UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA

Josemar Barrêto Júnior

SISTEMA DE AQUISIÇÃO, MONITORAMENTO E PERSISTÊNCIA DE DADOS ORIUNDOS DE UMA REDE DE SENSORES SEM FIO PARA SUPERVISÃO DA EFICIÊNCIA ENERGÉTICA EM MOTORES INDUSTRIAIS.

João Pessoa
2011

Josemar Barrêto Júnior

SISTEMA DE AQUISIÇÃO, MONITORAMENTO E PERSISTÊNCIA DE DADOS ORIUNDOS DE UMA REDE DE SENSORES SEM FIO PARA SUPERVISÃO DA EFICIÊNCIA ENERGÉTICA EM MOTORES INDUSTRIAIS.

Monografia apresentada como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação, pelo curso de Ciência da Computação da Universidade Federal da Paraíba.

Orientador: Hamilton Soares da Silva
Coordenador de Estágio: Hamilton Soares da Silva

João Pessoa
2011

Josemar Barrêto Júnior

SISTEMA DE AQUISIÇÃO, MONITORAMENTO E PERSISTÊNCIA DE DADOS ORIUNDOS DE UMA REDE DE SENSORES SEM FIO PARA SUPERVISÃO DA EFICIÊNCIA ENERGÉTICA EM MOTORES INDUSTRIAIS.

Monografia apresentada como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação, pelo curso de Ciência da Computação da Universidade Federal da Paraíba.

Habilitação: Bacharelado em Ciência da Computação

Data de aprovação: ____/____/____

Banca Examinadora:

_____

Professor Hamilton Soares da Silva
Orientador
Universidade Federal da Paraíba

_____

Professora Tatiana Aires Tavares
Universidade Federal da Paraíba

_____

Professor Ed Porto Bezerra
Universidade Federal da Paraíba

Dedicatória

À meus pais e irmãs, pela educação e apoio em todos os momentos de minha vida. À Maria Amélia, pela dedicação, companheirismo e incentivo.

AGRADECIMENTOS

Agradeço a meu orientador Hamilton Soares da Silva por compartilhar seus conhecimentos seja dentro ou fora da sala de aula, sempre de forma altruísta.

Agradeço também ao professor Francisco Antônio Belo, pela oportunidade de trabalhar lado a lado e adquirir conhecimentos valiosos para minha formação.

Agradeço a todos os professores do Departamento de Informática da UFPB pela competência e dedicação no momento de ensinar, formando assim excelentes profissionais nos quais eu poderei, com orgulho, me incluir.

Agradecimento especial a todos os membros do laboratório de energia solar, em especial Ruan Delgado Gomes e Abel Cavalcante Lima Filho, pela amizade e contribuição dada tanto para este trabalho quanto para a minha formação.

“A imaginação frequentemente nos leva para mundos que nunca existiram. Mas sem ela vamos a lugar nenhum.”

(Carl Sagan)

RESUMO

A grande maioria das indústrias usam sistemas mecânicos impulsionados por motores elétricos. Tais sistemas são responsáveis por até dois terços do consumo energético, muito embora funcionem com cerca de apenas 60% de sua eficiência, causando um desperdício de energia. Desse modo, um sistema de monitoramento da eficiência energética de motores se destaca como um importante fator estratégico para as indústrias. A tecnologia atual permite que sistemas de supervisão usando redes de sensores sem fio tornem-se uma alternativa viável aos métodos de monitoramento mais comuns. No entanto, tais sistemas exigem um subsistema de aquisição e exibição de dados para efetivo controle. Neste trabalho foi desenvolvido um software capaz de adquirir dados oriundos de uma rede de sensores sem fio criada para obter informações de eficiência energética e torque em motores industriais. O sistema é capaz de exibir os dados em gráficos e armazená-los em um banco de dados relacional. O trabalho também descreve testes que foram realizados com o software, a fim de garantir que o mesmo seja capaz de receber os dados na taxa máxima enviada pela rede de sensores, sem erros.

Palavras-chave: Eficiência Energética, Sistemas de Aquisição de Dados, Redes sem fio, Java.

Lista de Figuras

SUMÁRIO

1. INTRODUÇÃO

1.1 Motivação

O uso racional e eficiente de energia tem sido uma preocupação constante em diversos setores da sociedade atual, por motivos que vão desde o impacto ambiental à redução de custos. Esta questão é ainda mais importante para as indústrias, uma vez que o uso eficiente de recursos é um fator estratégico altamente relevante para a sobrevivência das mesmas em um meio competitivo.

A grande maioria dos processos de produção de um ambiente industrial usam sistemas mecânicos impulsionados por motores elétricos. Tais sistemas são responsáveis por pouco mais de dois terços do consumo de energia elétrica. Geralmente, apenas motores elétricos de 500 HP ou mais são monitorados, devido ao seu alto custo. Porém os motores de capacidade inferior correspondem a cerca de 99,7% dos motores em serviço. Em média, tais motores operam com até 60% da sua carga nominal, devido às instalações superdimensionadas ou condições de baixa carga, resultando assim em energia desperdiçada. (LU et al, 2008). Portanto, sistemas de automação industrial inteligentes e de baixo custo para monitorar a eficiência energética em motores de porte pequeno e médio podem despontar como uma alternativa interessante para as industrias otimizarem o uso de sua energia elétrica.

