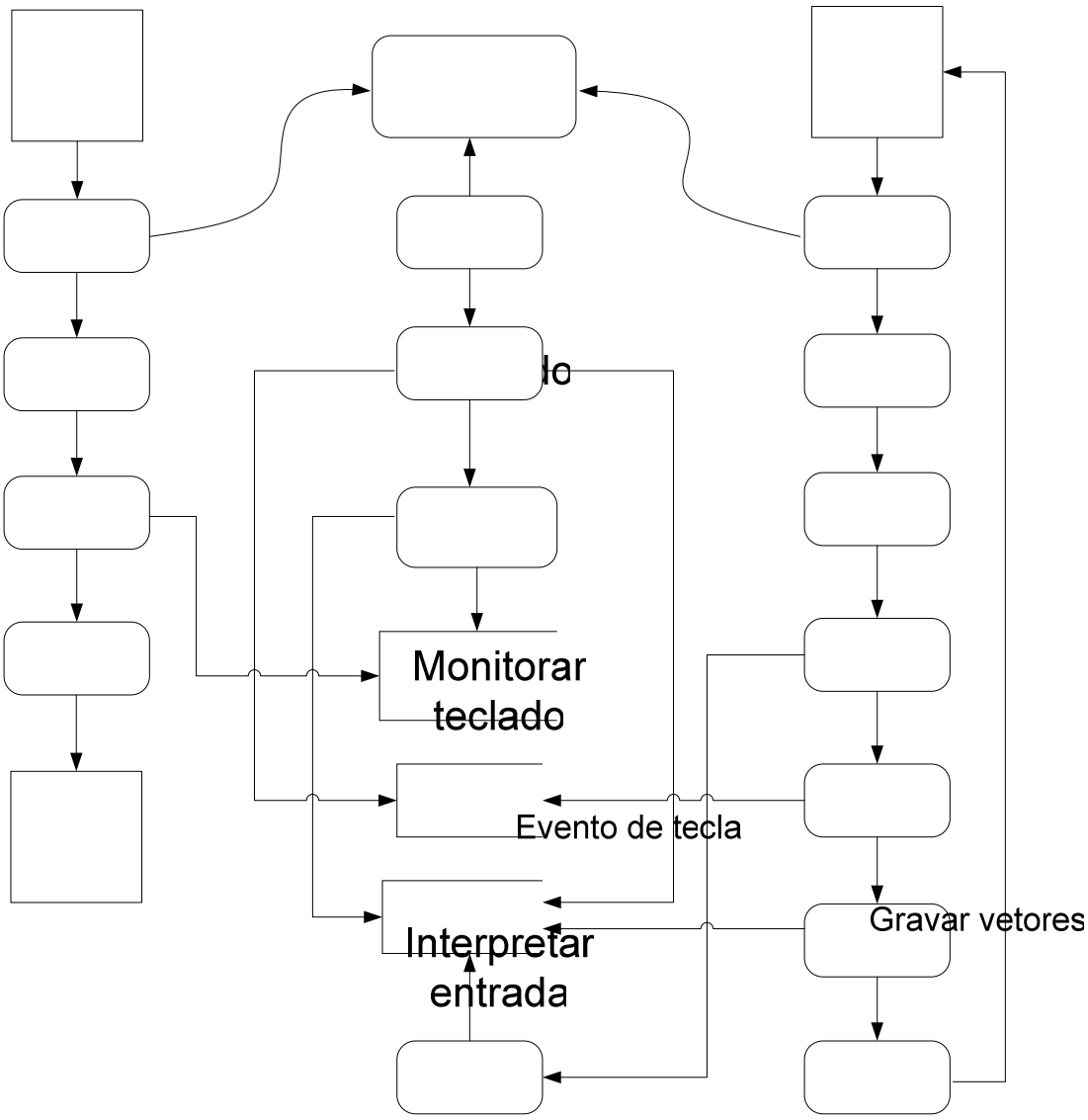


Figura 4.10 – Diagrama de fluxo de dados

A seguir, serão mostradas as partes do “firmware” que foram desenvolvidas separadamente, para após a comprovação de funcionamento individual formar o programa.

A referência [65] foi utilizada com bastante freqüência para o auxílio na utilização dos recursos do micro-controlador, por apresentar uma teoria bem fundamentada e alguns exemplos de código fonte.

4.6.1 Inicialização

A inicialização do “firmware” é uma parte do programa bastante importante, pois ali devem estar as configurações de funcionamento do micro-controlador que, se não estiverem de acordo com a necessidade do programador, podem causar problemas no algoritmo ou mal funcionamento das rotinas. Inclusive, possíveis problemas que podem parecer estar em funções, podem estar em algum erro de parametrização.

A primeira configuração é a desabilitação do “watchdog timer”, pois um “reset” automático do micro-controlador não é desejável para este aplicação.

Também foi realizada a configuração do registrador “BCSCTL1”, que é um dos responsáveis pelo controle do módulo oscilador (BCS). Foi selecionada a configuração para a operação do micro-controlador com “clock” externo, que é gerado pelo cristal de 8Mhz. Outras configurações, referentes aos registradores “IFG1” e “BCSCTL2”, também são referentes ao “clock” do dispositivo, e realizam a conferência da integridade do mesmo.

Os parâmetros da porta serial também foram ajustados: habilitação da recepção e envio de dados pela “UART 0”, no registrador “ME1”; ajuste de 8 bits de dados por byte, paridade ímpar e controle de pacotes por inatividade de linha, pelo registrador “U0CTL”; utilização do cristal externo, pelo “U0TCTL” e ajuste da taxa de transmissão através dos registradores “UBR00”, “UBR10” e “UMCTL0”. Ainda sobre a comunicação serial, a linha “U0CTL&= ~SWRST;” inicializa o funcionamento da porta.

A configuração dos “ADs” foram feitas através dos registradores “ADC12CTL0” e “ADC12CTL1”. No primeiro, os seguintes bits são ajustados: “MSC”, que significa “modo de conversão repetitiva”, ou seja, no momento da aquisição, todas portas (de “AD”) serão lidas, a uma freqüência de 200KHz; “REFON”, que significa a “ativação do gerador de tensão de referência interna; “ADC12ON”, que liga o módulo “ADC12”; e o “SHTO” que é o tempo de amostragem, configurado para amostrar o sinal durante 16 ciclos de “clock”. No segundo registrador é configurado o bit “SHP”, que ajusta a amostragem para durar o tempo definido pelo “bit” “SHTO”, do primeiro registrador, e o “CONSEQ_1”, que também significa “modo de conversão repetitiva”.

Os registradores “ADC12MCTLx” (“x” de 0 a 5) são configurados para ter referência de tensão com “AVcc” e “AVss”, ou seja, a alimentação do micro-controlador (0 a 3,48v), e ter como entrada os canais de “A0” a “A5”, respectivamente.

A variável “ACTL”, que é o registrador de controle do “timer” “A”, foi configurada para operar com o “clock” auxiliar, com contagem progressiva (de 0 até o valor determinado) e habilitação da interrupção de captura.

4.6.2 Comunicação pela Porta Serial

Como citado anteriormente, quando algum pacote chega pela porta serial, qualquer outro processamento deve ser interrompido para o atendimento deste (o protocolo ModBus, assim como a definição de “pacote” estão mostrados no ANEXO A).

Isso ocorre pela função de interrupção da UART que o micro-controlador possui. Ou seja, quando uma mensagem “ModBus” chega no pino da porta serial, o dispositivo gera automaticamente uma interrupção no processamento para atender a esse evento.

Na verdade, a interrupção é acionada sempre que chega um byte, e por isso, dentro da função da interrupção deve haver uma lógica para interpretar esse dado, pois ele pode ser de um pacote que está chegando, ou de um novo pacote. Isso é identificado pelo período de espaçamento entre um byte e outro (endereçamento por linha inativa, ou em inglês, idle line). Se um novo byte tiver chegado em menos de 10 tempos de bit (período para 10 bits serem enviados, a uma certa taxa de transmissão), este byte será do atual pacote que está sendo recebido. Entretanto, se o período de tempo for maior (representado pela letra “b” na Figura 4.11), o byte será caracterizado como um novo pacote, de forma a poder haver uma identificação dos campos de dados da mensagem.

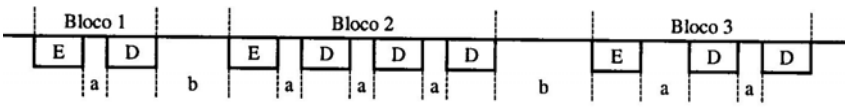


Figura 4.11 – Endereçamento por linha inativa.

O trecho de programa a frente mostra a função da interrupção da serial. Pode-se observar que logo que um byte é recebido e a interrupção é iniciada, ocorre uma comparação entre “U0RCTL” e “RXWAKE”. O primeiro citado é um registrador que contém informações da UART0, e o segundo é um campo deste registrador, que indica se o período entre os bytes foi maior ou menor que os 10 tempos de bit. Então a comparação “bit a bit” (símbolo “&”) será verdadeira se o período for maior, e falsa se for menor. Isso significa que se for verdadeira, o byte recebido é um valor de endereço, e por isso ele é comparado à constante “endereco”.

Ainda na primeira comparação, pode-se observar que o “U0RCTL” é zerado, e isso é feito para atribuir o valor “zero” ao bit 4 (“BRK”), que indica uma inatividade na linha maior que os 10 tempos de bit, ao bit 1(“RXWAKE “), que é um sinalizador do tipo de caractere recebido, e aos outros bits, que são sinalizadores de erros e outras funções.

Endereço	Nome		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0x0072	U0RCTL	Leitura	FE	PE	OE	BRK	URXEIE	URXWIE	RXWAKE	RXERR
		Escrita								
0x007A	U1RCTL	Reset	0	0	0	0	0	0	0	0

Figura 4.12 – Registrador de controle de recepção da UART.

Caso o valor do endereço recebido seja igual ao da constante “endereco”, então isso significa que o pacote enviado pela rede RS485 (todos dispositivos conectados recebem) é deste medidor, e ele deve receber o restante dos dados até o final do pacote. Por isso existe a linha “U0RCTL |= ~URXWIE”, para a habilitação da recepção dos bytes pela porta serial. Caso contrário (endereço for de outro medidor), a recepção de bytes é desabilitada, descartando qualquer dado que chegue pela UART, até que um novo pacote seja percebido.

A variável “L”, que é zerada quando um novo pacote é recebido (e o endereço é o correto), e que é incrementada a cada recepção de cada byte, é utilizada para o monitoramento do tamanho do pacote recebido, para ser utilizado na verificação da checagem de erro (“CRC”), mostrado mais adiante.

```
#pragma vector=UART0RX_VECTOR
__interrupt void usart0_rx (void)
{
 if (U0RCTL & RXWAKE)                    //verifica se o byte é o endereço
...
 L = L + 1;
}
```

Depois de corretamente recebido, o pacote deve ser analisado e interpretado, para o conhecimento da função que ele tem.

A primeira etapa é a certificação de que a mensagem não chegou com erros, devido à alguma falha na transmissão de dados. Assim, antes da interpretação do pacote, o micro-controlador deve recalcular o “CRC”, e comparar com o enviado no pacote. Se estiverem iguais, o pacote estará validado, e pronto para ser analisado.

```
unsigned char checa_CRC(void)
{
 unsigned char temp_return, temp2;
 for (temp2 = 0; temp2 < L; temp2++)
  CRC_pak[temp2] = serial_in[temp2];
 CRC = CRC16(L-2);
 CRC_HI = parte_alta(CRC);
 CRC_LO = parte_baixa(CRC);
 if ((CRC_HI == serial_in[L-1]) & (CRC_LO == serial_in[L-2]))
  temp_return = 1;
 else
  temp_return = 0;
 return(temp_return);
}
------------------------------------------------------------------------------
unsigned int CRC16(unsigned char quant_pacote)
{
 int Index;
 unsigned int temp1, temp_return = 0xFFFF;
 for (temp1 = 0; temp1 < quant_pacote; temp1++)
 {
   …
 }
 return (temp_return);
}
```

Após essa primeira etapa concluída, o pacote deve ser interpretado, de forma a poder entender o que o mestre, que enviou o pacote, está pedindo. Essa rotina é mostrada a seguir, na função “interpreta_pacote”.

```
void interpreta_pacote(void)
{
 unsigned char temp8;
 if (serial_in[1] == 0x10)
 {
  ppc = unir_alto_baixo(serial_in[7],serial_in[8]);
  n_ciclos = unir_alto_baixo(serial_in[9],serial_in[10]);
  ...
  n_fases=0;
  for(temp8=0; temp8 <= 5; temp8++)
    if(boo[temp8] == 1)
      n_fases = n_fases + 1;
  ...
  config_timer();
 }
 if (serial_in[1] == 0x03)
 {
  start_reg = unir_alto_baixo(serial_in[2],serial_in[3]);
  count_reg = unir_alto_baixo(serial_in[4],serial_in[5]);
 }
  formula_pacote(serial_in[1]);
}
```

A frente está mostrada a função “formula_pacote”, onde se observa que um valor, chamado “funcao”, deve ser passado como parâmetro. Esse valor pode ser “16” ou “03”, ou seja, função de escrita ou leitura de registradores, respectivamente.

É interessante observar que mesmo sendo uma função de escrita de registradores (“0x10”), o medidor (que é um dispositivo escravo) deve responder ao mestre da rede, ou seja, ao software gerenciador. Essa resposta é só uma confirmação de que a mensagem foi recebida e executada.

```
void formula_pacote(unsigned char funcao)
{
 unsigned char temp4;
 unsigned int tamanho_pak_out;
 serial_out[0] = endereco;
 serial_out[1] = serial_in[1];
 if (funcao == 0x10)
 {
  …
 }
 if (funcao == 0x03)
 {
  …
 }
  for (temp4 = 0; temp4 < tamanho_pak_out; temp4++)
   CRC_pak[temp4] = serial_out[temp4];
  CRC = CRC16(tamanho_pak_out);
  ...
}
```

A função abaixo mostrada é a que envia os dados pela serial. Isto é feito colocando-se os dados do vetor “serial_out” no “buffer” de saída da serial, ou seja, em “TXBUF0”, um a um. A linha de comando “while (!(IFG1 & UTXIFG0));” serve para controlar o fluxo dos dados, de forma a manter a taxa de transferência determinada.

```
void envia_pacote(unsigned char quant_pacote_out)
{
 unsigned char temp5;
 for (temp5 = 0; temp5 < quant_pacote_out; temp5++)
 {
  TXBUF0 = serial_out[temp5];
  while (!(IFG1 & UTXIFG0));
 }
 ...
}
```

Como um teste para a comunicação trifásica, foi construído um vetor representativo dos pontos de três fases de tensão, como mostrado a seguir. Deve-se reparar que na função “leituraSRAM”, a leitura é feita deste vetor fictício, e não da memória externa.

```
unsigned int seno3f [] = {1900,1998,2095,2190,2283,2371,2456,2534,2607,2673,
  2731, 2782,2824,2857,2881,2895,2900,2895,2881,2857,2824,2782,2731,2673,2607,2534,
  2456,2371,2283,2190,2095,1998,1900,1802,1705,1610,1517,1429,1344,1266,1193,
  1127,1069,1018,976,943,919,905,900,905,919,943,976,1018,1069,1127,1193,1266,
  1344,1429,1517,1610,1705,1802,2776,2725,2665,2599,2525,2446,2361,2272,2179,
  2084,1986,1888,1790,1693,1598,1506,1418,1335,1257,1185,1120,1062,1013,972,
  940,917,904,900,906,922,947,981,1024,1075,1135,1201,1275,1354,1439,1528,1621,
  1716,1814,1912,2010,2107,2202,2294,2382,2465,2543,2615,2680,2738,2787,2828,
  2860,2883,2896,2900,2894,2878,2853,2819,1056,1007,967,936,915,903,900,907,924,
  950,985,1029,1082,1142,1210,1284,1364,1450,1539,1632,1728,1825,1924,2021,2118,
  2213,2304,2392,2475,2552,2624,2688,2744,2793,2833,2864,2885,2897,2900,2893,
  2876,2850,2815,2771,2718,2658,2590,2516,2436,2350,2261,2168,2072,1975,1876,
  1779,1682,1587,1496,1408,1325,1248,1176,1112};
-----------------------------------------------------------------------------------------------------
void leituraSRAM (unsigned int start_reg,unsigned int count_reg)
{
for(x=0;x <= (2*count_reg-1);x++)
 {
  serial_out[x+3]=parte_alta(seno3f[start_reg+(x/2)]);
  serial_out[x+4]=parte_baixa(seno3f[start_reg+(x/2)]);
  x=x+1;
 }
}
```

A Figura 4.13 mostra um teste prático realizado, para a validação da comunicação. Como eram três vetores de 64 pontos por ciclo (1 ciclo cada), foram feitos 3 pedidos, da seguinte forma: 1° pedido, do registrador 0 ao 63; 2° pedido, do registrador 64 ao 127; 3° pedido, do registrador 128 ao 191.

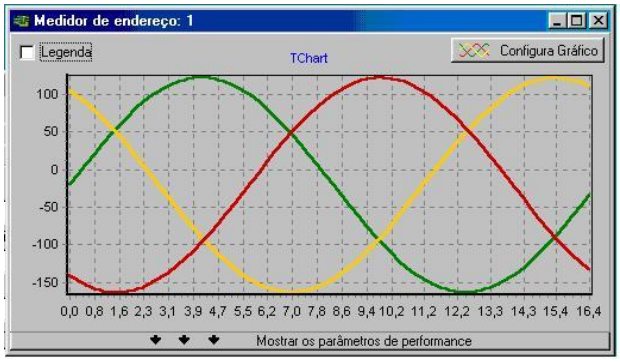


Figura 4.13 – Teste prático da comunicação serial.

4.6.3 Principal

Para a explicação do funcionamento da estrutura “principal” do programa, deve-se antes explanar a estratégia de funcionamento. Como já citado no capítulo de software, o medidor deve ser configurado por meio da comunicação serial, podendo, a partir disso, começar a adquirir os sinais elétricos pelos canais de “AD”. Mas antes, o medidor deve calcular qual a freqüência da rede, de forma a enviá-la junto na resposta à configuração dos parâmetros. Embora isso ainda não esteja implementado (o envio da freqüência), deverá ser feito para flexibilizar o sistema perante freqüências diferentes dos 60Hz, já que esta é utilizada tanto para a geração dos gráficos como dos cálculos dos parâmetros de performance no software.

Então, seguindo o código fonte, observa-se que o primeiro comando é a chamada da rotina “calcfreq()”, que é uma função que calcula e retorna a freqüência da rede. Em seguida, (após ter mostrado a freqüência no display), o programa entra em um laço “do-while”, aguardando o recebimento do pacote de configuração. Neste ponto é interessante lembrar que dentro da configuração dos parâmetros tem a chamada para a função “config_timer”, que faz os ajustes do “timer A”. Após configurado (número de ciclos, pontos por ciclo, fases e “timer”), o programa do micro-controlador é direcionado outro laço (um “for” infinito), onde fica constantemente chamando a função “carregaSRAM” (explicada no próximo índice), e monitorando a chegada de algum pacote de comunicação pela porta serial.