Um sistema adequado de monitoramento dos motores, por aumentar a eficiência do uso de energia elétrica, também trará um retorno rápido de investimento. Por exemplo, se um motor de 250 CV estiver subdimensionado e operando com eficiência de 88%, a substituição do mesmo por um motor de 100 CV, operando com rendimento de 94%, permitiria um retorno de investimento em apenas dois meses (WEG, 2010).

O monitoramento de energia e detecção de falhas em sistemas industriais tem sido feito, tradicionalmente, de forma *off-line* ou com sistemas com fio constituídos de vários cabos de comunicação e diversos tipos de sensores. A instalação e manutenção destes cabos e sensores são, normalmente, muito mais caras que o próprio custo dos sensores (LU, 2009). Somado ao custo alto tem-se a falta de flexibilidade, já que os cabos dificultam a instalação e manutenção destes sistemas de monitoramento.

Uma alternativa promissora a tais sistemas reside nas redes de sensores sem fio. Os avanços tecnológicos na área de comunicações sem fio e na eletrônica têm permitido a implementação de sensores e atuadores multifuncionais, de baixo custo e baixo consumo de energia (LU, 2009). As redes de sensores sem fio industriais trazem diversas vantagens sobre

os sistemas tradicionais de monitoramento, incluindo auto-organização, rápida implantação, flexibilidade e capacidade inerente de processamento inteligente (GUNGOR, 2009).

No entanto, um sistema de monitoramento wireless completo necessita de um subsistema de aquisição e processamento dos dados capturados. Na indústria, é comum o uso de sistemas SCADA (Supervisory Control And Data Acquisition), que além de ser responsável por coletar dados, realizar quaisquer análises necessárias e então exibir a informação para certo número de telas de operação ou displays (WRIGHT, BAILEY, 2003), também realiza a tarefa de operar o sistema. Tais sistemas são bastante heterogêneos, tendo suas características muitas vezes ditadas por sua aplicação.

Neste trabalho será apresentado um sistema de aquisição, exibição e persistência de dados oriundos de uma rede de sensores sem fio para monitoramento em tempo real da eficiência energética em motores elétricos de ambientes industriais. A rede de sensores que serve de base para este trabalho utiliza o padrão Zigbee/IEEE 802.15.4 para medida para medida e estimação de diferentes parâmetros dos motores como: consumo, eficiência energética e torque. Os parâmetros serão estimados utilizando valores obtidos a partir de sensores de corrente, tensão e velocidade. A rede será formada por nós inteligentes, que realizarão o processamento local dos dados provenientes dos sensores para estimação dos parâmetros desejados.

O sistema descrito neste trabalho possui três módulos com funções distintas. O módulo de aquisição tem por função receber os dados em tempo real da rede sem fio, identificando valores como o endereço do motor de origem, sua eficiência, consumo e torque, transformando este conjunto de valores em um pacote para os outros módulos do sistema. O módulo de visualização deverá permitir ao usuário a visualização dos dados em forma de um gráfico independente para cada valor adquirido. O último módulo tem por função armazenar todos os dados adquiridos em um banco de dados relacional, se o usuário assim desejar.

1.2 Objetivos

O objetivo geral do trabalho consiste no projeto e implementação de um sistema que permita o monitoramento de dados capturados em tempo real de uma rede de sensores sem fio. O sistema deve tratar os dados adquiridos, exibi-los de forma inteligível para o usuário e armazenar os mesmos em um banco de dados relacional. O sistema também deve

ser robusto e resistente a falhas. A perda de dados, comum a sistemas de aquisição em tempo real, deverá ser minimizada ao máximo, deixando-a sempre em um patamar aceitável.

Os objetivos específicos do trabalho são:

- Desenvolver o módulo de aquisição e tratamento dos dados oriundos da rede de sensores sem fio;
- Desenvolver o módulo de interface, responsável por toda a interação entre o usuário e o sistema;
- Desenvolver o módulo de persistência, utilizando um banco de dados relacional.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo descrever todos os conceitos teóricos e práticos necessários para o desenvolvimento do sistema.

2.1 Redes de Sensores Sem Fio

Nesta última década, o grande avanço tecnológico nas áreas de sensores, circuitos integrados e comunicação sem fio permitiu o desenvolvimento de microdispositivos de baixo custo, com poder de processamento e capacidade de comunicação a curtas distâncias. O desenvolvimento desta tecnologia levou ao surgimento das redes de sensores sem fio (RSSF), que são redes formadas por dispositivos munidos de sensores com capacidade de comunicação via radiofrequência. Os sensores destes dispositivos são construídos de forma a produzir respostas às mudanças de variáveis físicas como temperatura, umidade ou campo magnético. O objetivo principal das RSSF é, em geral, realizar monitoramento de ambientes e localização de objetos. Estas redes formam um novo tipo de redes ad hoc com um conjunto de características e desafios peculiares (LOUREIRO, 2003) (AKYILDIZ, 2010).