Essa última tarefa citada é feita pela verificação de duas variáveis: a “Flag_serial”, que sinaliza a chegada de um pacote, e do bit “BRK” do registrador “U0RCTL”, que indica a

detecção de término do pacote. Quando as duas são verdadeiro, o programa se direciona para as rotinas de comunicação serial, formulando o pacote de resposta e o enviando.

```
calcfreq();                          //calcula perído de sinal amostrado no canal ad0
fr=10000000/t1;
GotoLCD(2,10);  PrintNum(2,fr);  GotoLCD(2,15);  PrintLCD("Hz");
do
{
 if(Flag_serial == 1)                //chegou um pacote pela serial ?
   ...
   {
    interpreta_pacote();              //interpreta a informação do pacote e responde
    ...
   }
 GotoLCD(1,1);
 PrintLCD("waiting config  ");
}
while((ppc==0)&&(n_ciclos==0));
ClearLCD();
ntp =ppc*n_ciclos;
for (;;)
{
 if(Flag_serial == 1)                //chegou um pacote pela serial ?
  ...
   {
    interpreta_pacote();              //interpreta a informação do pacote e responde
    ...
   }
 carregaSRAM();
}
```

4.6.4 Aquisição dos Sinais e Leitura/Escrita na Memória Externa

A aquisição dos sinais tem como gatilho para a leitura do ponto instantâneo a ser lido a finalização da contagem do “timer A”, cujo valor a contar deve ser exatamente o período entre um ponto e outro.

```
void carregaSRAM(void)                // carrega vetor na freq do nro de pontos por ciclo
{
 P4DIR |= 0xff;                       //configura porta para escrita
 do
 {   // Nothing   }
 while((np <= (ntp-1))&&(Flag_serial == 0));
 np=0;
 cz=0;
}
```

Pelo fato de esse processo ser complexo, uma série de detalhes são precisos para coordenar o correto funcionamento.

Primeiramente pode-se observar que se a variável “n_ped_feitos” for diferente de zero, o que significa que existem pedidos pendentes ao medidor, o mesmo não adquire dado nenhum, e sai da interrupção. Mas se não houver nenhuma comunicação pendente, a leitura pelo conversor “AD” é ativada, através da linha de comando “ADC12CTL0 |= ADC12SC;”.

Os valores capturados nos “ADs” são então salvos em variáveis (de “ad0” a “ad5”), mas só serão armazenados em memória se a variável “cz” estiver em 1.

Após detectado o cruzamento por zero da tensão da fase “A” (cz = 1), essa verificação não mais ocorrerá, e o micro-controlador irá adquirir os pontos até completar o vetor. Quando todos pontos tiverem sido capturados, o contador de pontos adquiridos é zerado (np = 0) e o processo de leitura dos sinais recomeça.

Durante a aquisição, a cada ponto o programa confere quais fases deve estar adquirindo, e isso é feito através do vetor “boo[x]”, onde as posições de zero a cinco representam as fases “TA”, “TB”, “TC, “CA”, CB” e “CC”, respectivamente. Então, se o valor “boo[2]” for “1”, significa que a tensão da fase “C” deve ser capturada (armazenada em memória), por exemplo.

E quanto à posição da memória onde esse valor deve ser armazenado, isto é coordenado pela expressão “np+z*ntp”, que significa o seguinte: “np” é o número do ponto (no range de zero até “pontos por ciclo”) em que a aquisição está; o “z” é uma variável local que é incrementada a cada ponto armazenado; e o “ntp”, que é o “número total de pontos” (número de ciclos vezes o número de pontos por ciclo), é multiplicado pelo “z”, deslocando o ponto para o setor correto, dentro do vetor. Abaixo está mostrado um exemplo disto, com a configuração de 64 pontos por ciclo, e um ciclo. Mais exemplos com mais ciclos poderiam ser dados, mas dessa forma fica mais simples de se entender. E deve-se atentar para que no segundo exemplo o “z” máximo é “3”, pois são 4 fases “ativas”.

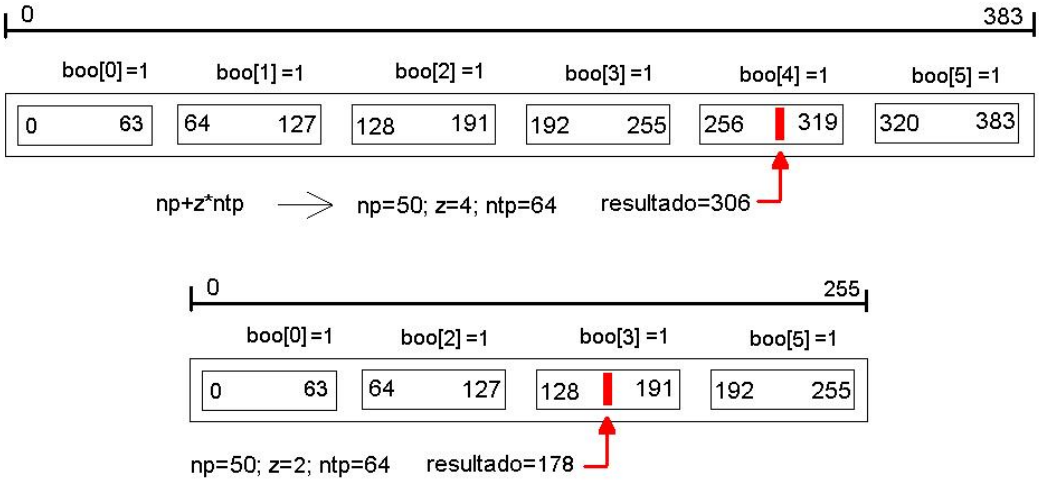


Figura 4.14 – Exemplos de armazenamento de um ponto.

```
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
 if(n_ped_feitos == 0)
 {
  ADC12CTL0 |= ADC12SC;            // Sampling open
  ad0= ADC12MEM0;
  ad1= ADC12MEM1;                  // Move results, IFG is cleared
  ...
  e=d;  d=c;  c=b;  b=a;  a=ad0;     //filtro passa-baixa p/ detecção de borda de subida
  if(cz==0)
   if(((ad0>=2000)&&(ad0<=2100))&&(a > e))   //passagem por zero em borda de subida
   {
    cz=1;  e=4095;   np=0;
   }
  if(cz==1)
  {
   z=0;
   if(boo[0]==1)
   {
    enderecaSRAM(0,np+z*ntp);
    escritaSRAM(parte_alta(ad0),parte_baixa(ad0));          z=z+1;          //Tensão A
   }
  ...
   np=np+1;
```

Observando-se a forma como o ponto é escrito na memória, duas funções novas são percebidas: “enderecaSRAM(A17..A16,A15..A0)” e “escritaSRAM(parte_alta,parte_baixa)”. A primeira é a informação do endereço no qual o dado deve ser armazenado, e a segunda a escrita na memória. Mais detalhes sobre essas duas funções podem ser encontradas no APÊNDICE D, assim como a “lerSRAM(posição_ponto)”, mostrada no quadro abaixo, que lê um dado da memória para enviar pela comunicação serial.

Quanto a esse recurso da memória “SRAM” e da “FPGA”, o importante de se entender é que sempre antes de ler ou escrever um dado na memória (primeiro a parte alta do valor de 16 bits, depois a parte baixa), deve-se informar o endereço à “FPGA”, também pelas partes alta e baixa do endereço.

```
void leituraSRAM (unsigned int start_reg,unsigned int count_reg)
{
 unsigned int x;
 for(x=0;x <= (2*count_reg-1);x++)
 {
  enderecaSRAM (0,start_reg+x/2);
  lerSRAM(x);
  x=x+1;
 }
}
```

4.7 Resultados

A partir da escolha certa do micro-controlador a ser utilizado, do planejamento do funcionamento do medidor e do projeto dos algoritmos da “firmware”, pôde-se atingir um nível de desenvolvimento do sistema bastante interessante.

Como é possível observar na Figura 4.15, conseguiu-se realizar a leitura trifásica dos sinais de tensão e corrente, e transferir os dados ao software, de modo a que esse apresentasse os gráficos e os cálculos dos parâmetros de performance. Entretanto, como se pode observar em alguns pontos do gráfico, existe um pequeno ruído nos sinais lidos, que deve ser sanado em implementações futuras.

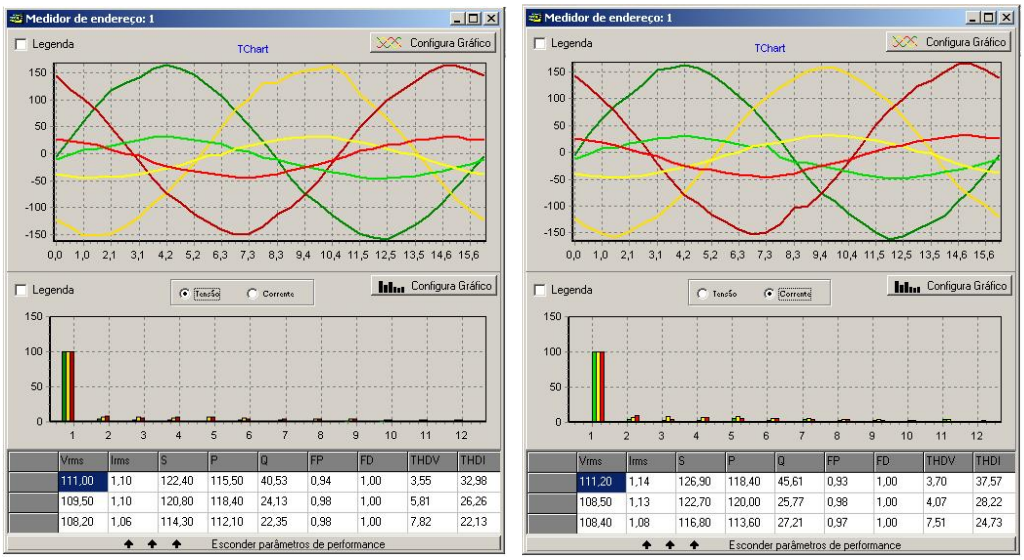


Figura 4.15 – Aquisição trifásica de corrente e tensão, de uma carga resistiva.

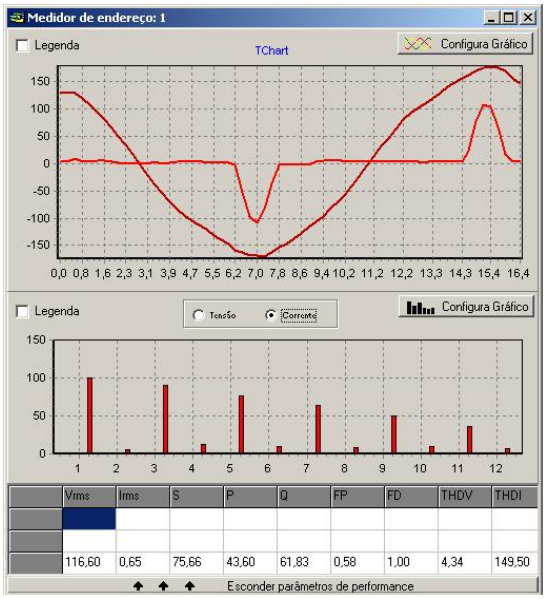


Figura 4.16 – Tensão e corrente de um monitor de computador.

Uma observação bastante importante de ser feita é o benefício que a inclusão da memória “SRAM” externa trouxe ao dispositivo. Ela possibilitou que número de ciclos da ordem de 10, 15 ou mais fossem possíveis. Claro que em casos de muitos ciclos, muitas fases e/ou um número elevado de “pontos por ciclo” a comunicação serial apresentou limitações, devido ao tempo necessário para a transferência dos dados. As Figura 4.17 e Figura 4.18 mostram algumas aquisições de mais de um ciclo.

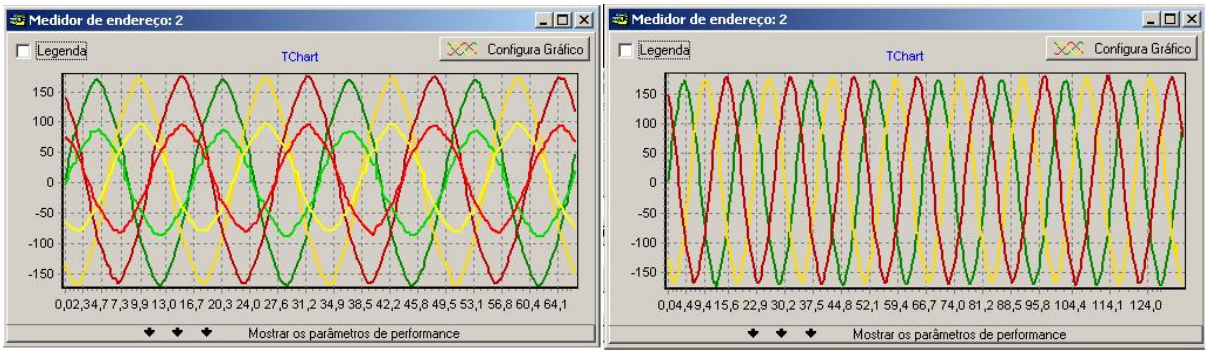


Figura 4.17 – Aquisição de alguns ciclos de uma carga resistiva.

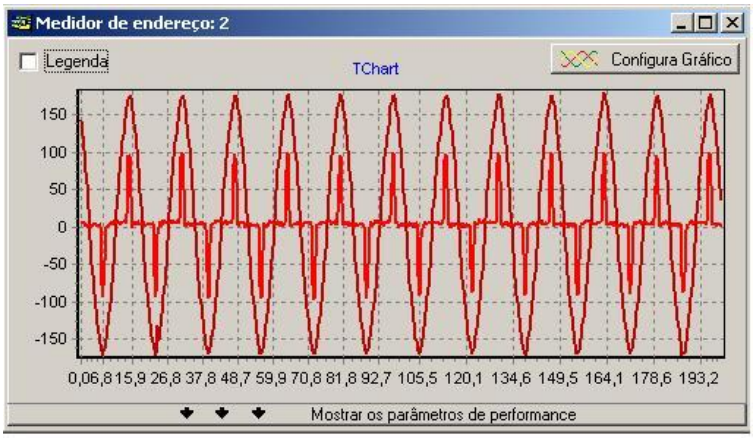


Figura 4.18 – Doze ciclos de tensão e corrente de um monitor de computador.

Quanto ao custo do hardware desenvolvido, estima-se que este tenha ficado na ordem de R$1100,00, incluindo-se neste cálculo todos os componentes eletrônicos, a confecção das placas, o serviço terceirizado de montagem, a caixa externa, o lucro e os impostos incidentes.

Considera-se este valor tenha ficado dentro do planejado, pois está só um pouco acima do medidor simples pesquisado, e bastante abaixo dos medidores mais avançados.

4.8 Conclusões

O medidor digital implementado demonstrou, após alguns testes práticos, algumas qualidades e algumas limitações, mostradas a seguir.

Quanto ao micro-controlador, o mesmo correspondeu às necessidades para o projeto, oferecendo uma resolução de “AD” de 12 bits (enquadrando-se na resolução 505 da Aneel) e um custo adequado. A capacidade de processamento, apesar de em algumas condições ficar próximo do limite, proporcionou o funcionamento planejado.

A estruturação do programa do micro-controlador também se mostrou bem planejada, pelo fato de ter proporcionado, de forma fácil, manutenções e modificações durante a fase de desenvolvimento e testes.

Como limitações, podem-se citar os ruídos e a taxa de transmissão da comunicação serial. O primeiro citado é bastante indesejado, pois ele pode significar algum desvio grosseiro da forma de onda senoidal, quando na verdade isso não existe.

A taxa de transmissão, apesar de não representar nenhum erro, é uma limitação do volume de dados a ser transferido do medidor ao software, o que também pode refletir na periodicidade mínima das consultas.

Por isso, como implementação futura, pode-se citar em primeiro lugar a troca da rede serial pela rede “Ethernet”, para comunicação de dados. E como forma de melhorar o desempenho do equipamento, se pode acrescentar um co-processador para trabalhar em paralelo com o principal, de forma a obter-se funções de análise de dados, podendo inclusive ter recursos de inteligência artificial para a busca de distúrbios.

Outra tendência interessante de ser avaliada é a escrita dos dados diretamente do medidor no banco de dados, eliminando-se a necessidade de um software intermediário para a execução dessa tarefa.

5 DESENVOLVIMENTO DO BANCO DE DADOS

A informação, tanto do ponto de vista popular quanto do ponto de vista científico envolve um processo de redução de incerteza. Na linguagem diária, a idéia de informação está ligada à de novidade e utilidade, pois informação é o conhecimento disponível para uso imediato e que permite orientar a ação, ao reduzir a margem de incerteza que acerca as decisões cotidianas. Na sociedade moderna, a importância da disponibilidade da informação ampla e variada cresce proporcionalmente ao aumento da complexidade da própria sociedade.

Informações são constituídas por um conjunto de dados, sendo estes definidos como um conjunto de registros a respeito de um evento. Quando um conjunto de dados possui um significado, ele se torna uma informação, e no momento que esta é transmitida a alguém, ela está sendo compartilhada com essa pessoa. Mas para que haja comunicação, é necessário que o destinatário da informação a receba e compreenda. A informação transmitida, mas não recebida, não foi comunicada. Comunicar significa tornar comum a uma ou mais pessoas uma determinada informação, oriundas de um sistema de interesse.

Atualmente, os banco de dados são essenciais para todos os ramos de negócios. Eles são usados para manter registros internos, apresentar dados aos consumidores na rede mundial de computadores (WWW) e fornecer suporte a muitos outros processos comerciais e industriais [25].

O poder dos banco de dados vem de um corpo de conhecimento e tecnologia que se desenvolveu ao longo de várias décadas e é encarnado em um tipo de software especializado chamado “sistema de gerenciamento de banco de dados” (SGDB). Um “SGDB” é uma

ferramenta poderosa para criar e gerenciar grandes quantidades de dados de forma eficiente e permitir que esses dados persistam durante longos espaços de tempo com segurança. Esses sistemas estão entre os tipos mais complexos de software disponíveis.

5.1 Escolha do banco de dados

Existem diversos softwares “SGDB” disponíveis no mercado, alguns até disponibilizados gratuitamente para “download” na internet. A escolha do “SGDB” deve-se ao fato de oferecer recursos aos usuários tais como:

- Armazenamento persistente. Do mesmo modo que um sistema de arquivos, um “SGDB” admite o armazenamento de quantidades muito grandes de dados que existem independentemente de quaisquer processos que estejam os utilizando, porém, o “SGDB” vai muito além do sistema de arquivos. Este proporciona flexibilidade, por meio de estruturas que permitem o acesso eficiente a quantidades muito grandes de dados.

- Interface de programação. Um “SGDB” permite ao usuário acessar e modificar dados através de uma linguagem de consulta poderosa. Mais uma vez, a vantagem deste sobre um sistema de arquivos é a flexibilidade para manipular dados armazenados de forma muito mais complexa que a simples leitura e gravação de arquivos.

- Gerenciamento de transações. Um “SGDB” admite o acesso concorrente a dados, isto é, o acesso simultâneo por muitos processos distintos (chamados de transações) ao mesmo tempo. Para evitar algumas das conseqüências indesejáveis do acesso simultâneo, o “SGDB” admite isolamento (garantindo que as transações sejam executadas uma de cada vez) e a atomicidade (execução completa ou não execução da função). Um “SGDB” também admite a resiliência, que é a capacidade de se recuperar de muitos tipos de falhas e erros.

A escolha do “SGDB” utilizado neste trabalho baseou-se nas informações acima citadas, pois o sistema em questão deveria ser capaz de manipular com eficiência uma grande quantidade de dados e de gerenciar o acesso de múltiplos usuários.

Assim, foi feito um levantamento de quais banco de dados se adequariam aos parâmetros desejados. A Tabela 5.1 mostra uma pequena comparação das características de 4 sistemas possíveis.

Pode-se observar que o “PostgreSQL” foi o único banco de dados que satisfez todas as características levantadas, sendo que dentre elas estavam as mais procuradas: integridade referencial, transações concorrentes, “triggers” e isenção de custo para a utilização.

Mais especificamente, o “PostgreSQL” foi o escolhido por permitir suporte a múltiplas transações “online” concorrentes entre usuários, cópias de segurança de um ou todos banco de dados sem a necessidade de desconectar os usuários e a utilização de gatilhos, para a captação instantânea de algum valor de interesse. Ainda, este banco de dados oferece condições para a utilização de procedimentos armazenados, do conceito “objeto-relacional”, de funções e tipos de dados definidos pelo usuário, entre outros [59].

Outros bancos de dados também foram analisados, mas descartados da escolha. O banco de dados “Oracle”, por exemplo, é um dispositivo bastante robusto e que satisfaria perfeitamente nas questões técnicas, entretanto é pago, da mesma forma como o “Interbase”, da Borland. Outros bancos, como o “MySQL”, o “SQL Server” e o “Microsoft Access” também foram rejeitados por não serem gratuitos, não possuírem gatilhos, não darem suporte a uma aplicação multi-usuários ou outras características.