O surgimento dos sensores inteligentes foi o principal responsável pelo aumento de interesse nas RSSF dos últimos anos. Normalmente, trata-se de dispositivos pequenos, com restrições de recurso e de baixo custo. Estes sensores são capazes de coletar dados do ambiente e passa-los adiante através de radiofrequência. Porém, sensores equipados com processadores são capazes de gerar novos dados, indo além da tarefa de mero recolhedor de dados do ambiente observado e diminuindo o tráfego na rede. As informações obtidas são repassadas para um nó sorvedouro (também chamado de controlador ou monitor), podendo passar por nós intermediários. Este nó controlador pode ser usado localmente ou através de outras redes (por exemplo, Internet), por meio de um gateway. Os nós devem ser dotados da capacidade de auto-organização, pois os mesmos podem possuir mobilidade. Outros aspectos que influem na dinâmica da topologia da rede podem ser citados, como obstruções no ambiente e interferências na faixa de espectro utilizada para comunicação (AKYILDIZ, 2010) (VERDONE, 2007) (YICK, 2010).

As RSSFs, em oposto às redes tradicionais, possuem recursos escassos, como: fonte de energia limitada (alimentação via baterias), baixo alcance de comunicação, baixa

largura de banda e capacidade restrita de processamento e armazenamento. Tais restrições devem ser levadas em conta ao se desenvolver tecnologias e protocolos para tais tipos de rede (AKYILDIZ, 2010) (YICK, 2010).

As RSSFs geralmente possuem pouca ou nenhuma infraestrutura. Uma RSSF não estruturada é implantada de forma ad hoc e exige auto-organização. Por outro lado, em uma RSSF estruturada, todos os nós ou parte deles são implantados de forma pré-planejada. As restrições de projeto de uma RSSF são normalmente dependentes da aplicação e baseadas no ambiente a ser monitorado. O ambiente monitorado é uma peça chave na determinação do tamanho da rede, do esquema de implantação e da topologia. Implantação ad hoc é preferível quando o local de implantação é de difícil acesso ou quando a rede é composta por uma vasta quantidade de nós (YICK, 2010).

As aplicações das RSSF são classificadas geralmente em aplicações de monitoramento (monitoring) e aplicações de rastreamento (tracking). A Figura 1 apresenta exemplos de aplicações de RSSF nas duas categorias.

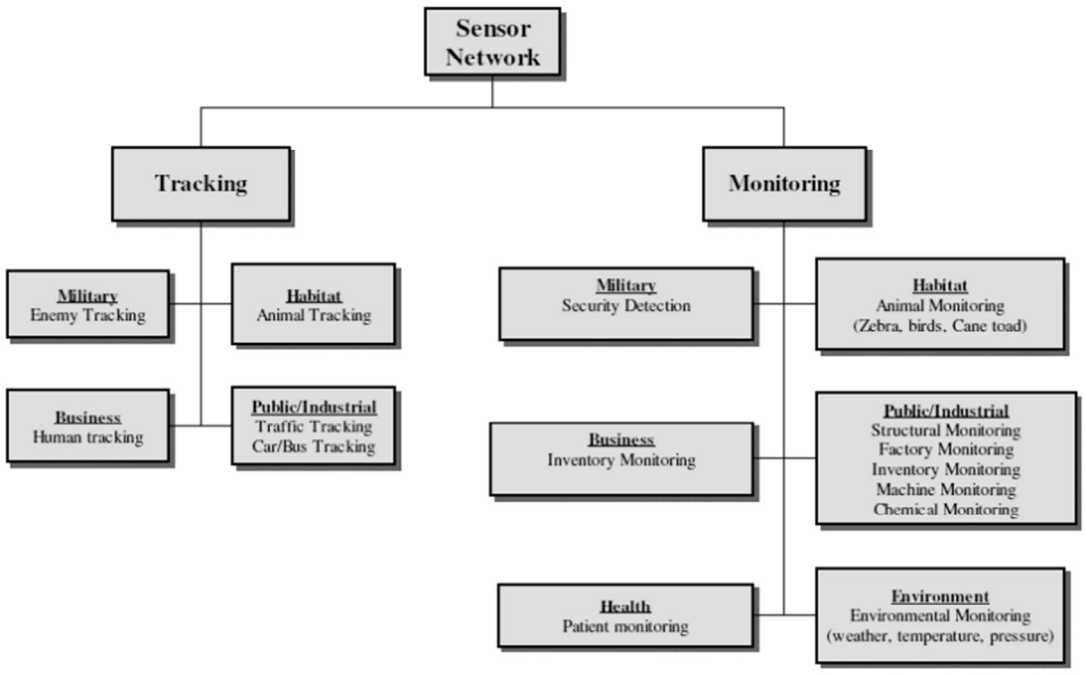


Figura 1 – Aplicações de RSSF.