A estruturação, construção e utilização de um banco de dados é uma tarefa relativamente complexa, a qual requer estudos para o conhecimento das técnicas e ferramentas a serem utilizadas. Por isso, será mostrado um pouco de teoria a respeito do assunto.

Tabela 5.1 – Comparação de alguns “SGBD”.

Características	MySQL	PostgreSQL	Interbase	Adabas
Freeware	✓	✓	✓	Versão D personal
Open source	Linux, NT	Linux, NT, SunSl, WinME, 2000, XP, MacOS	Linux, NT, Win98,NT	Unix, IBM AIX, SNI Sinix, Sun Solaris,
Stored procedures Triggers	✗	✓	✓	✓
Transações concorrentes	✗	✓	✗	✓
Sem limite de usuários	✓	✓	✓	3 sessões concorrentes
Integridade referencial	✗	✓	✓	✓
Transações	✗	✓	✓	✗
ODBC free	✓	✓	✓	✗
Estabilidade	✓	✓	✓	✓
Disponibilidade (backup HOT)	✗	✓	✓	✓
Interface amigável	*	***	***	*
Ferramentas de apoio ao desenvolvimento	Exige ferramentas adicionais			
Extrator de relatórios	Exige ferramentas adicionais			

FONTE: NICHOLS, Giselli. “Conquistando Novas Fronteiras PostgreSQL”.

5.2 Modelo de Banco de Dados

Um modelo de dados é uma definição abstrata, autônoma e lógica dos objetos, operadores e outros elementos que, juntos, constituem a “máquina abstrata” com a qual os usuários interagem. Os objetos permitem a modelagem da estrutura de dados, e os operadores, o comportamento dos mesmos [25].

O projeto de um banco de dados ocorre usualmente em três etapas. A primeira etapa, a modelagem conceitual, procura capturar formalmente os requisitos de informação de um banco de dados. A segunda etapa, o projeto lógico, objetiva definir, em nível de “SGDB”, as estruturas de dados que irão implementar os requisitos identificados na modelagem conceitual. A terceira etapa, o projeto físico, que define os parâmetros de acesso ao BD, procurando otimizar a performance do sistema como um todo.

Este trabalho, na modelagem conceitual, utiliza a abordagem entidade-relacionamento (“ER”), considerada um padrão de modelagem de dados. Quanto ao projeto lógico, será

coberto o projeto propriamente dito (transformação de modelos “ER” em modelos relacionais) e, finalmente, no projeto físico será modelado o banco de dados utilizando o “PostgreSQL”, auxiliado pelo sistema gerenciador de banco de dados “PgAdmin”.

Neste ponto do trabalho, é importante que seja esclarecido que o banco de dados e a interface de acesso remoto a ele (página da internet) não foram desenvolvidos na íntegra pelo autor deste trabalho. Essa tarefa está vinculada ao trabalho de conclusão de curso de um integrante do grupo de pesquisa, e é encontrado na referência [51].

5.2.1 Modelagem conceitual

Um modelo conceitual é uma descrição do banco de dados de forma independente da implementação em um “SGDB”. O modelo conceitual registra quais dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados em nível de sistema gerenciador.

A técnica mais difundida é a abordagem entidade-relacionamento (ER). Nesta técnica, um modelo conceitual é usualmente representado através de um diagrama, chamado “Diagrama Entidade Relacionamento” (DER) [32].

5.2.1.1 Identificação das Entidades

A entidade é um conceito fundamental da abordagem “ER”, significando um conjunto de objetos da realidade modelada, sobre os quais se deseja manter informações no banco de dados. Uma entidade pode representar tanto objetos concretos da realidade (substantivos, como “os produtos”, “as vendas” ou “as medidas”), como também objetos abstratos (“tipo de informação”, por exemplo). Em um “DER”, uma entidade é representada através de um retângulo que contém o nome da entidade.

5.2.1.2 Atributos

Conforme mencionado no início da seção, o modelo “ER” permite a especificação de propriedades de entidades. Uma propriedade é participar de um relacionamento. Outra é ter um atributo. O conceito de atributo serve para a associação de informações a ocorrências de entidades ou de relacionamentos.

Na prática, atributos não são representados graficamente para não sobrecarregar os diagramas, já que muitas vezes entidades possuem um grande número de atributos. Prefere-se usar a representação textual que aparece separadamente do diagrama “ER”. No caso de ser usado um software para construção destes modelos, o próprio software encarrega-se do armazenamento da lista de atributos de cada entidade em um dicionário de dados.

A partir dessa pequena introdução ao diagrama “ER”, pode-se mostrar algumas entidades e atributos desenvolvidos. Logo abaixo, são mostradas duas entidades: a “usuário”, que servirá para garantir a privacidade dos dados, por meio de controle das pessoas que tem acesso, e a “administrador”, para o registro dos administradores do banco, que poderão configurar parâmetros como “pontos de medição”, “equipamentos” e outros.

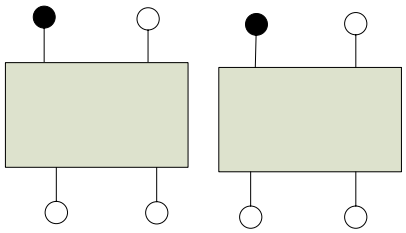


Figura 5.1 – Entidades “usuário” e “administrador”.

Pode-se observar que em ambas as entidades, “usuário” e “administrador”, existe um atributo como chave primária que garante a unicidade na ocorrência das entidades.

Além destas, também é necessário uma entidade para o armazenamento de dados a respeito das empresas, como nome, endereço, cidade, telefone, e-mail para contato, etc.

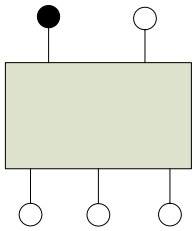


Figura 5.2 – Entidade “cadastro”.

Outra entidade de grande importância é a “ponto”, devido ao sistema de monitoramento dos usuários poder possuir muitos pontos de medição, tantos quantos forem necessários. Observa-se na Figura 5.3 que essa entidade possui os atributos necessários para identificar a localização dos pontos de medição.

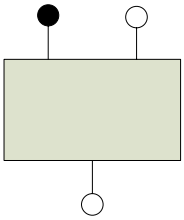


Figura 5.3 – Entidade “ponto”.

Também, duas entidades bastante relevantes, relativas às medições das informações elétricas, são a “inst” e a “hist”. A primeira possui os atributos necessários para armazenar os dados relativos aos vetores representativos da tensão e corrente, enquanto a segunda serve para o armazenamento dos parâmetros de performance, associados a alguma medição, em algum ponto e horário.

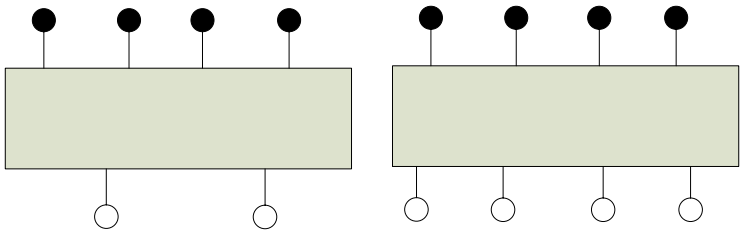


Figura 5.4 – Entidades armazenadores de informações elétricas.

Devido a cada medição estar associada a um “ponto”, e neste ponto ter um medidor de algum determinado modelo, então foi criada uma entidade chamada “modelo”, que contém algumas informações técnicas do aparelho em si.

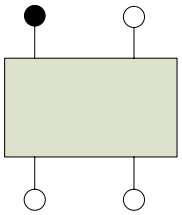


Figura 5.5 – Entidade “modelo”.

5.2.1.3 Relacionamentos

cod_

Além de especificar as ocorrências de entidades sobre as quais se deseja manter informações, o modelo “ER” deve permitir a especificação das propriedades destas ocorrências que serão armazenadas no banco de dados.

Lembrando que, em um relacionamento binário “R” entre duas entidades “A” e “B”, a cardinalidade máxima de “A” em “R” indica quantas ocorrências de “B” podem estar associadas a cada ocorrência de “A” [32].

senha_a

Uma das propriedades sobre as quais pode ser desejável manter informações é a associação entre as ocorrências de entidades. Exemplificando, pode ser desejável saber quais pessoas estão associadas a uma organização.

E da mesma forma como foi feito com as entidades, quando se refere à associações particulares dentro de um conjunto, deve-se referenciar à ocorrência ou instância de relacionamentos.

Ainda complementando a relação entre um relacionamento e a entidade, pode-se dizer que este último tem o papel de definir que função uma “instância da entidade” cumpre dentro de uma “instância do relacionamento”.

Em um “DER”, um relacionamento é representado através de um losango, ligado por linhas aos retângulos, representativos das entidades que participam do relacionamento. A Figura 5.6 mostra o relacionamento entre a entidade “cadastro” e a entidade “ponto”.

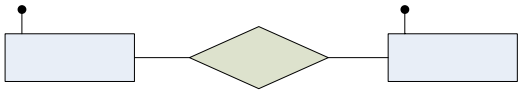


Figura 5.6 – Relacionamento entre a empresa e os pontos de medição.

Estas duas entidades estão relacionadas através da quantidade de pontos encontrados na empresa. Uma empresa possui uma quantidade “n” de pontos de medição, porém, um ponto somente existe em uma empresa.

Um ponto de medição é identificado pela empresa através do código do ponto (cod_ponto), o qual o distingue este dos demais pontos de uma mesma empresa. Esta entidade realiza inúmeras medições, que são armazenadas como informações pelas entidades “instantaneo” e “historico”. Um mesmo ponto pode realizar estes três tipos de medidas.

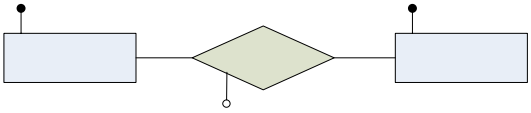


Figura 5.7 – Relação entre um “ponto” e “grandezas”.

Analisando o atributo “tipo”, da Figura 5.7, pode-se representá-lo de maneira singular, como mostrado nas figuras abaixo.

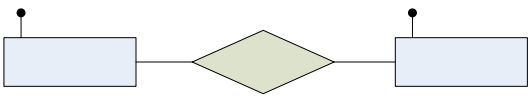


Figura 5.8 - Relacionamento entre “ponto de medição” e “medida de histórico”.

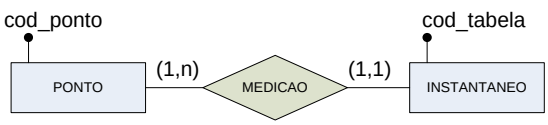


Figura 5.9 - Relacionamento entre ponto de medição e medida Instantânea.

Como já visto, as “medições” são realizadas por “equipamentos”. E estes, são armazenados na entidade “equipamento”, que serve para identificar quais aparelhos uma determinada empresa possui.

O relacionamento mostrado a seguir é um caso de relacionamento ternário (em lugar do binário), pois a entidade “equipamento” pode aparecer em mais de um “ponto de medição”, assim como um ponto pode ter mais que um “equipamento”, e ainda, um “equipamento” pode estar cadastrado, mas não estar em “ponto” nenhum, e por isso é que o relacionamento ternário se faz necessário. Assim, criou-se uma entidade que faz a distribuição dos modelos de equipamentos nos pontos de medição.

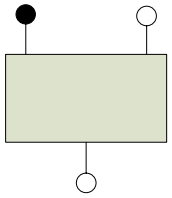


Figura 5.10 – Entidade “equipamento”, identificada em nível de implementação.

No caso de relacionamentos de grau maior que dois, o conceito de cardinalidade de relacionamento é uma extensão não trivial do conceito de cardinalidade em relacionamentos binários. No caso de um relacionamento ternário, a cardinalidade refere-se a pares de entidades. Em um relacionamento “R” entre entidades “A”, “B” e “C” a cardinalidade máxima de “A” e “B” dentro de “R” indica quantas ocorrências de “C” podem estar associadas a um par de ocorrências de “A” e “B” [32].

Associando-se esta explicação teórica ao relacionamento entre “ponto”, “modelo” e “equipamento”, mostrado na Figura 5.11, pode-se verificar que cada par de ocorrência (“ponto” e “modelo”) está associado a no máximo um “equipamento”.

Ainda, pode-se verificar que para um par “ponto – equipamento”, muitos modelos podem estar associados, assim como muitos pontos podem estar vinculados ao par “equipamento – modelo”.

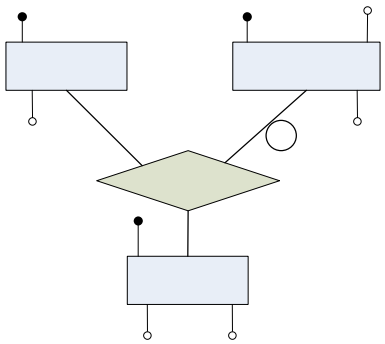


Figura 5.11 - Relacionamento “ponto” e “modelo”, controlada pela entidade “equipamento”.

5.2.1.4 Generalização/Especialização

Este conceito é importante para atribuir propriedades particulares a um subconjunto das ocorrências (especializadas) de uma entidade genérica. No “DER”, o símbolo para representar generalização/especialização é um triângulo, como mostrado na Figura 5.12, que expressa que a entidade “ponto” é dividida em dois subconjuntos, as entidades “inst” e “hist” (representado por “instantâneo” e “histórico”), cada uma com as suas próprias propriedades.

Associada ao conceito de generalização/especialização existe a idéia de herança de propriedades. Herdar propriedades significa que cada ocorrência da entidade especializada possui, além de suas próprias propriedades (atributos, relacionamentos e generalizações/especializações), também as propriedades da ocorrência da entidade genérica correspondente. Ou seja, a entidade “inst” possui os atributos “tensão” e “corrente”, e mais “cód_ponto” e “localidade”, ocorrendo o mesmo com a entidade “hist”.

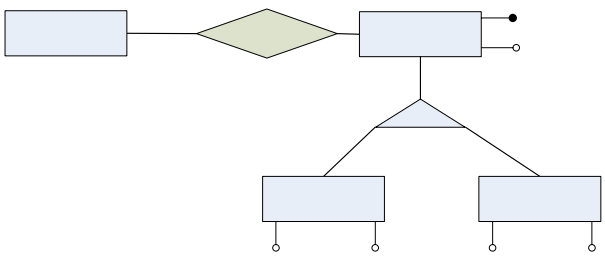


Figura 5.12 – Generalização entre as entidade “ponto”, “inst” e “hist”.

Resumindo, o diagrama expressa que todo ponto de medidas instantâneas tem como atributos “código”, “localidade”, “tensão” e “corrente”, e é identificada pelo “código”, e esta obrigatoriamente relacionada a exatamente um “engenheiro”. Da mesma maneira, todo ponto histórico tem como atributos “código”, “localidade”, “tensão” e “corrente RMS”, é identificada pelo “código”, e está obrigatoriamente relacionada a exatamente um “engenheiro”, que na verdade está representando um “usuário”.

A representação em diagramas de entidade e relacionamento descreve a implementação da “base de dados” utilizada no projeto. O entendimento destas representações é de extrema necessidade para possibilitar as operações sobre o banco de dados de maneira consistente, e também para possibilitar, de forma flexível, eventuais mudanças ou incrementos futuros.

5.2.2 Modelo Lógico

Um modelo lógico é uma descrição de um banco de dados no nível de abstração visto pelo usuário do “SGDB”. Assim, o modelo lógico é dependente do tipo particular de “SGDB” que está sendo usado. Neste trabalho os modelos lógicos referem-se a um banco de dados relacional, cujos dados estão organizados na forma de tabelas.

O modelo lógico de um “BD” relacional deve definir quais as relações que o banco contém e, para cada tabela, quais os nomes das colunas ou domínios. Para manter o relacionamento entre as tabelas, utilizam-se cópias das colunas ou campos entre as tabelas, e para identificar uma única linha (ou tupla, nomenclatura mais acadêmica) em uma dada tabela, é necessário que se defina as chamadas “chaves primárias”.

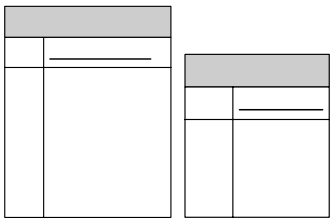
A “chave primária”, conforme dito anteriormente, é uma coluna ou uma combinação de colunas cujos valores distinguem uma linha das demais dentro de uma tabela. Outro tipo de chave de extrema utilidade neste projeto é a “chave estrangeira”. Esta chave é uma coluna ou uma combinação de colunas, cujos valores aparecem necessariamente na chave primária de alguma outra tabela. É ela quem permite a implementação de relacionamentos no banco de dados relacional. No próximo índice, será descrita a notação dos modelos relacionais.

5.2.2.1 Conversão do Modelo Conceitual em Modelo Lógico

A regra geral para conversão do modelo ER para modelo relacional é: toda entidade transforma-se em tabela, e seus atributos em campos (colunas desta tabela, onde o atributo identificador define a chave primária).

Por questões facilitadoras ao entendimento dos relacionamentos, nem todos os atributos das entidades e dos relacionamentos foram representados no modelo conceitual. Entretanto, neste momento serão descritas as demais propriedades anteriormente escondidas.

A Figura 5.13 mostra as tabelas “cadastro” e “ponto”. Neste modelo, observa-se que as chaves primárias são, na tabela “cadastro”, o “cod_empresa”, e na tabela “ponto”, “cod_ponto”. O campo “cod_empresa” é referenciado na tabela “ponto” através do campo “empresa”, ou seja, este campo corresponde à chave estrangeira da tabela “ponto”. Esta relação serve para identificar quando irá ocorrer um determinado ponto para um cliente cadastrado.



CADASTRO = {cód_empresa + nome_emp. + cid. + end. + n° + compl. + fone + data}.

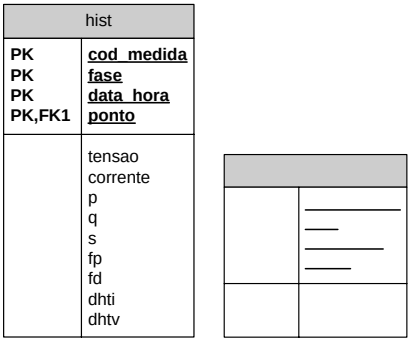
PONTO = {cód_ponto + localidade + endereço + numero + empresa}.

Figura 5.13 – Tabelas de “cadastro” e “ponto”

Vale a pena ressaltar que a regra geral, apesar de tornar o sistema mais flexível, normalmente gera muitas tabelas, o que pode comprometer o desempenho do sistema.

A seguir, estão apresentadas as tabelas “hist” e “inst”. Observa-se que as chaves primárias da primeira tabela são “cód_medida”, “fase”, “data_hora” e “ponto”, e sua chave estrangeira também é representada pelo campo “ponto”. Isto significa que a medição é identificada através de uma chave composta (as quatro chaves primárias identificadas acima),

e a chave estrangeira identifica o ponto de medição capacitado a realizar as medidas dos parâmetros de performance. Da mesma maneira ocorre entre as tabelas “ponto” e “inst”.

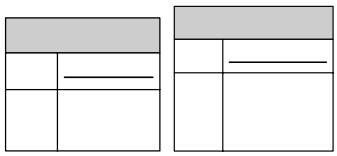


HIST = {cód_medida + fase + data_hora + p+ ponto + tensão + corrente + p + q + s + fp+ fd + dhtv + dhti}.

INST = {cód_medida + fase + data_hora + ponto + tensão + corrente}.

Figura 5.14 – Entidades armazenadoras de grandezas elétricas.

O relacionamento ternário entre as entidades “ponto” e “modelo”, realizado pela entidade “equipamento”, e descrita na Figura 5.11, é representado pelas tabelas baixo.



EQUIPAMENTO = {cód_equip + modelos + ponto}.

MODELOS = {cód_modelo + modelo + fabricante + empresa}.