As aplicações de RSSF possuem um conjunto de atributos que determinam o rumo das pesquisas em RSSF. Aplicações específicas como o monitoramento ambiental, monitoramento de saúde e monitoramento industrial, possuem características e requerimentos

específicos que devem reger a implantação da rede de sensores. Em geral, existem características-chave que precisam ser asseguradas pelas RSSFs como, por exemplo, segurança, robustez, confiabilidade, taxa de transferência adequada e determinismo. Dentre essas características, a falta de confiabilidade é a principal razão para muitos usuários não implantarem equipamentos sem fio em geral. A preocupação com confiabilidade não diz respeito apenas a falhas nos equipamentos, mas também com a confiabilidade da transmissão e recepção dos dados (YICK, 2010) (FETTE, 2011). Muito dessa preocupação está relacionada à interferência no espectro de comunicação das redes sem fio. As redes sem fio sofrem com interferência tanto interna, ou seja, causada por outros nós da própria rede, como externa, quando causada por outras redes ou por outros dispositivos que operam na mesma faixa de frequência.

Mesmo com o desenvolvimento de uma variedade de tecnologias no decorrer dos anos, é necessária a realização de mais trabalhos experimentais para fazer com que essas aplicações se tornem mais confiáveis e robustas. Avaliações de desempenho proveem informações valiosas para o desenvolvimento de ferramentas e soluções, visando melhorar o desempenho dos sistemas avaliados e determinar seus benefícios e limitações (YICK, 2010).

As RSSF possuem um grande potencial de aplicação, podendo ser inseridas em diversas áreas. Dessa forma, é de suma importância o estudo para o aprimoramento dessas redes em vários aspectos, permitindo o uso mais eficiente dessa tecnologia, bem como a inserção em novas áreas de aplicação.

2.1.1 Padrão IEEE 802.15.4 e Protocolo Zigbee

A grande demanda de sistemas para a área de monitoramento e controle aliada aos avanços tecnológicos ocorridos na área da microeletrônica, em especial dos dispositivos de transmissão de dados sem fio, bem como a disponibilidade de sensores de baixo custo com capacidade computacional, fizeram crescer o interesse de pesquisadores e empresas pelo desenvolvimento de uma variedade de aplicações, tais como: monitoramento de temperatura ou umidade de ambientes, detecção de incêndio ou incidentes, rastreamento de veículos, monitoramento ambiental, automação residencial, biotecnologia, monitoramento e controle industrial, segurança pública e de ambientes em geral. Nesse contexto, a tecnologia Zigbee surgiu como uma alternativa para atender o desenvolvimento destas aplicações. O protocolo

Zigbee foi proposto no ano de 2002, tratando-se de um padrão para redes de sensores sem fio, desenvolvido pela Zigbee Alliance. A Zigbee Alliance é uma associação de empresas que trabalham em conjunto para desenvolver tecnologias baseadas em uma padrão aberto global, que possibilitem comunicação sem fio confiável, com baixo consumo de energia e baixo custo, para aplicações de monitoramento e controle que não exijam grande largura de banda. Em comparação com outros protocolos, o Zigbee apresenta vantagens no que diz respeito ao consumo de energia, escalabilidade e tempo levado para adição de novos nós. A Tabela 1 apresenta um comparativo entre o protocolo Zigbee e os protocolos Bluetooth e Wi-Fi (SANTOS 2010).

Tabela 1 – Comparação entre protocolos.

Parâmetro	Zigbee	BlueTooth	Wi-Fi
Taxa de Transferência	250 Kbps	1 Mbps	54 Mbps
Modulação	DSSS	FHSS	DSSS/OFDM
Alcance	300 m	10 m	100 m
Tempo de acesso à rede	30 ms	3 s	3 s
Tempo de transição ao estado ativo	15 ms	3 s	-
Corrente de transmissão	30 mA	40 mA	400 mA
Corrente em *standby*	3 uA	200 uA	20 mA
Duração de baterias	1000 dias	1 a 7 dias	0,5 a 5 dias
Número de nós	65536	7	32

O protocolo Zigbee implementa a camada de rede. As camadas física e de controle de acesso ao meio são implementadas de acordo com o padrão IEEE 802.15.4. São definidas três faixas de frequência para comunicação: 868 MHz, 915MHz e 2.4GHz. O padrão especifica vinte e sete canais para comunicação, sendo um para a primeira faixa, dez para segunda e dezesseis para a terceira (KARL, 2005). A Tabela 2 mostra as faixas de frequência alocadas para cada canal na banda de 2.4 GHz (GOMES, 2010).