Figura 5.15 – Entidades “equipamento” e “modelos”.

Neste relacionamento, a tabela “equipamento” apresenta como campos a chave primária “cód_equip”, e duas chaves estrangeiras, “modelo” e “ponto”. A chave estrangeira “modelo”, referencia a chave primária “cod_modelo”, e a outra chave estrangeira, “ponto”, referencia a chave primária “cod_ponto”, da tabela “ponto”. Na tabela “modelos” tem-se a chave

estrangeira “empresa”, que referencia a chave primária “cod_empresa” da tabela “cadastro”. Desta forma é possível identificar quais os medidores estão instalados para cada empresa cadastrada.

O modelo lógico completo com suas interligações está mostrado na Figura 5.16.

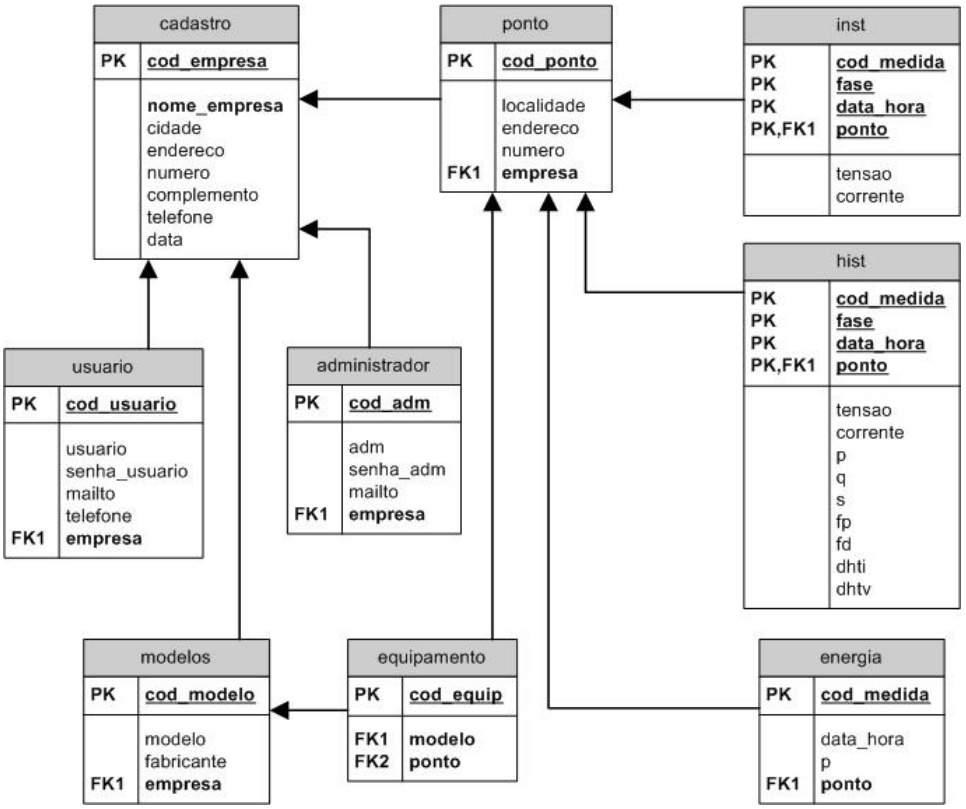


Figura 5.16 – Modelo lógico do banco de dados.

5.2.3 Projeto físico

Depois de realizada a análise conceitual e o desenvolvimento dos modelos conceitual e lógico, é necessário definir como o banco de dados será fisicamente, de que tipo serão os dados e qual será o “SGDB” utilizado. Como já descrito anteriormente e justificado, o banco de dados escolhido foi o “PostreSQL”, o qual tem o “PgAdmin” como sistema gerenciador.

5.2.4 Implementação

A implementação do banco é a fiel reprodução da estruturação teórica realizada. Nesta etapa, é demonstrado como são implementadas as tabelas e os relacionamentos entre estas, através das chaves primárias e estrangeiras definidas na modelagem conceitual. A Figura 5.17 demonstra como é construída uma tabela utilizando o SGDB.

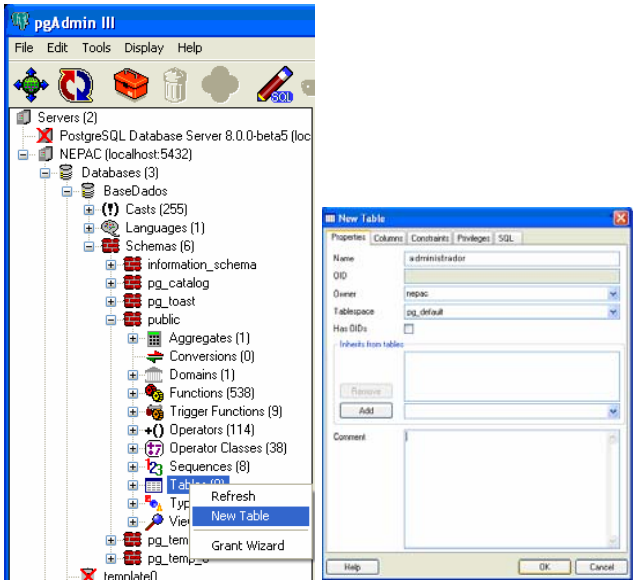


Figura 5.17 – Ambiente de desenvolvimento do banco de dados.

5.3 Interface WEB

Uma boa interação entre o gerenciador, usuários e o sistema, garante uma maior flexibilidade na tomada de decisões, conferindo um adequado grau de confiabilidade ao sistema. A interface, que se dá por meio de páginas dinâmicas disponibilizadas via “WEB”, permitirá ao gerenciador controlar o acesso de usuários às informações armazenadas no banco, cadastrar e descadastrar medidores e usuários do sistema assim como visualizar formas de onda e os parâmetros de performance instantâneos.

Para a execução de tarefas como inserção, recuperação, exclusão e alteração de dados no banco fez-se uso da linguagem SQL.

Também por meio desta linguagem foram implementadas funções como máximo e mínimo, valor médio, entre outras.

5.4 Servidor Apache

O servidor “Apache” é um dos melhores servidores “WEB” distribuídos de acordo com a filosofia de software livre. Totalmente compatível com o protocolo “HTTP”, suas funcionalidades são mantidas através de uma estrutura de módulos, permitindo inclusive ao usuário escrever seus próprios módulos, utilizando a “API” do software.

Este foi o servidor escolhido por ser, em primeiro lugar, gratuito, e também por apresentar grande estabilidade e variedade de ferramentas.

5.5 Utilização da Página da Internet

5.5.1 Página Inicial

A estruturação do banco de dados e da página da internet foi feita de forma a que diversas empresas e usuários pudessem utilizar os recursos de visualização remota das grandezas elétricas.

Pode-se observar na Figura 5.18 que na tela inicial da página, o usuário pode visualizar quais empresas estão cadastradas, e fazer o “login” na que estiver registrado. Após isto, se o mesmo for administrador (da empresa), ele poderá cadastrar/descadastrar pontos de medição, equipamentos, usuários, entre outros parâmetros. Mas caso não o seja, poderá somente visualizar os dados e as configurações, sem alterá-los.

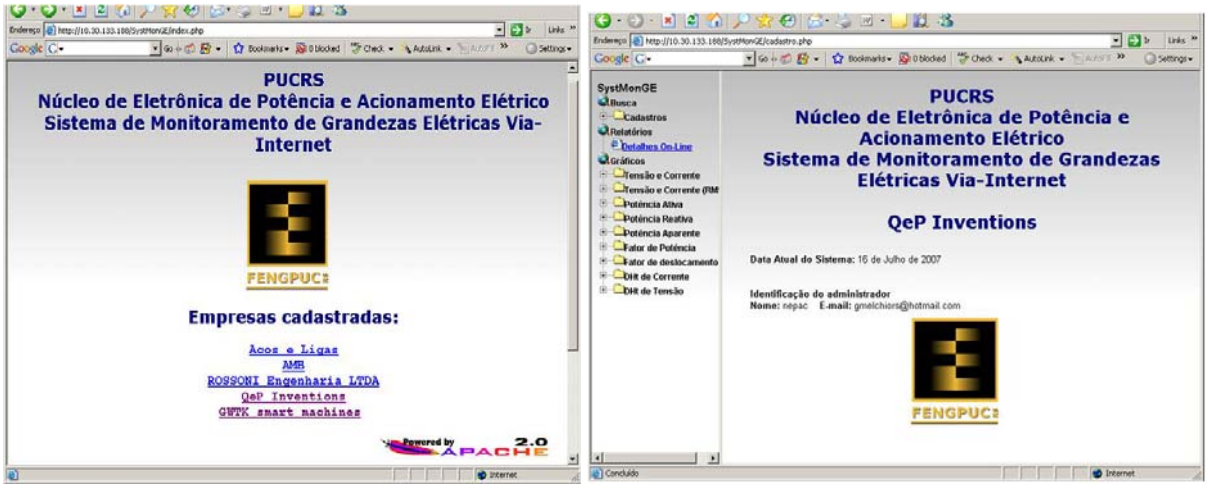


Figura 5.18 – Telas iniciais da página.

Abaixo estão mostradas duas telas de cadastro de usuário, realizadas por um administrador da empresa. Pode-se observar, à esquerda, a tela de inserção dos parâmetros do novo usuário, e à direita, a confirmação do cadastramento.

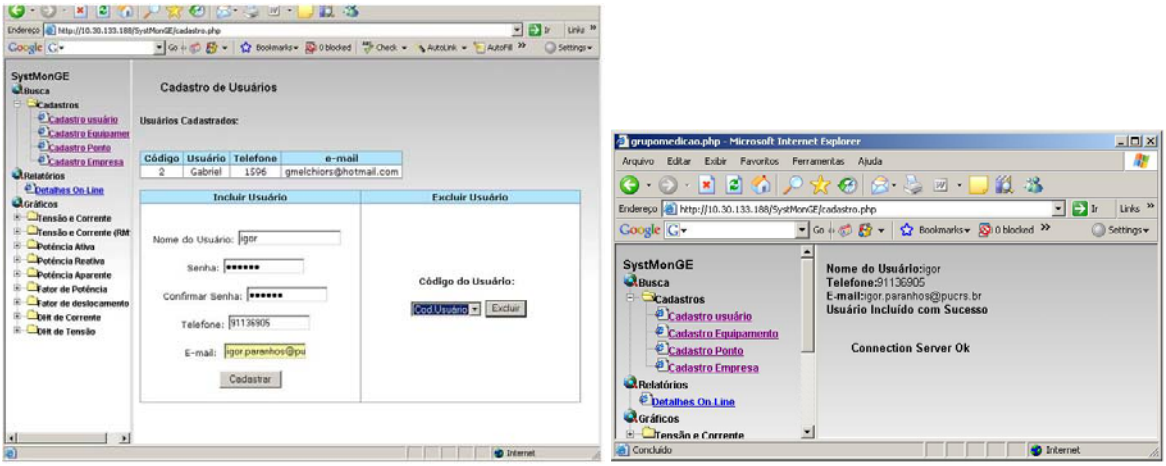


Figura 5.19 – Cadastro de usuário.

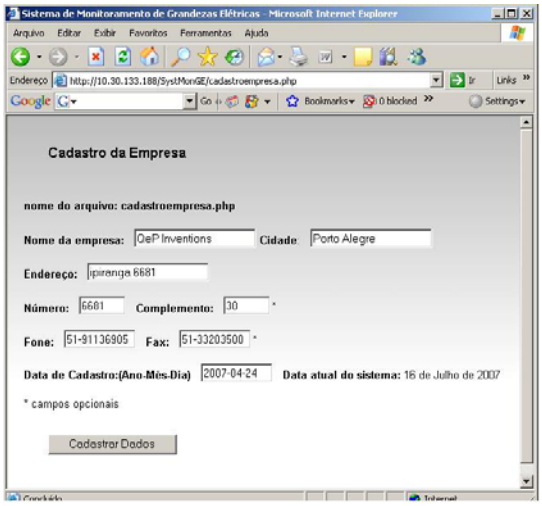


Figura 5.20 – Cadastro da empresa.

5.5.2 Configuração de Empresas e Usuários

Embora os administradores das empresas tenham autonomia para realizar cadastros e descadastros de pessoas e objetos, estes não têm permissão para cadastrar empresas ou realizar demais configurações. Esta tarefa cabe ao administrador do sistema (o gerenciador do banco de dados). A Figura 5.21 ilustra o cadastramento de uma nova empresa no sistema.

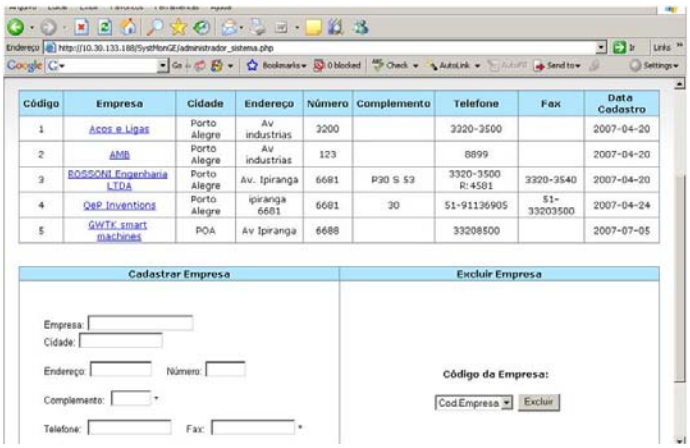


Figura 5.21 – Cadastro de empresas.

5.5.3 Consulta aos Dados

A função de consulta dos dados é uma das mais importantes desta página da internet, pois esta é o objetivo final dos usuários deste recurso.

Depois de estar conectado no sistema, por meio do “login”, o usuário pode ter acesso às informações elétricas relativas à sua empresa. Tanto os parâmetros de performance quanto a forma de onda dos sinais elétricos podem ser selecionados pelo mesmo, através de parametrizações por ponto de medição, equipamento ou data e hora.

Pode-se observar nas Figura 5.22 e a visualização da forma de onda dos sinais elétricos, obtida por meio de uma seleção do dia e do horário desejados. É interessante também observar que este filtro é feito de forma a induzir o usuário a escolher parâmetros que realmente existam no banco de dados. Ou seja, a escolha dos valores é feita através de “combo boxes”, na seguinte ordem: ano, mês, dia e hora. Dessa forma, o sistema evita a escolha de um horário em que não houve medições.

Já na Figura 5.24 pode-se ver uma consulta dos valores eficazes de tensão e corrente registrados para um determinado período de tempo. Como os valores mostrados na figura são simulados (lidos do medidor virtual, que gera senóides de amplitude variável randomicamente), os valores eficazes aparecem oscilando bastante.

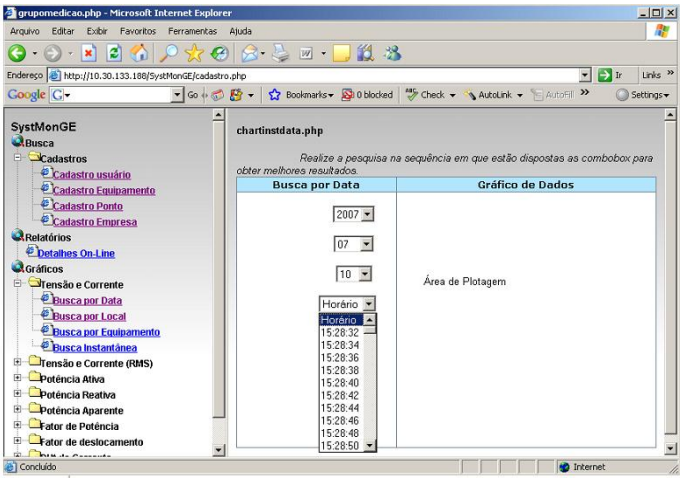
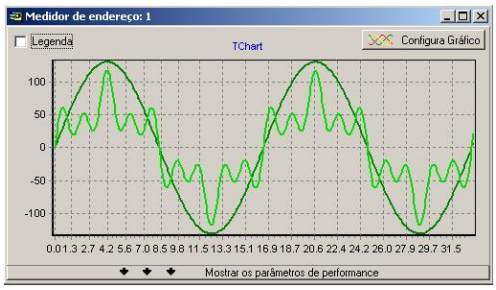
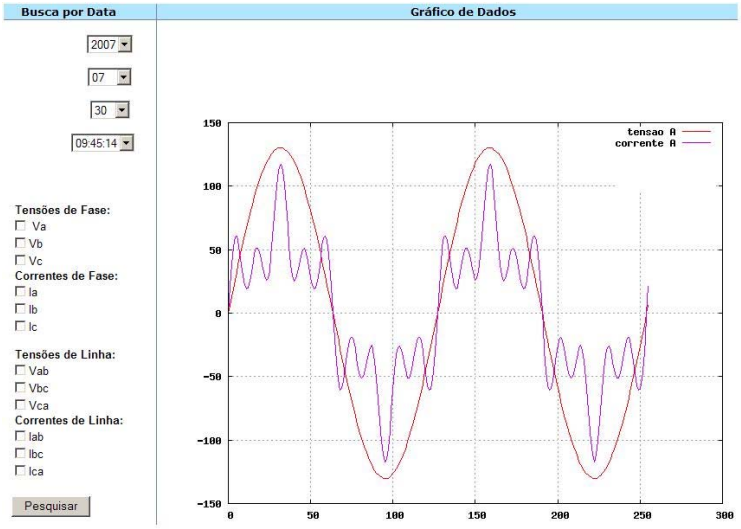


Figura 5.22 – Filtro dos dados por data e hora.



(1)



(2)

Figura 5.23 – Em (1), formas de onda obtidas pelo software, e armazenadas no banco. Em (2), reprodução desses sinais armazenados no banco, na data e hora especificados.

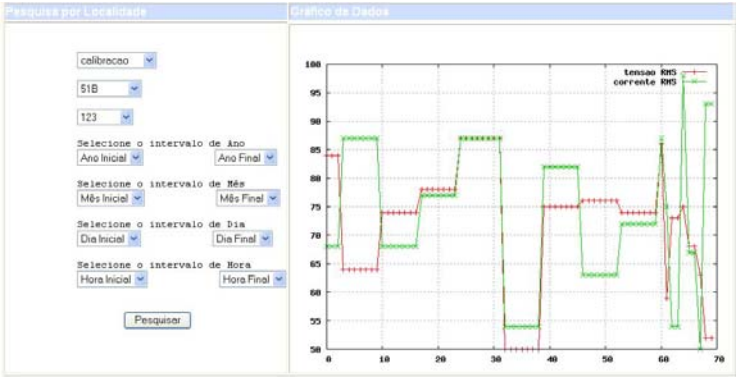


Figura 5.24 – Gráfico de tensões e correntes eficazes, em um período de tempo.

5.6 Conclusões

A utilização de uma página da internet para a consulta dos dados armazenados num banco proporcionou um significado bastante interessante ao sistema.

Por meio das últimas figuras deste capítulo pôde-se entender que muitas empresas podem utilizar o sistema independentemente, cada uma com direitos de delimitação de acesso (através dos administradores), e condições de cadastramento / descadastramento de usuários, equipamentos, pontos de medição, etc.

Além do mais, a possibilidade de visualização remota das formas de onda instantâneas e dos parâmetros de performance de diversos pontos de um planta industrial, permitiu o entendimento do valor que um sistema desses têm para usuários de energia elétrica, que estão preocupados em monitorar a qualidade da energia, de forma a assegurar a continuidade do processo produtivo.

A verificação de uma tendência do sistema elétrico a uma possível falha a partir das informações passadas e atuais, pode conduzir o usuário à uma manutenção preditiva e preventiva de equipamentos e máquinas, poupando o mesmo de possíveis prejuízos.

Ademais disso, existe a possibilidade da inserção de inteligência artificial no banco de dados para a interpretação dos dados, o que representa uma considerável ferramenta de auxílio para o controle da qualidade da energia.

6 CONCLUSÕES E DESENVOLVIMENTOS FUTUROS

6.1 Conclusões

De acordo com o exposto ao longo deste trabalho, o monitoramento de energia atualmente se torna cada vez mais importante e necessário, tanto para consumidores que, por exemplo, não querem ter a sua linha de produção parada devido a um mal funcionamento de equipamentos micro-processados sensíveis a distúrbios na qualidade da energia, como para as concessionárias, que querem ter controle do sistema para evitar perdas e entregar a energia livre de distúrbios aos seus clientes.

Nos últimos anos as cargas encontradas nas indústrias mudaram notoriamente as suas características elétricas. No passado, exemplos de cargas encontradas na rede elétrica eram transformadores, motores de indução, lâmpadas incandescentes e aquecedores resistivos. Hoje em dia, uma boa parte das cargas tem circuitos eletrônicos associados em sua concepção. Estes circuitos podem ser controladores, conversores de potência, entre outros, que agregam uma característica bem comum às mesmas: a não linearidade, o que origina muitos dos distúrbios elétricos encontrados na rede.

Em meio a este ambiente, surgiram alguns sistemas dedicados a adquirir e processar os sinais elétricos de corrente e tensão da rede, oferecendo informações e recursos ao usuário.

Estes sistemas ficaram conhecidos como “sistema de monitoramento elétrico”, os quais, dependendo da proposta, apresentam diferentes níveis tecnológicos e, consequentemente, custos.

Os benefícios do monitoramento da energia são muitos. Como citado acima, as empresas podem se prevenir de gastos bastante grandes devido à paradas indesejadas na produção, principalmente nas indústrias química e de transformação, onde uma falha no processo pode desperdiçar grande parte da matéria prima envolvida. Ademais disso, as informações de energia podem ser integradas às de produção e gestão, de forma a empresa tomar decisões mediante informações compartilhadas, e não a partir de soluções isoladas.

Com a evolução da tecnologia da informação muitos sistemas passaram a armazenar as informações adquiridas em um banco de dados, e disponibilizá-las pela internet, o que possibilitou uma aproximação administrativa de empresas globais, geograficamente distribuídas, de forma rápida e de baixo custo.

E além do recurso de acesso aos dados pela internet, o banco de dados proporcionou a geração de relatórios de diversos tipos, onde o usuário pôde selecionar os dados de interesse (valores eficazes, potências ou distorções harmônicas, por exemplo) por data, localidade, valores específicos, etc. Isto possibilitou não só uma fácil interpretação dos dados, como também pôde servir como detecção de algum distúrbio nos sinais elétricos, por meio de um gatilhamento configurado no banco. E ainda, há a possibilidade de se agregar recursos como inteligência artificial na análise dos dados, sendo uma considerável ferramenta para análise da qualidade da energia.

Entre outros fatores, os argumentos acima relatados fundamentam a crescente demanda de empresas pequenas, médias e grandes, assim como residências e prédios comerciais, por sistemas de monitoramento da energia elétrica, que adquiram as informações de interesse e as disponibilizem para o usuário fazer benefício dela.

Por isso, este foi o tema desta dissertação. Além do panorama a respeito do assunto, o trabalho teve como proposta a implementação de um medidor digital trifásico de grandezas elétricas, um software gerenciador “multi-medidor” que coletasse e processasse as informações, gerando gráficos, apresentando os parâmetros de performance e as armazenando em uma base de dados, um banco de dados robusto, para operação “multi-usuário” e uma página da internet, para a visualização remota dos dados. E sempre em paralelo ao

desenvolvimento de cada módulo, buscou-se o baixo custo, de forma a este sistema se situar entre os mais sofisticados e os mais simples, e financeiramente acessível não só para empresas de maior porte, mas também para empresas com recursos limitados.

Este é o ponto que o trabalho se dispôs a alcançar: um sistema de baixo custo, que agregasse um serviço de banco de dados, que atualmente é a grande tendência.

Esse baixo custo foi conseguido por meio da utilização de componentes de hardware mais simples e de softwares gratuitos, como é o caso “SGDB” “PgAdmin” e do banco de dados “PostgreSQL”.

Quanto à implementação prática, os objetivos foram atingidos de forma satisfatória. Conseguiu-se realizar a aquisição trifásica dos sinais elétricos, e fazer com que essas informações fossem passadas ao usuário na forma de gráficos e cálculos, assim como fossem armazenadas para posteriores consultas, que inclusive poderiam ser através da internet.

O medidor digital ofereceu um desempenho adequado, conciliando o nível tecnológico desejado com o baixo custo. E isso se deve ao micro-controlador escolhido ter cumprido todas tarefas propostas, sem deixar a desejar. Inicialmente colocava-se em dúvida se a freqüência de operação permitiria que o dispositivo lesse os sinais, e a cada ponto gravasse em uma memória externa, sem perder o tempo do próximo ponto. Mas felizmente o dispositivo se mostrou mais rápido para essa função que o período entre duas aquisições de ponto.

Ainda sobre o micro-controlador, os 12 bits e a freqüência de leitura do “AD” não permitiram que houvesse erros significativos de amostragem ou quantização, respectivamente. E a memória “RAM”, apesar de menor do que em outros dispositivos, não atrapalhou o desenvolvimento da plataforma, devido à utilização da memória externa, que se demonstrou bastante confiável.

Quanto à linguagem “C” utilizada para a elaboração do programa do micro-controlador, esta proporcionou facilidade e agilidade na programação, sendo que a versão do “IAR System” utilizada não era paga, contribuindo a manutenção do baixo custo. Deve-se salientar, porém, que o desenvolvimento dos algoritmos e o entendimento dos registradores utilizados para o correto funcionamento do micro-controlador não foram triviais. Bem pelo contrário, devido à dificuldade de análise das variáveis envolvidas com o programa em execução, a correção dos problemas se tornava de entendimento bastante abstrato.

Já a programação do software gerenciador, sob a plataforma Delphi 5, trouxe consigo a facilidade da interrupção “online” no programa para análise dos valores das variáveis. Além disso, tinha-se a possibilidade de executar o programa linha a linha, acompanhando os acontecimentos e verificando o exato momento da falha. Esse recurso realmente facilitou a implementação do código fonte, oferecendo velocidade no desenvolvimento. A respeito do ambiente de desenvolvimento, este se mostrou bastante amigável, e robusto o suficiente para oferecer todos os recursos necessários.

Entretanto, da mesma forma como foi no programa do micro-controlador, a elaboração dos algoritmos e das estratégias para que o funcionamento se desse de acordo com o desejado não foi nada fácil. Além da dificuldade com a comunicação serial, foi bastante difícil organizar uma estrutura que armazenasse as características de cada medidor, fizesse os pedidos de acordo com a configuração, recebesse a reposta ao pedido, plotasse no devido gráfico, calculasse e mostrasse os resultados na devida tabela (cada medidor tem a sua janela) e gravasse no banco de dados, sem haver mistura nenhuma dos dados.

Isto foi conseguido por meio de uma estruturação do aplicativo baseado nas técnicas de engenharia de software, e também fazendo-se uso dos recursos de orientação a objetos. Esta última técnica possibilitou um adequado nível de organização e entendimento dos algoritmos, que foi possível criar um programa relativamente complexo, com um número reduzido de linhas de código.

Alguns recursos foram chave para o funcionamento do software. Sem eles, possivelmente a estrutura não teria funcionado. O primeiro é a classe “TThread”, que permitiu processamentos paralelos de eventos, oferecendo vazão ao fluxo de dados. Por segundo, foi a classe “TList”, que permitiu a geração de listas de ponteiros e fácil manipulação dos mesmos. E também, muito importante, foi a utilização de ponteiros de objetos, os quais permitiram a identificação dos parâmetros e das janelas de gráfico e cálculo no software.

Outra ferramenta que auxiliou bastante na elaboração do processo de comunicação de dados e na conferência dos algoritmos de cálculo foi o “medidor virtual”, que também foi desenvolvido neste trabalho, em Delphi 5, e que está mostrado no item 3.6.

Enfim, a estruturação e organização do aplicativo foram fundamentais para a flexibilidade do mesmo, manutenção e futuras modificações. Pode-se concluir que, como a literatura indica, vale a pena investir um pouco de tempo no planejamento do projeto, embora

muitas vezes durante a implementação voltou-se aos diagramas estruturais para se fazer modificações. Sendo assim, pode-se dizer que, para este trabalho, o mais adequado foi o andamento em paralelo entre projeto e prática.

Da mesma forma como no software, a estruturação “no papel” e a orientação a objetos foram fundamentais para a construção do banco de dados e da página da “WEB”. A correta modelagem das entidades e dos atributos, assim como dos relacionamentos no diagrama “ER” proporcionou, após a conversão para o modelo lógico, um banco de dados íntegro.

Numa visão mais global do sistema, o funcionamento em conjunto da base de dados com a página da internet possibilitaram a visualização “quase que instantânea” dos dados medidos pelos medidores instalados nos pontos de monitoramento. Essa expressão “quase” foi dada, pois alguns segundos podem ter passado desde o momento exato da medição. Entretanto, tendo-se claro de que essa medição pode ter sido feito do outro lado do mundo, pode-se dizer que a visualização da mesma está sendo instantânea, oferecendo tempo suficiente para uma atitude preventiva.

Sendo assim, o objetivo do sistema foi alcançado, embora algumas limitações possam ser citadas.

O software, por exemplo, ainda é um arquivo executável, que necessita a instalação manual de outros arquivos importantes para o funcionamento. Sendo assim, seria interessante que houvesse um instalador automático para o programa. Outro detalhe é que os cálculos são feitos assumindo-se uma freqüência de 60 Hz, o que pode gerar erros. Deve-se, para sanar esta limitação, programar o software para pedir ao hardware a real freqüência do sinal elétrico. Ainda referente ao software, como a entrada dos parâmetros (número de ciclos, pontos por ciclo...) é feita manualmente (pela digitação no teclado), pode haver um erro na comunicação de dados, pois, como já explicado, os pacotes devem conter exatos 64 pontos.

O hardware, por sua vez, apresentou erros mediante um recadastramento no software, ou seja, quando um segundo ou terceiro pacote de configuração era recebido. Além disso, a freqüência calculada muitas vezes apresenta um pequeno desvio do real valor. E ainda, os ruídos nas placas de circuito impresso não eram desprezíveis, sendo algumas vezes perceptíveis na leitura do “AD”.

O banco de dados e a página da internet apresentam uma limitação na velocidade de busca às informações, ou seja, no momento da filtragem dos dados (escolha da data e da hora), o sistema poderia apresentar maior agilidade. Ademais disso, a escolha das fases também apresentou erros, algumas vezes apresentando um resultado que não correspondia ao que foi solicitado.

6.2 Desenvolvimentos Futuros

Devido à velocidade com a qual a ciência evolui nos dias de hoje, onde a todo o momento uma nova tecnologia surge, uma nova ferramenta ou um novo recurso é disponibilizado no mercado, a maioria dos sistemas baseados em software e/ou micro-eletrônica estão em constante mudança.

Com o sistema proposto e implementado neste trabalho não é diferente. Principalmente devido às diferentes áreas tecnológicas que ele abrange. Possivelmente novos micro-controladores com melhor desempenho aparecerão, recursos de software, meios físico e protocolo de comunicação, recursos de banco de dados e páginas de internet, entre outros.

Mas, desprendendo-se do futuro e focando mais no presente, pode-se citar alguns recursos nos quais este sistema pode melhorar.

Em primeiro lugar, embora a comunicação serial tenha funcionado bem (para a proposta), e seja de um custo menor, em comparação com outras, deve-se almejar taxas de transferência maiores, como é o caso do padrão “Ethernet”, ou mesmo da comunicação via porta “USB”. Como mostrado no capítulo do hardware, a comunicação “Ethernet”, cuja taxa de transmissão é 10Mbps na versão mais simples, possibilita a transmissão de uma quantidade muito maior de dados em um mesmo período de tempo, comparando com a serial. Isto significa a possibilidade de aquisições de um número maior de ciclos (20, 30, 100...) e de pontos por ciclo (512, 1024). E migrar para essa tecnologia não iria requerer obrigatoriamente mudança de protocolo, pois o “ModBus / RTU” também tem a sua versão via “TCP/IP”.

Outra possibilidade é agregar funções de qualímetro, onde o medidor em paralelo com a aquisição dos sinais verifica os vetores em busca de distúrbio elétricos, registrando o evento

em memória, identificando o tipo e o horário do acontecimento. Mas para isso, o micro-controlador deveria ser substituído por outro com características mais avançadas, de forma que o processamento se desse com uma freqüência mais alta. Outra possibilidade é a inserção de um co-processador em paralelo.

Ademais disso, além do software gerenciador que pode oferecer mais recursos, pode-se implementar inteligência artificial ao banco de dados, com o intuito analisar os dados em busca de distúrbios nos sinais elétricos de tensão e corrente.

Ainda, existe uma tendência bastante grande para que o hardware escreva os sinais elétricos diretamente no banco de dados, economizando a intermediação de um software local, e permitindo que a análise seja feita diretamente pela internet.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ALECRIM, Emerson. **Tecnologia BlueTooth**. Disponível em <http://www.infowester.com/bluetooth.php> Acessado em 24/07/2007.

[2] Alfa Instruments. **Comandos de pesagem para modbus RTU / ASCII**, 2002. 46p. Apostila.

[3] ALVES, Mário F., FERNANDES, Délio E. B. **Um Sistema Para o Gerenciamento da qualidade da Energia Elétrica**. *Anais do XIII Congresso Brasileiro de Automática – CBA 2000*, pp. 1932-1937, 11-14 de Setembro de 2000.

[4] ANQUETIL, Nicolas. Diagrama de Casos de Uso. Disponível em: <www.ucb.br/ucbtic/mgcti/paginapessoalprof/Nicolas/Disciplinas/UML/node5.html> Acesso em 25/07/2007.

[5] ARAÚJO, K. F. C. de. **Delphi 5 com Banco de Dados para Internet**. 2° edição, São Paulo: Érica, 2000. 324 p.

[6] BASU, M.; BASU, B. **Power Quality (PQ) Signals Analysis by Continuous Wavelet Transform.** *IEEE Power Electronics Specialists Conference*. PESC 2007. p. 2614-2618.

[7] BAUER, Ivo. **Modlink.** Disponível em <http://www.ozm.cz/ivobauer/modlink/Index.htm>. Acesso em 25/07/2007.

[8] BOLLEN, Math H. J. **Understanding Power Quality Problems.** New York: IEEE Press Marketing, 2000.

[9] BONATTO, B. D., WATANABE, E. H., MERTENS JR., E. A., **Eletrônica de Potência e Qualidade de Energia Elétrica: Pesquisas Cooperativas na ELECTRO, COPPE/UFRJ e UBC.** Disponível em

<http://www.aneel.gov.br/biblioteca/Citenel2001/trabalhos%5C09.pdf> Acesso em 22/07/2007.

[10] BRIGHT, James A.; WIE-JEN LEE. **Integrates Monitoring, Protection and Control System for Industrial and Commercial Power Systems.** *IEEE Transactions on Industry Applications*, Vol. 36, n° 1, p. 11-15, Jan. 2000.

[11] CÂMARA, F. **Conhecendo o Delphi 6**, Florianópolis: Visual Books, 2001. 144 p.

[12] CANTU, Marco. **Dominando o Delphi 5: a Bíblia**. São Paulo, Makron Books, 2000. 860p.

[13] CASTALDO, D., GALLO, D., LANDI, C., LANGELLA, R., TESTA, A. **Power Quality Analisys: a Distributed Measurements System**. *IEEE Power Tech Conference Proceedings*, Vol. 3, 6 pp., Bologna, 23-26 Junho de 2003.

[14] CHAN, Shun-Yu, TENG, Jen-Hao, SU, Yu-Te; **Internet-Based Virtual Power Quality Recorders.** *IEEE TENCON 2004*, vol. C, pp. 287- 290, 21-24 de Nov. de 2004.

[15] CHANG, D., CHENG, Ming-Jieh, CHAN, Chun-Yu. **Development of a Novel Power Quality Monitoring and Report-Back System**. IEEE *TECON 2006 Region 10 Conference*. pp. 1-4, 2006.

[16] CHILUKURI, M.V., DASH, P.K., BASU, K.P. **Time-frequency based pattern recognition technique for detection and classification of power quality disturbances**. *IEEE International Conference on Analog and Digital Techniques in Electrical Engineering*, Vol. 3, pp. 260-263, 2004.

[17] CHEN, Y., HWANG, J. K. **A Reliable Energy Information system for Promoting Voluntary Energy Conservation Benefits**. *IEEE Transactions on Power Delivery*, vol. 21 n°1, pp. 102-107, Janeiro de 2006.

[18] CHEN, S., ZHANG, C. L., LIU, Y.Z. **A Multi-Channel Monitoring System for System-Wide Power Quality Measurements.** *IEEE International Conference on Power System Technology*, vol. 2, pp. 953-958. 2000.

[19] CHUN LI, WILSUN XU, HUGHES B., GURNEY J., NEILSON B. **Virtual PQ Troubleshooter**. *IEEE power & Energy Magazine*, pp. 24-31, may/june 2003.

[20] CHUNG, I.Y., WON, D.J., KIM, J.M., AHN, S.J., MOON, S.I., SEO, J.C., CHOE, J.W. **Development of Power Quality Diagnosis System for Power Quality Improvement**. *IEEE Power Engineering Society Meeting*, pp. 1256-1261, 2003.

[21] Confederação Nacional da Industria. **Normas Internacionais**. Disponível em <http://www.normalizacao.cni.org.br/normas_tecnicas_internacionais.htm> Acesso em 24/07/2007.

[22] CRISTALDI, C., FERRERO, A. **A Method and Related Digital Instrument for the Measurement of the Electric Power Quality**. *IEEE Transactions on Power Delivery*, vol. 10, n° 3, pp. 1183-1188, Julho de 1995.

[23] _____, _____, SALICONE, S. **A Distributed System for Eletric Power Quality Measurements**. *IEEE Instrumentation and Measurement Tecnology Conference*, pp. 2130-2135, Budapest, Hungaria, 21-23 de Maio de 2001.

[24] COURY, D.V. **Qualidade de Energia.** Disponível em <www.sel.eesc.usp.br/protecao/qualidadegeral.htm> Acesso em 23/07/2007.

[25] DATE, C. J. **Introdução a Sistemas de Banco de Dados**. 7ª edição, 3ª tiragem. Rio de Janeiro: Campus, 2000.

[26] FERREIRA, Marcos dos S. **Delphi 5.0: Tópicos Avançados.** São Paulo: Érica, 2000, 270p.

[27] FRANCO, E. **Qualidade de Energia.** Disponível em <www.engecomp.com.br/pow_qual.htm> Acesso em 23/07/2007.

[28] GHOSH, Arindom, LEDWICH, Gerard. **Power Quality Enhancement Using Custom Power Devices**. Massachusetts, USA: Kluwer Academic Publishers, 460p. 2002.

[29] GUNTER, Erich W. **On Creating a New Format for Power Quality and Quantity Data Interchange.** *IEEE PES TD 2005/2006*, pp. 354- 358, 21-24 de Maio de 2006.

[30] HAKIMIE, H., RAMACHANDARAMURTHY, V.K., MUKERJEE, R.N. **Intelligence-Driven Power Quality Monitoring**. *IEEE International Conference on Power Electronics and Drives Systems*, vol. 2, pp. 1278-1282, Nov 2005.

[31] HENVILLE, C.F. **Digital relay reports verify power system models**. *IEEE Transactions on Power Delivery*, Vol. 13, pp. 386-393, Abril de 1998.

[32] HEUSER, Carlos Alberto. **Projeto de Banco de Dados**. N° 4, 5° ed. Porto Alegre: Sagra Luzzato, 2004.

[33] IBRAHIM, W. R. A.; MORCOS, M. M. **Artificial Intelligence and Advanced Mathematical Tools for Power Quality.** *IEEE Transactions on Power Delivery*, Vol. 17, n° 2, p. 668-673, April 2002.

[34] _____, _____, KREISS, D. G. **An adaptive neuro-fuzzy intelligent tool and expert system forpower quality analysis. I. An introduction**. *IEEE Power Engineering Society Summer Meeting*, vol. 1, pp. 493-498, 18-22 de Julho de 1999.