Tabela 2 – Faixas de frequências alocadas aos canais na faixa de 2.4 GHz

Canal	Frequência mais Baixa (Hz)	Frequência Central (Hz)	Frequência mais Alta (Hz)
1	2.404	2.405	2.406
2	2.409	2.410	2.411
3	2.414	2.415	2.416
4	2.419	2.420	2.421
5	2.424	2.425	2.426
6	2.429	2.430	2.431
7	2.434	2.435	2.436
8	2.439	2.440	2.441
9	2.444	2.445	2.446
10	2.449	2.450	2.451
11	2.454	2.455	2.456
12	2.459	2.460	2.461
13	2.464	2.465	2.466
14	2.469	2.470	2.471
15	2.474	2.475	2.476
16	2.479	2.480	2.481

Em uma rede Zigbee os nós podem ser de três tipos: PAN Coordinator, FFD (Full Function Device) e RFD (Reduced Function Device). Toda rede Zigbee deve estar associada a um coordenador (PAN Coordinator), que é responsável pela inicialização, distribuição de endereços, manutenção da rede e reconhecimento de todos os nós, entre outras funções. Além disso, ele pode servir como ponte entre várias outras redes ZigBee. Os nós RFD podem funcionar apenas como nós finais. Os nós finais são responsáveis pelas funções de sensoriamento e/ou atuação. Os nós FFD funcionam como nós finais na rede, mas podem também exercer a função de roteador intermediário entre nós, sem precisar do intermédio do coordenador. Por meio de roteadores, uma rede ZigBee pode ser expandida e obter maior cobertura (SANTOS, 2010) (KARL, 2005).

A topologia de uma rede Zigbee pode ser organizada de três formas: estrela malha e árvore. Uma rede em estrela é formada por um nó coordenador e múltiplos nós finais. O

coordenador gerencia a rede e roteia todos os pacotes. Dessa forma, os nós finais se comunicam entre si através do coordenador. Uma rede em árvore é formada a partir de uma hierarquia, na qual cada conjunto de nós finais se comunica com um determinado roteador, que por sua vez se comunica com o coordenador. Uma rede em malha funciona de modo ad hoc, na qual um roteador pode se comunicar com qualquer outro diretamente; ou seja, sem a necessidade de intermediação por parte do coordenador. Nessa topologia a rede é auto-organizável, podendo se ajustar automaticamente, tanto na sua inicialização como na entrada ou saída de novos nós na rede. A topologia em malha permite também aumentar a abrangência da rede, visto que a comunicação de nós distantes com o coordenador pode ser feita a partir de vários saltos, através dos roteadores (CAO, 2008). A Figura 2 ilustra as três topologias (SANTOS 2010).

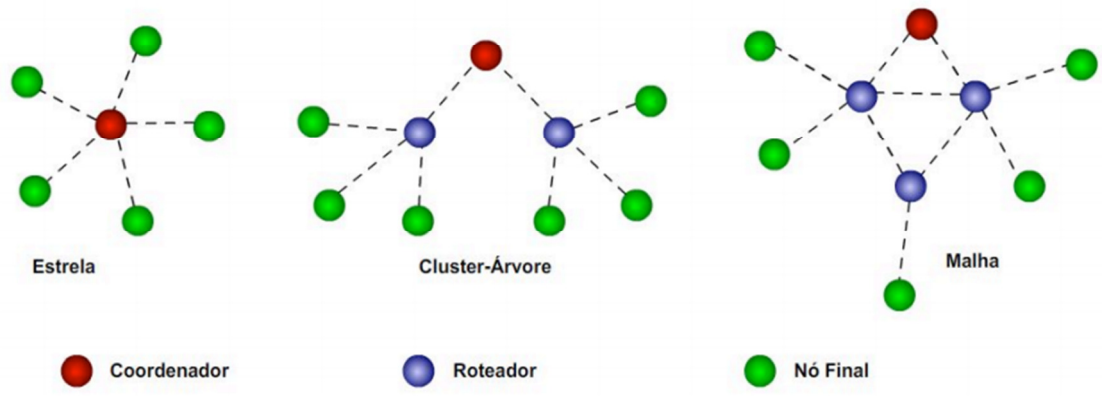


Figura 2 – Topologias de redes ZigBee.

2.2 Sistema Eletrônico Embarcado Sem Fio para Monitoramento de Motores em um Ambiente Industrial

Nesta seção será descrito por completo o sistema que realiza a leitura dos dados de eficiência e torque de motores industriais do qual o subsistema proposto neste trabalho faz parte. O sistema inicia-se com os sinais obtidos pelos transformadores de Potencial (TP) e Transformadores de Corrente (TC) utilizados para a aquisição dos sinais elétricos do motor.

Os TP são transformadores de tensão geralmente destinados à medição de energia elétrica de média e alta tensão. Eles basicamente reduzem de forma linear a tensão elétrica para que esta se torne compatível com o nível de tensão elétrica de trabalho em circuitos eletrônicos. A aquisição da tensão no motor é feita através de dois TP medindo a tensão elétrica entre fases do cabo de força do motor. Os TP reduzem a tensão real em 1:1000 possibilitando o processamento dos dados pelo microcontrolador.

O TC utilizado para medir corrente é um sensor de efeito Hall. O cabo condutor de corrente elétrica gera um campo eletromagnético induzido na região próxima à sua superfície e o sinal de saída é proporcional à corrente real.