[35] JIN, Wang, XINRAN, Li, SHENG, Su. **Research on the application of the recorded fault data analysis based on COMTRADE standard in electric power load modeling**. *IEEE Power Engineering Conference*, vol. 2, pp. 1058-1063, Dec. de 2005.

[36] KANITPANYACHAROEAN, W., PREMRUDEEPREECHACHARN, S. **Power quality problem classification using wavelet transformation and artificial neural networks**. *IEEE Power Systems Conference and Exposition*, Vol. 3, pp. 1496-1501, 2004.

[37] KEZUNOVIC, M., SEVCIK, D. R., LUNSFORD, R., POPOVIC, T. **Integration of Substation Data**, *Texas A&M Conference for Protective Relay Engineers, College Station*, pp. 505-510, Texas, March 2007.

[38] KHAN, A. K. **Monitoring Power for the Future**. *Power Engineering Journal*. Vol 15, pp. 81-85, April 2001.

[39] KIESSLING, G., SCHWABE, S. **Software Solution for Fault record Analysis in Power Transmission and Distribution**. *IEE Developments in Power System Protection*, vol. 1, pp. 192- 195, 5-8 April 2004

[40] KOAY, B. S., CHEAH, S. S., SNG, Y. H. **Design and Implementation of Bluetooth Energy Meter.** IEEE Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia, Vol. 3, pp. 1474-1477, 15-18 de Dezembro de 2003.

[41] LEE, R. P. K., TSE, N. **A Web-Based Multi-Channel Power Quality Monitoring System for a Large Network**. *IEE Power System Management and Control*, N° 488, 2002.

[42] LEOU, Rong-Ceng, CHANG, Ya-Chin, TENG, Jen-Hao. **A Web-based Power Quality Monitoring System**. *IEEE Power Engineering Society Summer Meeting*, vol.3, pp. 1504-1508, 2001.

[43] LESTON, L.A. **Desenvolvimento de um Qualímetro Medidor de Energia Elétrica Implementado com Microcontrolador**. Porto Alegre, 2003. 52p. Disciplina de estágio supervisionado (Engenharia de Controle e Automação). Pontifícia Universidade Católica do Rio Grande do sul.

[44] LIN, T., DOMIJAN, A. **Novel Índices for Power Quality Measurements.** IEEE 11th International Conference on Harmonics and Quality of Power, pp. 66-71, 2004.

[45] MAIA, Luiz Paulo, MORELLI, Edurado. **Programação Orientada a Objetos**. Disponível em <http://www.training.com.br/lpmaia/pub_prog_oo.htm/> Acesso em 25/07/2007.

[46] MÄKINEN, Antti, PARKKI, Marko. **Power Quality Monitoring as Integrated with Distribution Automation.** *IEE CIRED 2001, Conference Publication n° 482*, p. 2.54, 18-21 de Junho de 2001.

[47] MANMEK, T., GRANTHAM, C., PHUNG, T. **A New Efficient Algorithm for Online Measurement of Power System Quantities. Industrial Electronics Society**. *IECON 2004, 30th Annual Conference of IEEE*, Vol. 2, pp. 1184-1189, 2004.

[48] MANO, Otávio S. **Estudo e Desenvolvimento de um Medidor de Qualidade de energia Implementado em Hardware Flexível**. Trabalho de Conclusão de Curso – Pontifícia Universidade Católica do RS, Porto Alegre, 2005.

[49] MATOS, Glastone. **Guia Rárido da SQL**. Rio de Janeiro, SQL Magazine, 1°edição, p. 8-11, 2002.

[50] MAZUMDAR, Joy., VENAYGAMOORTHY, Ganesh K., HARLEY, Ronald G. **Application of Neural Networks for Data Modeling of Power Systems With Time Varing Nonlinear Loads**. *IEEE Procedings of the Symposium on Computational Intelligence and Data Mining*, pp. 705-711, 2007.

[51] MELCHIORS, José Gabriel. **Implementação de um Sistema de Monitoramento de Grandezas elétricas via Internet Conceituado em Software Livre**. Trabalho de Conclusão de Curso – Pontifícia Universidade Católica do RS, Porto Alegre, 2006.

[52] MELHORN, Christopher J., MCGRANAGHAN, Mark F. **Interpretation and Analysis of Power Quality Measurements**. *IEEE Transactions on Industry Applications*, vol. 31, pp. 1363-1370, Nov/Dez 1995.

[53] Modicon Inc. **Modbus over Serial line Specification & Implementation guide V1.0**. 2002 <http://www.modbus.org/> Acesso em 23/07/2007.

[54] _____. **Modbus application protocol Specification V1.1a**. 2002 <http://www.modbus.org/> Acesso em 23/07/2007.

[55] _____. **Modicon Modbus protocol reference guide**. 1996 <http://www.modbus.org/> Acesso em 23/07/2007.

[56] MONTEIRO, Maxwell E., MOURA, Elton S., DRAGO, Adrian B., BONINO, Luiz O. **An Internet-Based Power Quality Monitoring System**. *IEEE International Symposium on Industrial Electronics*, vol. 1, pp. 333-336, 9-11 de junho de 2003.

[57] MOUSSA, Abd-E., ABDALLAH, Emtethal N. **Hardware-Software Structure for On-Line Power Quality Assessment: Part 1**. *Procedings of the ASME/IEEE Joint Rail Conference*, pp. 147-152, 6-8 de Abril de 2004.

[58] MÜLLER, S. L. **Filtro Ativo Série para Condicionamento de Tensões**, Porto Alegre 2003. 239p. Dissertação de Mestrado em Engenharia Elétrica. Pontifícia Universidade Católica do RS.

[59] NICHOLS, Giselli. **Conquistando Novas Fronteiras PostgreSQL**. Rio de Janeiro, SQL Magazine, 1°edição, p. 42 e 43, 2002.

[60] OLIVER, J. A., LAWRENCE, R., BANERJEE, B. B. **Power Quality**. *IEEE Industry Applications Magazine*, pp. 21-30, Setembro/Outubro de 2002.

[61] PARANHOS, Ígor A., MANO, Otávio S., Melchiors, José G., LÍBANO, Fausto B., BRAGA, Rodrigo A. M., MÜLLER, Sérigo L., FEHLBERG, Rafael P. **Power Energy Meter in a Low Cost Hardware/Software**, *IEEE ISIE 2006*, Montréal, Canadá, 9-12 de Julho de 2006.

[62] _____, _____, _____, _____. **Power Quality Monitoring System in a Low Cost Hardware/Software.** IEEE EPE2007, Dinamarca, 2-5 de Set. de 2007.

[63] PAULI, S. KRAHL, B.O. LEUSCHNER, B. **Know Your Power**. *IEEE Industry Applications Magazine*, Vol 11, pp. 21- 30, Mar/Apr 2005.

[64] PEREIRA, T. F., **Delphi 5: banco de dados e MIDAS**, São Paulo: Érica, 346p, 2000.

[65] PEREIRA, Fábio. **Microcontroladores MSP430: Teoria e Prática.** 1ª Ed. São Paulo: Érica, 2005.

[66] Píccolo, H. L., **Estrutura de Dados**, Brasília: MSD, 159p, 2000.

[67] PINTO, J.S. de P. **Microsoft SQL-Server**. Paraná. 62p. Apostila

[68] PHADKE, A. G., JODICE, J. A. **COMTRADE: A New Standard for Common format for Transient Data Exchange**. *IEEE Transactions on Power Delivery*, vol. 7 n°4, pp. 1920-1926, Outubro de 1992.

[69] PRESSMAN, R.S., **Engenharia de Software**, São Paulo: Makron Books, 1056p. 1995.

[70] REISDORPH, K., **Aprenda em 21 dias Delphi 4**, Rio de Janeiro: Campos, 919p, 1999.

[71] RMS Sistemas Eletrônicos. **ANAWIN - Programa de Análise em PC.** Disponível em: <http://www.rms.ind.br/mais8.html> Acessado em: 24/07/2007.

[72] RUBENKING, N. J., **Programação em Delphi Para Leigos**, São Paulo: Berkeley, 372p, 1995.

[73] SAWYER, D. **Non-stop Monitoring** [power quality]. *IEE Review*, vol. 45, pp. 126-127, May 1999.

[74] SEIXAS FILHO, C. **Elementos de programação Multithreading em delphi**. Disponível em

<http://www.cpdee.ufmg.br/~seixas/PaginaSDA/Download/SDADownload.htm> Acesso em 22/07/2007.

[75] _____. **Protocolos Orientados a Caracter**. Disponível em <http://www.cpdee.ufmg.br/~seixas/PaginaSDA/Download/SDADownload.htm> Acesso em 22/07/2007.

[76] SLATTERY, Colm. **High Performance Multichannel Power-Line Monitoring with Simultaneous-Sampling ADCs**. *Analog Dialogue 41-01,* 3 p. Janeiro de 2007.

[77] SMITH, R. "**Quick Reference for RS485, RS422, RS232 and RS423**." <www.rs485.com/rs485spec.html> Acesso em 23/07/2007.

[78] SOARES, L.F.G.; LEMOS, G.; COLCHER, S., **Redes de Computadores: Das LAN´s MAN´s e WAN´s às Redes ATM**, Rio de Janeiro: Campus, 705p. 1995.

[79] SOUZA, G.Z. **Desenvolvimento de um Banco de Dados para um Sistema de Monitoramento CIM Focado nas Filosofias MRP e MRPII**. Porto Alegre, 2004. 131p. Trabalho de conclusão de curso (Engenharia de Controle e Automação). Pontifícia Universidade Católica do Rio Grande do sul.

[80] SRINIVASAN, K., LAFOND, C., JUTRAS, R. **A portable integrated power quality measurement system**. IEEE Instrumentation and Measurement Technology Conference, pp. 648-651, Maio de 1993.

[81] STALLINGS, W. **Data and Computer Communications**, 6° edição, USA: Prentice-Hall, 810p. 2000.

[82] STANEK, J. "**Introduction to RS422 & RS485**." Outubro 1998. <http://www.hw.cz/english/docs/rs485/rs485.html>

[83] STERLING JR., D. J.; WISSLER, P., **The Industrial Ethernet Networking Guide**, USA: Thompson Delmar Learning, 331p. 2003.

[84] STONES, J., COLLINSON, A. **Power Quality.** *IEEE Power Engineering Journal*, pp. 58-64, Abril de 2001.

[85] STRANGIO, Christopher E. **The RS232 Stardard: A tutorial with signal names and definitions**. Disponível em <www.camiresearch.com/Data_Com_Basics/RS232_standard.html> Acessado em 24/07/2007.

[86] TSE, N., CHAN, W. L., LAI, L. L. **A Low-Cost Power Quality Meter for Utility and Consumer Assessments.** IEEE International Conference on Eletric Utility Deregulation and Restructuring and Power Tecnologies 2000, pp. 96-101, 4-7 de Abril de 2000.

[87] WON, D.J., CHUNG, I.Y., KIM,J.M., MOON, S.I., SEO, J.C., CHOE, J.W. **Development of Power Quality Monitoring System with Central Processing Scheme**. *IEEE Transaction on industry applications*, Vol. 36, No.5, pp. 915-919, Set/Out 2000.

[88] WONG, Kit Po. **Artificial Intelligence and Neural Network Applications on Power System.** IEE 2nd International Conference on Advances in Power system Control, Operation and Management, pp. 37-45, Dez. 2003.

[89] YARBOROUGH, T., CAROLSFELD, R.S. **Using Distributed Power Quality Monitoring for Better Electrical System Management**. *IEEE Transaction on industry applications*, Vol. 36, No.5, pp. 1481-1485, Set/Out 2000.

[90] ZHU, T. X., TSO, K. S., LO, L. K. **Wavelet-Based Fuzzy Reasoning Approach to Power-Quality Disturbance Recognition**. *IEEE Transactions on Power Delivery*, Vol. 19, No.4, pp. 1928-1935, 2004.

APÊNDICE A – Descrição dos Casos de Uso

Caso de Uso:	Acessar a página da Internet
Atores:	Operador
Propósito	Visualizar os do BD pela WEB
Pré-condições:	
Pós-condição:	Página da WEB apresentada
Curso Típico de Eventos	
Ator	**Sistema**
1. Seleciona função abrir página da WEB.	2. Envia o comando ao windows para abrir o navegador da WEB na página do sistema monitorador (SystMonge).

Caso de Uso:	Sair do Software
Atores:	Operador
Propósito	Fechar o software
Pré-condições:	
Pós-condição:	Software fechado
Curso Típico de Eventos	
Ator	**Sistema**
1. Seleciona função fechar software.	2. Se houverem medidores cadastrados, pede confirmação para sair.
3. Usuário escolhe “sim” ou “não” para sair do software.	4. Interpreta a escolha do usuário. 4.1. Sim, interrompe as comunicações ativas e prossegue. 4.2. Não, sai do processo
	5. Fecha conexão com BD.
	6. Fecha porta serial.
	7. Desaloca memória,libera ponteiros e listas.
	8. Fecha a janela principal.

Caso de Uso:	Organizar Janelas
Atores:	Operador
Propósito	Gerenciar as janelas ativas
Pré-condições:	Existirem janelas ativas
Pós-condição:	Janela alvo mostrada
Curso Típico de Eventos	
Ator	**Sistema**
1.Clicou no menu para ver quais janelas estão ativas.	2. Sistema limpa o menu, percorre todas “janelas de medidores” ativas, e as disponibiliza no menu.
3. Operador escolhe qual janela deve receber o foco.	4. Sistema mostra a janela escolhida.

Caso de Uso:	Resetar software (Novo)
Atores:	Operador
Propósito	Fechar o software
Pré-condições:	
Pós-condição:	Software fechado
Curso Típico de Eventos	
Ator	**Sistema**
1. Seleciona função resetar software.	2. Se houverem medidores cadastrados, pede confirmação.
3. Usuário escolhe “sim” ou “não” para resetar o software.	4. Interpreta a escolha do usuário. 4.1. Sim, interrompe as comunicações ativas e prossegue. 4.2. Não, sai do processo.
	5. Fecha conexão com BD.
	6. Fecha porta serial
	7. Deleta todo texto das editboxes e tabelas.
	8. Limpa as listas de ponteiros.
	9. Fecha todas as janelas abertas.
	10. Abre a janela de configuração dos medidores.

Caso de Uso:	Cadastrar Medidores
Atores:	Operador
Propósito	Gerenciar as informações a serem adquiridas
Pré-condições:	
Pós-condição:	Medidores cadastrados e configurados
Curso Típico de Eventos	
Ator	**Sistema**
1. Pede para abrir a janela para o cadastramento dos medidores.	2. Abre a janela para o cadastramento dos medidores.
3. Entra com os parâmetros do medidor, e clica no botão para cadastrar.	4. Sistema verifica se a porta serial e o BD estão ativados. 4.1 Sim, pula para a etapa 7. 4.2 Não, informa o evento, oferece a abertura da janela de configuração do protocolo e sai do processo.
5. Usuário escolhe “sim” ou “não” para a configuração da porta serial e do BD.	6. Interpreta a escolha do usuário. 6.1. Sim, abre primeiro a janela para configuração da porta serial, e depois a do BD. 6.2. Não, sai do processo.
	7. Verifica se os dados estão corretos, e se o endereço ainda não existe. 7.1 Se estiver tudo ok, cadastra o medidor no software, envia as configurações pela porta serial e aguarda a confirmação. 7.2 Se não estiver ok, avisa e sai do processo.
	8. Se o item “Gerar gráfico” estiver selecionado, então criar uma nova janela de gráfico, associando ao medidor um ponteiro para ela.
	9. Criar uma instancia de Medidor (TThread).
	10. Chamar a função para atualizar a janela de medidores.

Caso de Uso:	Salvar cadastro de medidores
Atores:	Operador
Propósito	Armazenar as configurações dos medidores
Pré-condições:	
Pós-condição:	Medidores cadastrados e configurados
Curso Típico de Eventos	
Ator	**Sistema**
1. Solicita para salvar o cadastro dos medidores	2. Abre a janela para receber o nome e o diretório do arquivo.
3. Informa o nome do arquivo e o diretório.	4. Verifica se o arquivo já não existe. 4.1 Sim, pede nome diferente ao arquivo. 4.2 Não, continua.
	5. Imprime medidores cadastros no “memo”.
	6. Salva Strings do “memo” em arquivo “txt”.

Caso de Uso:	Abrir cadastro de medidores
Atores:	Operador
Propósito	Abrir configurações dos medidores
Pré-condições:	
Pós-condição:	Medidores cadastrados e configurados, a partir do arquivo de dados.
Curso Típico de Eventos	
Ator	**Sistema**
1. Solicita para abrir o cadastro dos medidores	2. Sistema verifica se a porta serial e o BD estão ativados. 2.1 Sim, pula para a etapa. 2.2 Não, informa o evento, oferece a abertura da janela de configuração do protocolo e sai do processo.
3. Usuário escolhe “sim” ou “não” para a configuração da porta serial e do BD.	4. Interpreta a escolha do usuário. 4.1. Sim, abre primeiro a janela para configuração da porta serial, e depois a do BD. 4.2. Não, sai do processo.
	5. Abre a janela para receber o nome e o diretório do arquivo.
6. Informa o nome do arquivo e o diretório.	7. Tenta ler o arquivo. 7.1 Tudo OK, continua 7.2 Falha ao abrir o arquivo, informa erro e sai do processo.
	8. Cadastrar os medidores em memória.

Caso de Uso:	Começar consulta aos medidores
Atores:	Operador
Propósito	Gerenciar as informações a serem adquiridas
Pré-condições:	Medidores já cadastrados
Pós-condição:	Processo de consulta em andamento
Curso Típico de Eventos	
Ator	**Sistema**
1. Usuário seleciona os medidores a serem consultados, e começa processo de consulta.	2. Sistema verifica se a porta serial esta ativada 2.1 Sim, continua. 2.2 Não, abre a janela de configuração do protocolo.
	3. Verifica se Banco de dados está configurado. 3.1 Sim, continua. 3.2 Se não, abre janela de configuração do banco de dados.
	4. Dispara contador de cada medidor

Caso de Uso:	Enviar pedido a um medidor, pela porta serial
Atores:	Instancia de medidor
Propósito	Requisitar para algum medidor físico as informações elétricas
Pré-condições:	Medidores cadastrados
Pós-condição:	Pacote enviado pela porta serial
Curso Típico de Eventos	
Ator	**Sistema**
1. Cria uma Thread “Pedido”, passando como parâmetro o ponteiro que aponta para ele próprio.	2. Cria a Thread “Pedido”, e manda executar.

Caso de Uso:	Thread Pedido
Atores:	Instancia de “Pedido”
Propósito	Requisitar para algum medidor físico as informações elétricas
Pré-condições:	Medidores cadastrados
Pós-condição:	Pacote enviado pela porta serial
Curso Típico de Eventos	
Ator	**Sistema**
1. Calcula quantos pedidos devem ser feitos, e a cada pedido qual deve ser o registrador inicial	2. Verifica se a porta serial esta aberta. 2.1 Sim, continua. 2.2 Não, requisita abertura.
	3. Envia pacote pela serial.

Caso de Uso:	Receber pedido
Atores:	Porta serial
Propósito	Receber uma resposta pela porta serial e interpretar.
Pré-condições:	Pacote ter chegado corretamente
Pós-condição:	Vetores das informações elétricas em memória.
Curso Típico de Eventos	
Ator	**Sistema**
1. Porta serial avisa ter chegado informações no buffer.	2. Verifica se a mensagem chegou sem falhas. 2.1 Sim, prossegue. 2.2 Não, notifica erro no recebimento.
	2. Verifica qual instancia de medidor deve receber as informações, e as armazena nela.
	3.Verifica se a opção “plotar” e verdadeira. 3.1 Sim, cria Thread para plotar.
	4. Verifica se a opção “Gravar no banco de dados” e verdadeira. 4.1 Sim, cria Thread para gravar no banco de dados.
	5. Verifica se a opção “Calcular” e verdadeira. 5.1 Sim, cria Thread para calcular os parâmetros de performance..

Caso de Uso:	Visualizar o Cálculo das Medidas Elétricas
Atores:	Sistema
Propósito	Visualizar os dados calculados
Pré-condições:	Dados já armazenados em memória
Pós-condição:	Cálculos mostrados
Curso Típico de Eventos	
Ator	**Sistema**
1. Fez a consulta a um medidor. O sistema interpretou a resposta e armazenou os dados.	2. Verifica se o medidor esta programado para realizar os calculos. 2.1 Sim, continua. 2.2 Não, sai do processo.
	3. Dispara a Thread para calcular os parametros.

Caso de Uso:	Thread Calcular parâmetros de performance
Atores:	Sistema
Propósito	Gerar o grafico
Pré-condições:	Dados já armazenados em memória
Pós-condição:	Gráfico gerado
Curso Típico de Eventos	
Ator	**Sistema**
1. Sistema criou a thread.	2. Verificar o numero de ciclos, pontos por ciclo e canais.
	3. Fazer os cálculos e mostrar na janela apontada pelo ponteiro.
	4. Criar Thread do gráfico THD
	5. Liberar a thread.

Caso de Uso:	Visualizar Gráfico
Atores:	Sistema
Propósito	Visualizar os dados de um determinado medidor do sistema
Pré-condições:	Dados já armazenados em memória
Pós-condição:	Gráfico gerado
Curso Típico de Eventos	
Ator	**Sistema**
1. Fez a consulta a um medidor. O sistema interpretou a resposta e armazenou os dados.	2. Verifica se o medidor esta programado para mostrar o gráfico. 2.1 Sim, continua. 2.2 Não, sai do processo.
	3. Dispara a Thread para plotar.
	4. Manter gráfico em memória

Caso de Uso:	Thread plotar
Atores:	Sistema
Propósito	Gerar o grafico
Pré-condições:	Dados já armazenados em memória
Pós-condição:	Gráfico gerado
Curso Típico de Eventos	
Ator	**Sistema**
1. Sistema criou a thread.	2. Verificar o numero de ciclos, pontos por ciclo e canais.
	3. Fazer o gráfico na janela apontada pelo ponteiro.
	4. Liberar a thread.

Caso de Uso:	Conectar ao Banco de dados
Atores:	Sistema
Propósito	Estabelecer conexão com banco
Pré-condições:	Conexão física com o BD
Pós-condição:	Conexão ativa ao banco de dados
Curso Típico de Eventos	
Ator	**Sistema**
1. Ator requisita conexão com o banco de dados.	2. Sistema verifica conexão com o BD. 2.1 Conectado, sai. 2.2 Não conectado, requisita usuária e senha.
3. Coloca o usuário e a senha	4. Tenta fazer logon no BD 4.1 Se logado, próximo passo 4.2 Não logado, requisitar senha de novo

Caso de Uso:	Gravar no Banco de dados
Atores:	Sistema
Propósito	Estabelecer conexão com banco
Pré-condições:	Conexão física com o BD
Pós-condição:	Conexão ativa ao banco de dados
Curso Típico de Eventos	
Ator	**Sistema**
1. Fez a consulta a um medidor. O sistema interpretou a resposta e armazenou os dados.	2. Verifica se o medidor esta programado para gravar no banco de dados. 2.1 Sim, continua. 2.2 Não, sai do processo.
	3. Dispara a Thread para gravar no banco de dados.

Caso de Uso:	Thread para Gravar no Banco de dados
Atores:	Sistema
Propósito	Gravar no banco de dados
Pré-condições:	Conexão física com o BD
Pós-condição:	Conexão ativa ao banco de dados
Curso Típico de Eventos	
Ator	**Sistema**
1. Sistema criou a thread.	2. Verifica se a conexão com o banco de dados esta ativa. 2.1 Sim, continua. 2.2 Não, acusa erro.
	3. Grava no banco de dados.

Caso de Uso:	Thread para Gravar no Banco de dados
Atores:	Sistema
Propósito	Gravar no banco de dados
Pré-condições:	Conexão física com o BD
Pós-condição:	Conexão ativa ao banco de dados
Curso Típico de Eventos	
Ator	**Sistema**
1. Sistema criou a thread.	2. Verifica se a conexão com o banco de dados esta ativa. 2.1 Sim, continua. 2.2 Não, acusa erro.
	3. Grava no banco de dados.

APÊNDICE B – Diagramas de Seqüência

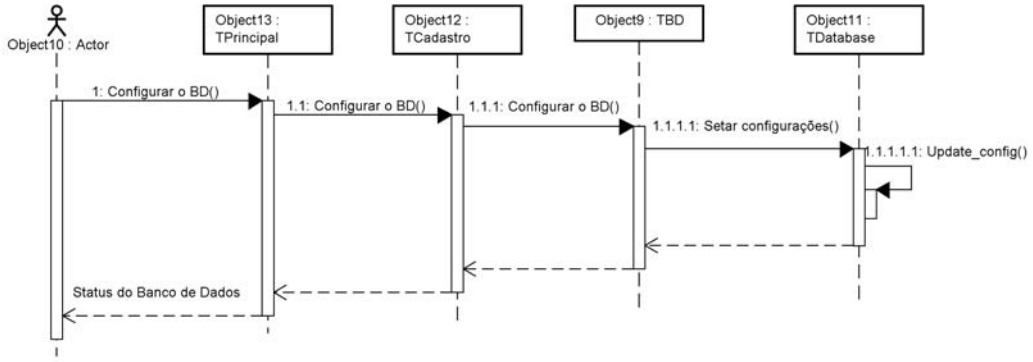


Figura 1. – Configurar banco de dados.

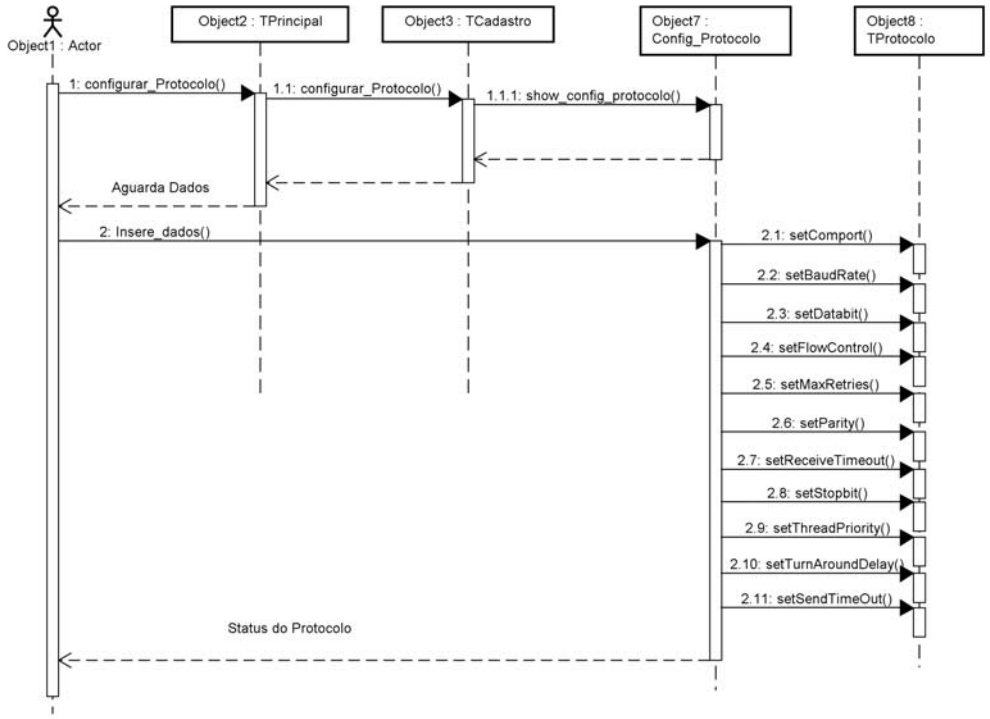


Figura 2. – Configurar protocolo.

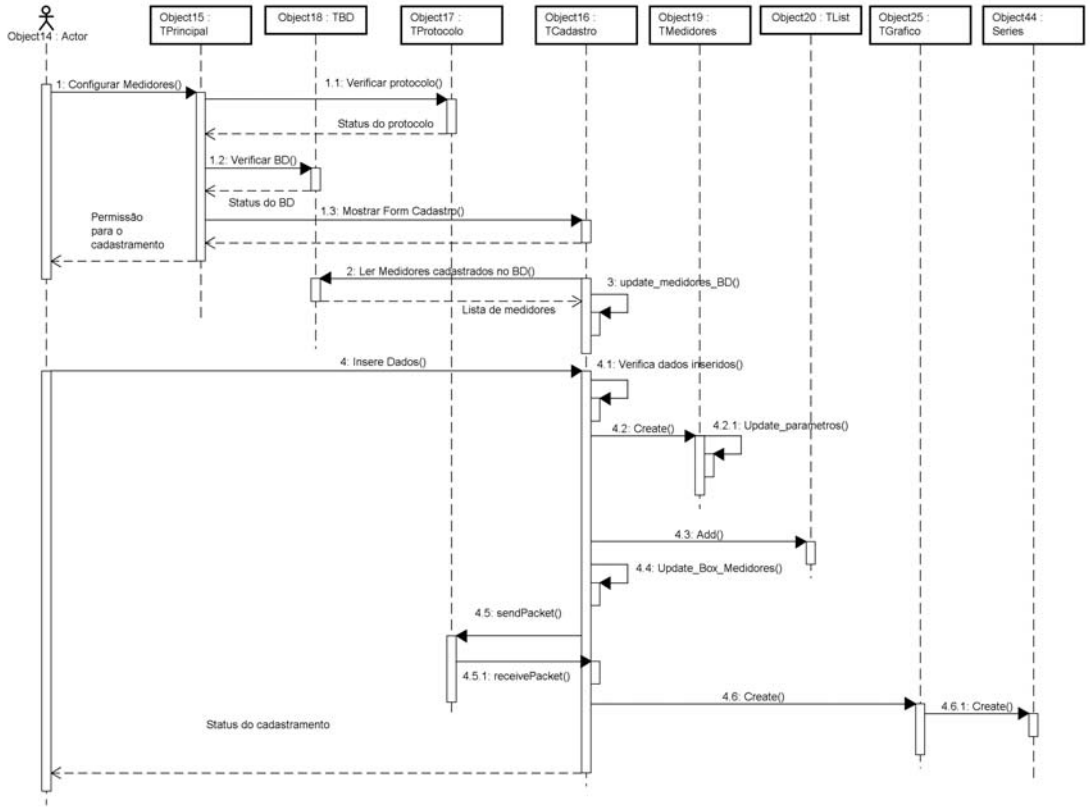


Figura 3. – Cadastrar medidores.

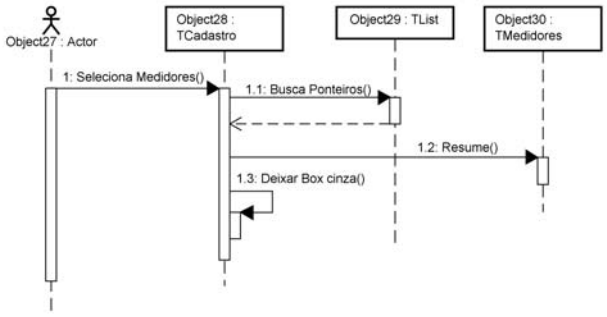


Figura 4. – Começar consulta aos medidores.

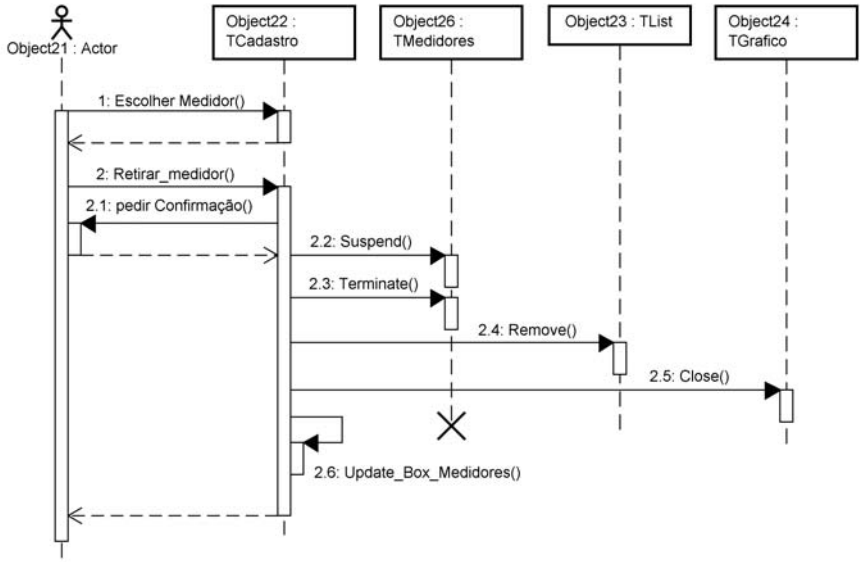


Figura 5. – Retirar medidor da lista.

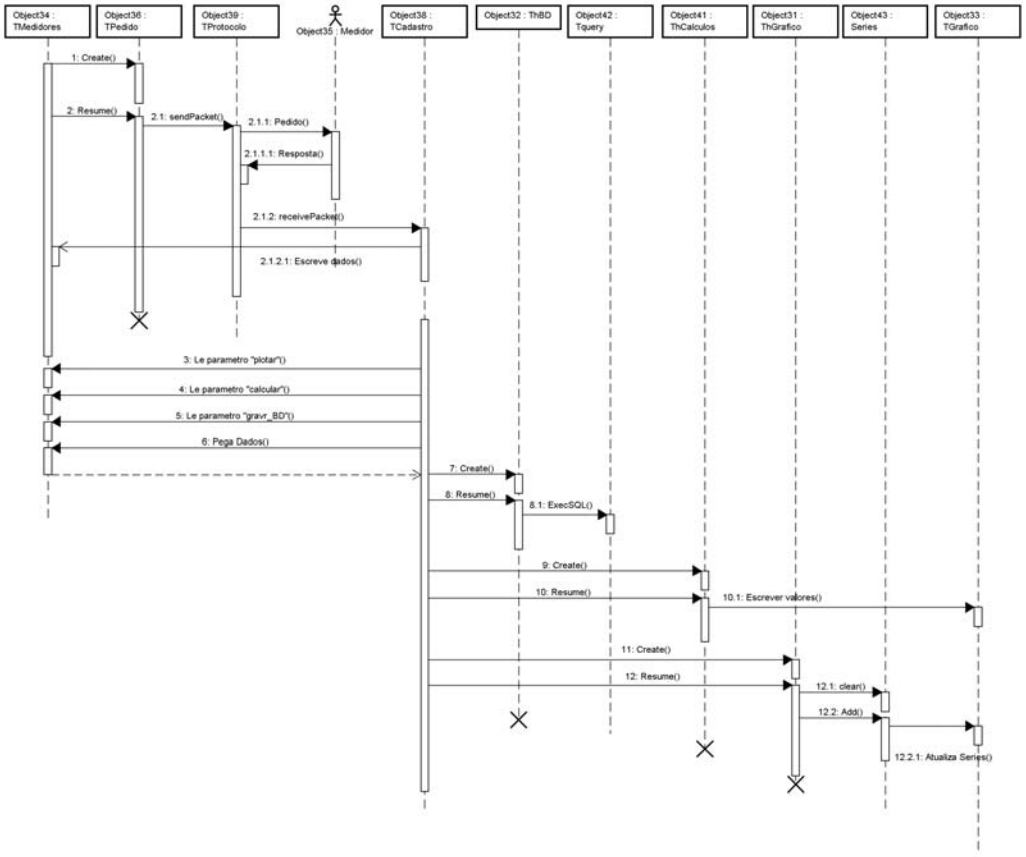


Figura 6. – Adquirir e mostrar dados.

APÊNDICE C - Algoritmos para o Cálculo dos Parâmetros de Performance

Os algoritmos utilizados para o cálculo dos parâmetros de performance foram os tradicionalmente encontrados na literatura de eletrônica de potência, e estão mostrados na tabela abaixo. Mais detalhes sobre as fórmulas podem ser obtidos em [48].

TABELA 1. – Algoritmos utilizados para o cálculo dos parâmetros de performance.

Vrms	$Vrms = \sqrt{\left(\frac{1}{T}\right) \cdot \sum_{n=0}^{j} ((V(n)^2 + V(n+1)^2) \cdot (pc)}$			Active Power	$P = \frac{1}{T}\int_0^T V(t) \cdot I(t) \cdot dt$
Irms	$Irms = \sqrt{\left(\frac{1}{T}\right) \cdot \sum_{n=0}^{j} ((I(n)^2 + I(n+1)^2) \cdot (pc)}$			Reative Power	$Q = \sqrt{S^2 - P^2}$
THDv	$THDv\% = \frac{Vrms(hamonic)}{Vrms(\ fundamental)}$			THDi	$THDi\% = \frac{\sqrt{Irms^2 - Irmsfund^2}}{Irmsfund}$
Harmonic Amplitude	$H = \sqrt{\sum_{i=1}^{f} ((((V(i)2f\sin(2\pi fnt(i)) + (V(i+1)2f\sin(2\pi fnt(i+1)) \cdot (PC))^2 + (V(i)2f\cos(2\pi fnt(i)) + (V(i+1)2f\cos(2\pi fnt(i+1)))/2) \cdot (PC))^2}$				
Power Factor	$FP = \frac{P}{S}$	Fator de Deslocamento	$FD = \cos(\phi)$	Apparent Power	$S = VI$