Na Figura 3 é ilustrado o TC, utilizado para medir corrente do motor. O sensor é de fácil instalação, tendo em vista que o cabo passa por um orifício no seu centro e não há necessidade de desencapar ou cortar o cabo. A corrente de alimentação do motor é fornecida em uma razão de saída de 1:1000 e transformada em tensão através de um resistor. Desta forma, é transmitida ao conversor A/D a tensão proporcional à corrente e de fator de transformação conhecido.

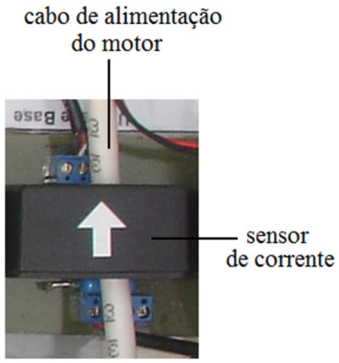


Figura 3 - TC utilizado para medir a corrente fornecida ao motor.

O microcontrolador com conversor analógico/digital integrado realiza as aquisições dos dados remotamente. O microcontrolador a ser utilizado é o dsPIC33FJ64GP706. PICs são microcontroladores fabricados pela Microchip. A sua peculiaridade na arquitetura (tipo Harvard modificada), possibilita que enquanto uma instrução é executada outra já pode ser buscada na memória em paralelo. Esta característica põe os PICs à frente da maioria dos microcontroladores encontrados no mercado, pois estes

últimos apresentam arquitetura Von Neumann, nos quais a Unidade Central de Processamento é interligada à memória por um único barramento (*bus*), o que torna o processamento mais lento. O dsPIC33FJ64GP706 possui 16 bits para transmissão dos dados e 24 bits para instruções, com capacidade de realizar mais de 40 Milhões de instruções por segundo. Ele apresenta um grande conjunto de funcionalidades para processamento de sinais (DSP *engine*), dentro de uma arquitetura de 16 bits.

A determinação do torque no entreferro é feita através da amostragem digital das tensões e correntes que alimentam o motor e estimação (sensorless) da velocidade angular do eixo. Estes dados são processados e é fornecido o sinal do torque.

A medida do torque no entreferro é feita utilizando a Equação 1. Esta equação é válida tanto para os motores ligados em Y, sem conexão com o neutro, como em motores ligados em Δ através de três fios (HSU *et al*, 1995):

$$T_{ag} = \frac{p\sqrt{3}}{6}\left\{(i_a - i_b)\int [v_{ca} + r(2i_a + i_b)]dt + (2i_a + i_b)\int [(v_{ab} - r(i_a - i_b)]dt\right\} \quad (1)$$

Onde r é a resistência do enrolamento do estator, i_a *e* i_b são as correntes de linha na entrada do motor e v_{ca} e v_{ab} são tensões entre fases.

As integrais da Equação 1 que representam o fluxo de acoplamento correspondente o torque. Nestas integrais deve-se levar em consideração a fase inicial da força controeletromotriz que gera um valor DC, pois acúmulo dessas componentes leva a saturação do sinal. Portanto o algorítimo utilizado para o cálculo do torque no entreferro deve compensar os componentes DC gerados, assim como o valor off-set inerente aos sensores utilizados e ao processo de conversão analógico/digital.

O torque no eixo do motor é dado pela Equação 2, (abaixo reescrita para melhor compreensão), onde são consideradas as perdas mecânicas (atrito e ventilação) e perdas adicionais do motor.

$$T = T_{ag} - \frac{L_{mec}}{\omega} - \frac{L_{ar}}{\omega} \quad (2)$$

Onde:

L_{mec} – perdas mecânicas são aquelas causadas por atrito e ventilação;

L_{ae} – perdas adicionais (stray-load losses) do rotor (L_{ar})

ω – velocidade angular do eixo do motor

T_{ag} – torque no entrefero

A partir do torque pretende-se calcular a eficiência energética do sistema. A eficiência é calculada através do torque no entreferro, conforme a Equação 3 abaixo.

$$\eta = \frac{P_{saida}}{P_{entrada}} - \frac{T\omega}{P_{entrada}} = \frac{T_{ag}\omega - L_{mec} - L_{ar}}{P_{entrada}} \quad (3)$$

Onde:

$P_{entrada}$ – potência de entrada;

$P_{saída}$ – potência de saída;

T – torque no eixo do motor

No Layout da placa, utilizada para realizar aquisição dos dados e também para cálculo do torque e a eficiência energética do equipamento a ser monitorado, são levados em consideração os elementos de baixa e de alta frequência presentes no circuito. A intenção é isolar o terra analógico, utilizado na parte analógica dos canais de entrada do conversor A/D e de alimentação dos componentes da placa, do terra digital que envolve as componentes de alta frequência do cristal do microcontroaldor, que poderá ultrapassar a frequência de 40 MHz. Esta isolação é fundamental para diminuir a contaminação entre os circuitos envolvidos (KESTER, 1999, WU et al, 2004, MOONGILAN, 1998, LIAW, MERKELO, 1996). O acoplamento de referência entre os circuitos pode ser óptico, conforme pode ser visto na Figura 4.