A seguir, está mostrado o código fonte do arquivo “DLL”, que contém os algoritmos de cálculo.

```
function Calcula_rms(vet : array of single; ppc : word;
                     ciclos : byte; freq : single) : single; stdcall;
var
 Acum : Single; i : word;
begin
 Acum := 0;
 for i := 0 to (ciclos*(ppc-1))-1 do
  Acum := Acum + ((power(vet[i],2) + power(vet[i+1],2))/2)*(1/freq/(ppc-1));
 Result := sqrt((freq/ciclos)*Acum);
end;
-------------------------------------------------------------------------------
function Calcula_PAparente(Trms, Crms : single) : single; stdcall;
begin
 Result := Trms*Crms;
end;
```

```
function Calcula_FD(ten, corr : array of single; ppc : word;
                    ciclos : byte; freq : single) : single; stdcall;
var
 i : word;
 passo,ang1,ang2,fi,tm,Ra,Rf : single;
//-------Local function-----------------começa-----//
function fase(pt1,pt2 : single) : single;
var
 M,Ra1,Ra2,Rf1,Rf2 : single;
begin
 Ra1 := pt2*(2*freq)*sin(1*(2*freq)*pi*(tm+passo));
 Ra2 := pt1*(2*freq)*sin(1*(2*freq)*pi*tm);
 Rf1 := pt2*(2*freq)*cos(1*(2*freq)*pi*(tm+passo));
 Rf2 := pt1*(2*freq)*cos(1*(2*freq)*pi*tm);
 Ra := Ra + ((Ra1 + Ra2)/2)*passo;
 Rf := Rf + ((Rf1 + Rf2)/2)*passo;
 M := arctan2(Rf,Ra);
 result := M;
end;
//-------Local function-----------------termina-----//
begin
 passo := (1/freq)/ppc;
 ang1 := 0;  ang2 := 0;
 Ra := 0; Rf := 0; tm := 0;
 for i := 0 to (ppc-2) do
 begin
  ang1 := fase(ten[i],ten[i+1]);
  tm := tm + passo;
 end;
 Ra := 0; Rf := 0; tm := 0;
 for i := 0 to (ppc-1) do begin
  ang2 := fase(corr[i],corr[i+1]);
  tm := tm + passo;
 end;
 fi := ang2 - ang1;
 result := cos(fi);
end;
--------------------------------------------------------------------------------
function Calcula_PAtiva(ten, corr : array of single; ppc : word;
                    ciclos : byte; freq : single) : single; stdcall;
var
 i : word;  acum, passo : single;  produto : array of single;
begin
 acum := 0;
 passo := ((1/freq)/(ppc-1))/ciclos;
 setlength(produto,length(ten));
 for i := 0 to length(ten)-1 do
  produto[i] := ten[i]*corr[i];
 for i := 0 to length(produto)-1 do
  acum := acum + ((produto[i]+produto[i+1])/2)*passo;
 Result := Acum*freq/ciclos;
end;
-----------------------------------------------------------------
function Calcula_FP(Pativa, PAparente : single) : single; stdcall;
begin
 Result := Pativa/PAparente;
end;
```

```
function Calcula_Distorcao(vet : array of single; ppc : word;
            ciclos : byte; freq, Valor_Rms : single) : single; stdcall;
var
 harmo,tm,passo,Ra,Rf,R : single;  i : word;
//-------Local function-----------------começa-----//
function Arm(pt1,pt2 : single) : single;
var
 M,Ra1,Ra2,Rf1,Rf2 : single;
begin
 Ra1 := pt2*(2*freq)*sin(1*(2*freq)*pi*(tm+passo));
 Ra2 := pt1*(2*freq)*sin(1*(2*freq)*pi*(tm));
 Rf1 := pt2*(2*freq)*cos(1*(2*freq)*pi*(tm+passo));
 Rf2 := pt1*(2*freq)*cos(1*(2*freq)*pi*(tm));
 Ra := Ra + ((Ra1 + Ra2)/2)*passo;
 Rf := Rf + ((Rf1 + Rf2)/2)*passo;
 M := sqrt(power(Ra,2) + power(Rf,2));
 result := M;
end;
//-------Local function-----------------termina-----//
begin
 Ra := 0; Rf := 0; tm := 0; Harmo := 0;
 passo := (1/freq)/(ppc-1);
 for i := 0 to (ppc-2) do
 begin
  harmo := arm(vet[i],vet[i+1]);
  tm := tm + passo;
 end;
 R := harmo/sqrt(2);
 Result := (sqrt(power(Valor_Rms,2) - power(R,2))/R)*100;
end;
-------------------------------------------------------------------------------
function Calcula_harmonicas(vet : array of single; ppc : word;
          ordem_harmo, ciclos : byte; freq: single) : single; stdcall;
var
 i : word;
 tm, passo, Ra, Rf : single;  arr1,arr2,arr3,arr4 : array of single;
begin
 setlength(arr1,length(vet)+1);
 setlength(arr2,length(vet)+1);
 setlength(arr3,length(vet)+1);
 setlength(arr4,length(vet)+1);
 Ra := 0;   Rf := 0;   tm := 0;
 passo := (1/freq)/ppc;
 for i := 0 to (ppc-2) do
 begin
  arr1[i] := sin(ordem_harmo*2*pi*freq*tm);
  arr2[i] := cos(ordem_harmo*2*pi*freq*tm);
  arr3[i] := Vet[i]*freq*2*arr1[i];
  arr4[i] := Vet[i]*freq*2*arr2[i];
  tm := tm + passo;
  arr1[i+1] := sin(ordem_harmo*2*pi*freq*tm);
  arr2[i+1] := cos(ordem_harmo*2*pi*freq*tm);
  arr3[i+1] := Vet[i+1]*freq*2*arr1[i+1];
  arr4[i+1] := Vet[i+1]*freq*2*arr2[i+1];
   Ra := Ra+((arr3[i]+arr3[i+1])/2)*passo;
  Rf := Rf+((arr4[i]+arr4[i+1])/2)*passo;
 end;
 Result := sqrt(power(Ra,2) + power(Rf,2));
end;
```

APÊNDICE D – Programação e Utilização da Memória SRAM e da FPGA

Apêndice D.1 - Projeto Lógico da FPGA

Como explicado no capítulo 4, para o medidor adquirir mais de um ciclo de tensão e corrente trifásicos, fez-se necessário a utilização de uma memória externa. Mas para o armazenamento das informações nessa memória algumas novas tarefas seriam atribuídas ao micro-controlador, como o endereçamento, a leitura e a escrita das partes alta e baixa do dado.

De forma a se utilizar poucos pinos do micro-controlador e ser um hardware mais "compacto", ao invés da utilização de "latches" externos, elaborou-se uma lógica em "FPGA" (EPM3064). Abaixo estão mostradas algumas figuras do ambiente de desenvolvimento, do software "QuartusII", da "Altera".

Pode-se observar na Figura 1 o esquema lógico para o funcionamento da "FPGA", sendo que a Figura 2 mostra os componentes que compõe o "latch-18", e a Figura 3 o circuto de controle e proteção.

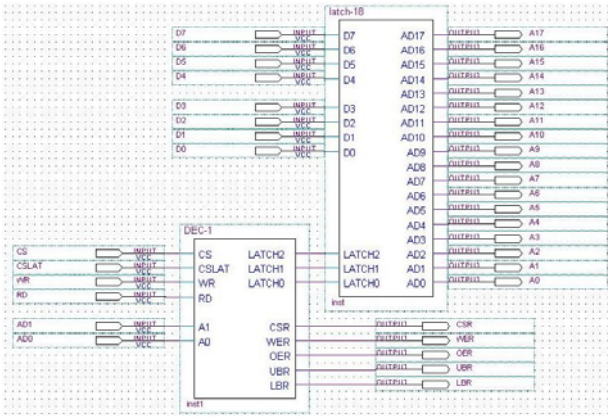


Figura 1. - Módulo de proteção e de "latches".

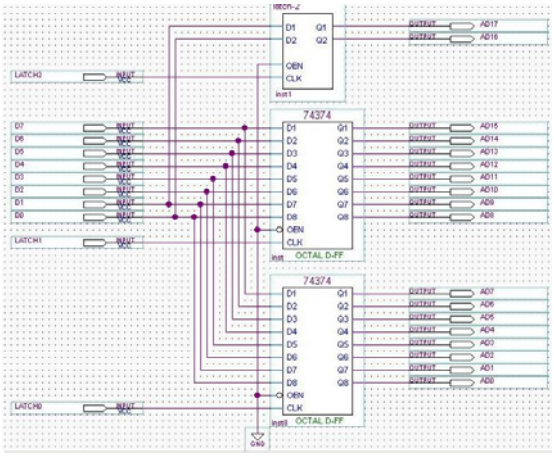


Figura 2. – “Latches”.

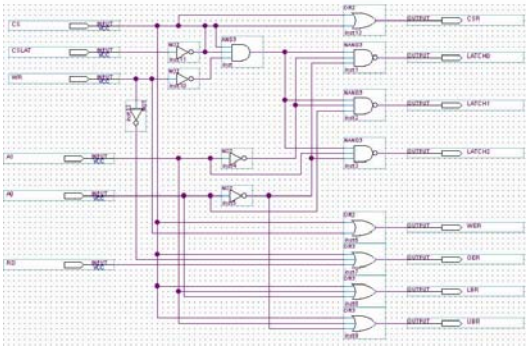


Figura 3. – Circuito de controle e proteção proteção.

A Figura 4 mostra uma simulação de um endereçamento para a leitura ou escrita em uma posição da memória “SRAM”.

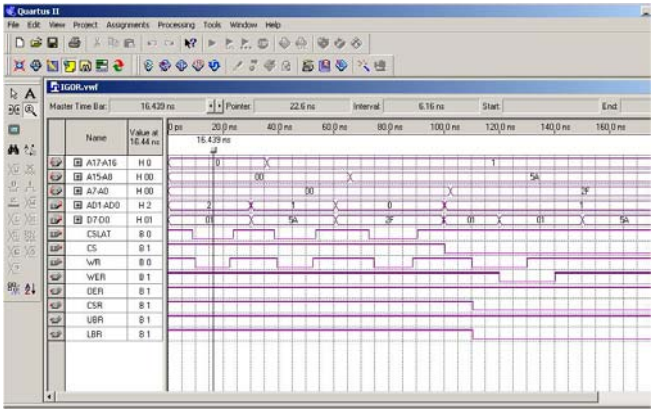


Figura 4. – Simulação

Apêndice D.2 – Algoritmos para Leitura e escrita na memória SRAM

O quadro abaixo está mostrando os algoritmos para o endereçamento na memória, e leitura e escrita de dados na mesma. Pode-se observar que a porta “P4” está sendo utilizada para o controle do processo, assim como a “P5” para a escrita e leitura dos dados.

Também é interessante se observar que a porta “P5” deve mudar de direção, ou seja, deve ser configurada para saída no momento de uma escrita, e de entrada para realizar uma leitura. Ainda se pode observar que na função de leitura, a informação é colocada diretamente no vetor que será enviado pela porta serial.

```
void enderecaSRAM(unsigned char part_mem,unsigned int end_mem)
{
 unsigned char end_hi,end_lo;   end_hi = parte_alta(end_mem); end_lo =parte_baixa(end_mem);
 P5DIR = 0XFF; // PORTA DE SAÍDAS
 P5OUT = part_mem; // A17 E A16
 P4OUT = 0XDB; // MUDA ESTADO DE CSLAT DE 1 PARA 0
 P4OUT = 0XDA; // MUDA ESTADO DE WR DE 1 PARA 0
 P4OUT = 0XD7; // MUDA ESTADO DE AD0 DE 0 PARA 1 E AD1 DE 1 PARA 0
 P5OUT = end_hi; // ENDEREÇO ALTO
 P4OUT = 0XD6; // MUDA ESTADO DE WR DE 1 PARA 0
 P4OUT = 0XD3; // MUDA ESTADO DE AD0 DE 1 PARA 0
 P5OUT = end_lo; // ENDEREÇO BAIXO
 P4OUT = 0XD2; // MUDA ESTADO DE WR DE 1 PARA 0
 P4OUT = 0XFF; // TERMINA FUNÇÃO DE ENDEREÇAMENTO
}
=====================================================
void escritaSRAM(unsigned char dado_hi,unsigned char dado_lo)
{
 P5DIR = 0XFF; // PORTA DE SAÍDAS
 P5OUT = dado_hi;
 P4OUT = 0XE7; // MUDA ESTADO DE AD1 DE 1 PARA 0
 P4OUT = 0XE6; // MUDA ESTADO DE WR DE 1 PARA 0
 P4OUT = 0XE3; // MUDA ESTADO DE AD0 DE 1 PARA 0
 P5OUT = dado_lo;
 P4OUT = 0XE2; // MUDA ESTADO DE WR DE 1 PARA 0
 P4OUT = 0XFF; // TERMINA FUNÇÃO DE ESCRITA
}
=====================================================
void lerSRAM(unsigned int posicao)
{
 P5DIR = 0X00; // PORTA DE ENTRADAS
 P4OUT = 0XE7; // MUDA ESTADO DE AD1 DE 1 PARA 0
 P4OUT = 0XE5; // MUDA ESTADO DE RD DE 1 PARA 0
 serial_out[posicao+3] = P5IN;
P4OUT = 0XE3; // MUDA ESTADO DE AD0 DE 1 PARA 0
 P4OUT = 0XE1; // MUDA ESTADO DE RD DE 1 PARA 0
 serial_out[posicao+4] = P5IN;
P4OUT = 0XFF; // TERMINA FUNÇÃO DE LEITURA
}
```

ANEXO A – O Protocolo Modbus

Anexo A.1 - Introdução

O protocolo Modbus foi desenvolvido para comunicar um dispositivo mestre com outros dispositivos escravos, fazendo-se uso de uma estrutura de mensagens compostas por bytes, sendo muito utilizado em automação industrial devido à sua simplicidade e facilidade de implementação. Esta comunicação pode utilizar diversos meios físicos, como a rede RS485, que é a utilizada neste trabalho, a ethernet (TCP/IP), MAP e etc [2],[54]. As Figuras 1 e 2 exemplificam a arquitetura Modbus.

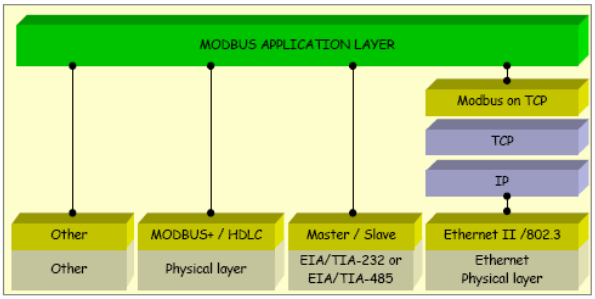


Figura 1. - Camadas Modbus.

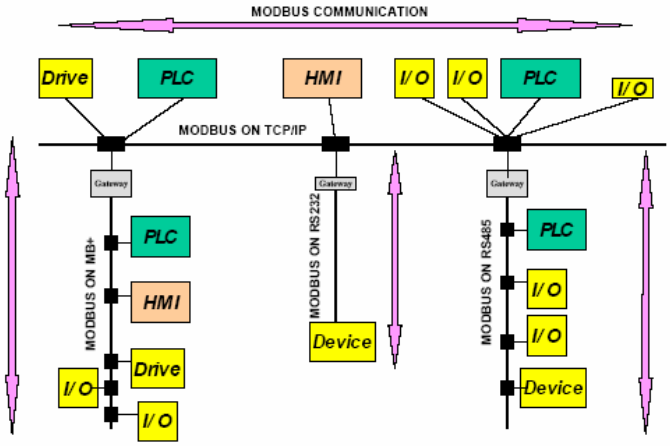


Figura 2. - Exemplo de arquitetura Modbus

A comunicação pode ser iniciada somente pelo mestre, a qualquer momento, independentemente do estado do escravo. Este ultimo, só realiza uma função sob a solicitação e controle do mestre. A mensagem pode ser endereçada a somente um escravo, ou ser “broadcast”, com o endereço especificado em “zero”, onde neste caso, os escravos não devem responder nada.

O protocolo basicamente determina como cada dispositivo vai identificar uma mensagem endereçada a ele, determinar o tipo de ação a ser executada, obter a informação e respondê-la, mesmo que seja uma resposta de “pedido inválido”, “erro na comunicação” ou “requerido mais tempo para processar” (exceptions) [75**],**[53**],**[54].

Anexo A.2 - Modos de Transmissão

Dois modos de transmissão são possíveis no modbus, e definem como as mensagens serão empacotados: ASCII e RTU. Neste trabalho foi utilizado o segundo modo (sendo somente ele descrito), devido à maior densidade de dados por mensagem, representando um melhor desempenho [2].

A quantidade de bits por byte é de 11, devido ao “startbit”, “stopbit” e “bit de paridade”. Todos os dispositivos devem estar com a mesma configuração de paridade. Caso esta seja “sem paridade”, o “bit de paridade” não é transmitido, mas no lugar dele vai um “stopbit” adicional [53].

O “framing” da mensagem funciona sem caracteres indicativos de início e fim, e sim tempos “vazios”, ou seja, uma mensagem é dada como terminada após um tempo de 3,5 caracteres em silêncio (este tempo é calculado de acordo com a taxa de transmissão). Isso quer dizer que se um byte estiver chegando a um tempo maior que 3,5 caracteres, ele é considerado como o início de um outro frame.

Assim como a divisão da mensagem é feita por tempos “vazios”, a divisão interna entre os bytes também. Entre cada byte deve haver até 1,5 caracteres de “silêncio”. Caso um intervalo maior que este for observado antes do final do frame, a mensagem deve ser descartada [53],[54]. A Figura 3 exemplifica a separação de frames e bytes.

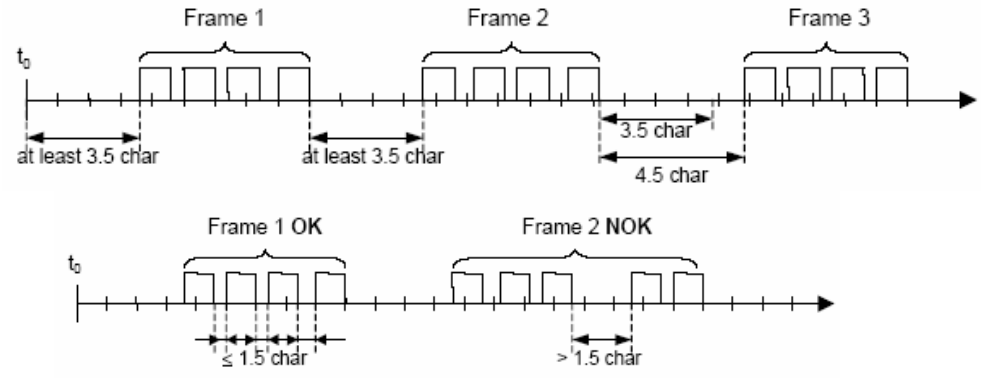


Figura 3. - Tempos de silêncio, na transmissão no formato RTU.

A composição do frame se dá de no máximo 256 bytes: um byte de endereço do dispositivo, um do código da função, até 252 bytes de dados e dois de checagem de erro (CRC – cyclical redundancy checking), mostrado na Figura 4.

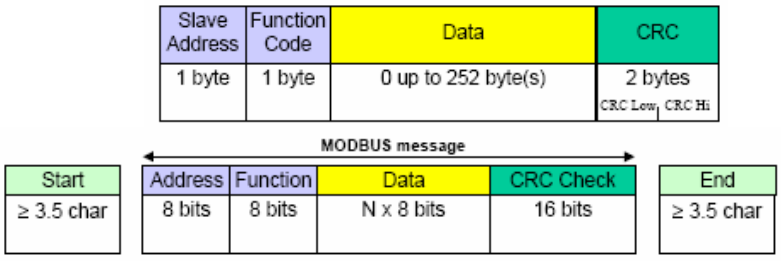


Figura 4. - Descrição do frame no modo RTU.

Os campos da mensagem funcionam da seguinte forma:

- Endereço – Composto por um byte, pode conter um valor decimal de no máximo 247, e é destinado a endereçar a mensagem. Caso o escravo realize com sucesso a função, ele coloca o seu próprio endereço na resposta.

- Função – Campo utilizado para a informação de que função o escravo deve cumprir. Caso a função tenha sido realizada, o número da função será o mesmo na resposta. Caso tenha havido insucesso na realização da função, o código desta é devolvido com o bit mais significativo alto. Neste caso, é de responsabilidade do mestre gerenciar e analisar mensagens de “exceptions”, reenviando o pedido que originou o erro ou uma função diagnóstico para identificar melhor a causa do erro.

- Dados – Caso não haja erros na execução da função, neste campo serão retornados informações pertinentes ao comando solicitado. Entretanto, se houver erro, este campo conterá o código da “exception”.

- Checksum (CRC para modo RTU) – Este campo é destinado a um valor calculado em cima dos dados, que tem a função de verificar a integridade da mensagem. Quando um dispositivo vai mandar uma mensagem, ele antes calcula o CRC, para este ir junto à mensagem. Quem a recebe, deve recalcular o CRC e verificar com o valor mandado. Caso sejam iguais, os dados são válidos. Caso contrário, houve falha na transmissão.

Anexo A.3 - Funções Modbus

Existem 3 categorias de funções do Modbus. São elas:

- Funções públicas: São funções já definidas, documentadas e sem possibilidade de mudanças por parte do usuário. Vão do número 1 a 65, tendo neste intervalo números reservados para funções a serem implementadas.

- Funções definidas pelo usuário: Vão do número 65 ao 72, e do 100 ao 110, e servem para o usuário implementar a sua própria função.

- Funções reservadas: São funções implementadas por empresas (que compraram esse direito) para comercializar produtos, e por isto não são funções públicas.

A Figura 5 mostra a organização da numeração das funções.

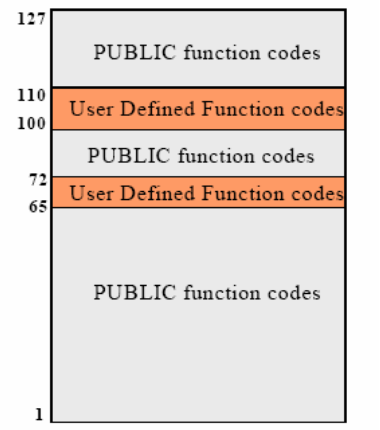


Figura 5. - Numeração das funções Modbus.