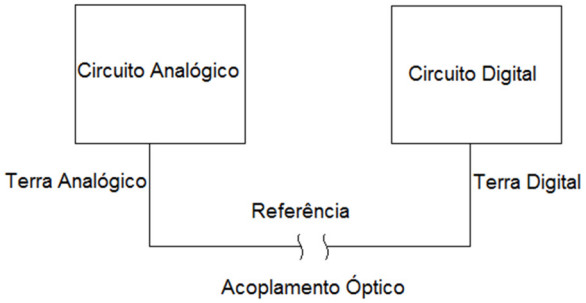


Figura 4 – Diagrama para isolação entre o terra analógico e o terra digital.

A unidade remota do sistema é posicionada na alimentação do motor para medir corrente e tensão. Na Figura 5 é ilustrado o esboço da unidade remota a ser instalada na entrada do motor. A unidade remota é formada por uma caixa metálica resistente a choque e a intempéries, onde os cabos de alimentação do motor atravessam o seu interior. Dentro da caixa estão os sensores de corrente que fazem a leitura da corrente elétrica do motor e os sensores de tensão que fazem a leitura da tensão elétrica de alimentação do motor. Na unidade remota existe uma fonte alimentadora e abaixadora de tensão que recebe energia da rede local e alimenta os circuitos dos sensores de corrente e a placa de conversão A/D e processamento de dados (PCADPD).

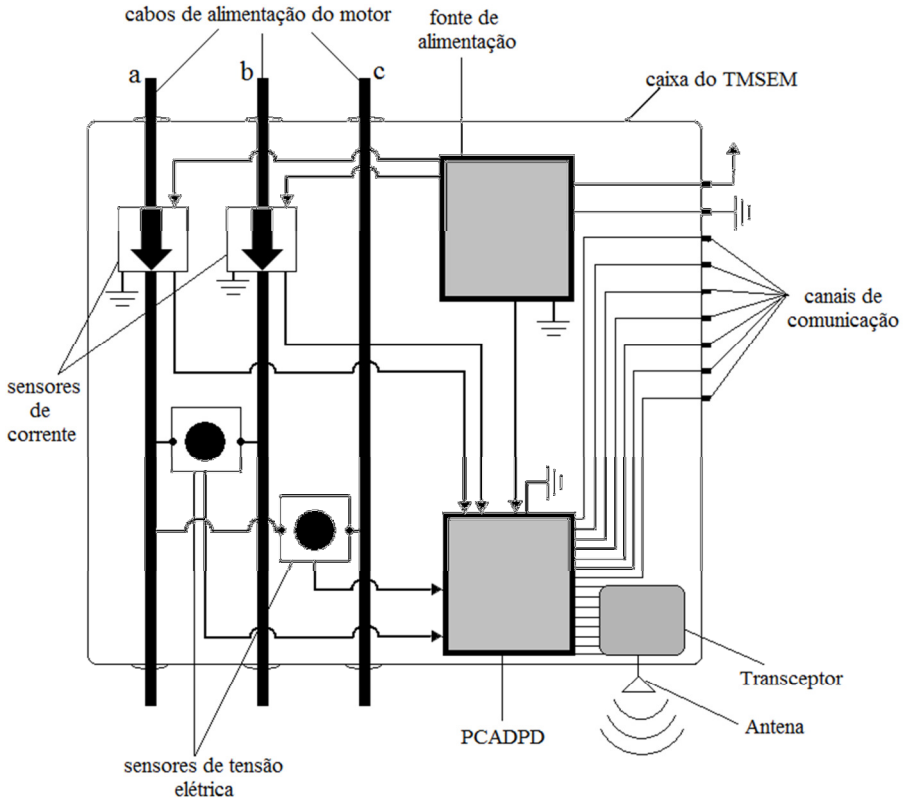


Figura 5 – Interior da unidade remota (vista superior).

A PCADPD é capaz de disponibilizar o sinal do torque no padrão 4 a 20 mA, no padrão serial e transmiti-lo sem fio utilizando o protocolo ZigBee. Ainda na PCADPD, estão disponíveis vários canais de entrada e saída que contemplam possíveis sensores auxiliares e canais de controle, bem como interface com o usuário. A PCADPD possui gravação *in-circuit* onde o firmware pode ser atualizado em campo.

Os sinais de torque e eficiência do motor são transmitidos ao transceptor ZigBee. O sistema é habilitado para transmissão do sinal no padrão IEEE 802.15.4. Os dados são enviados através da rede sem fio para um subsistema de monitoramento, onde os dados ficarão disponíveis para os operadores e podem ser armazenados em um banco de dados. A Figura 6 mostra o diagrama em blocos do sistema completo:

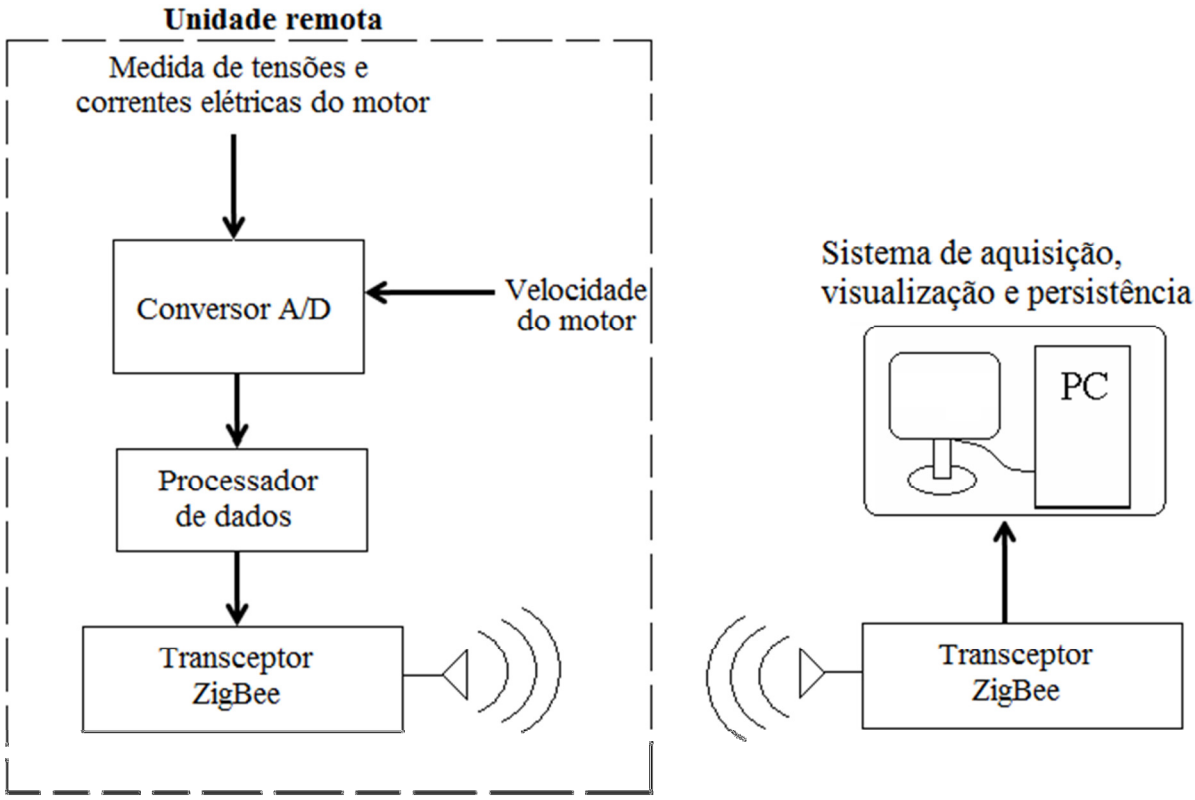


Figura 6 – Diagrama de blocos do sistema.

3. ESTADO DA ARTE

Neste capítulo dissertaremos brevemente sobre os sistemas SCADA atuais. Serão citados também outros trabalhos sobre aplicação de redes de sensores sem fio em ambientes industriais.

3.1 Sistemas SCADA

Sistemas de supervisão são bastante comuns na indústria. São também conhecidos como Sistemas SCADA (Supervisory Control And Data Acquisition), ou traduzido, Sistemas de Controle de Supervisão e Aquisição de Dados. Como a sigla sugere, não se trata de um sistema de controle completo, mas sim de um sistema focado no nível da supervisão. Desse modo, trata-se de um pacote de software que está posicionado acima do hardware com o qual possui uma interface, geralmente através de controladores lógicos programáveis (CLPs) ou outros módulos de hardware comerciais. Sistemas SCADA contemporâneos possuem, predominantemente, características de controle em malha aberta e utilizam comunicações de longa distância, embora possam existir alguns elementos de controle em malha fechada e/ou comunicações de curta distância (DANEELS, 1999). Em resumo, trata-se de um software de controle de processo que adquire, remotamente, dados em tempo real da produção, seja para monitorar variáveis de processo e estado das máquinas, ou seja para controle e regulação das mesmas. Alguns dos exemplos do uso de sistemas SCADA incluem aplicações nas indústria de energia, telecomunicações, transporte e controle de água e esgoto.

Os primeiros sistemas SCADA, basicamente telemétricos, permitiam informar periodicamente o estado corrente do processo industrial, monitorando sinais representativos de medidas e estados de dispositivos, através de um painel de lâmpadas e indicadores, sem que houvesse qualquer interface aplicacional com o operador.

Atualmente, os sistemas de automação industrial utilizam tecnologias de computação e comunicação para automatizar a monitoração e controle dos processos industriais, efetuando coleta de dados em ambientes complexos, eventualmente dispersos geograficamente, e a respectiva apresentação de modo amigável para o operador, com

recursos gráficos elaborados (interfaces homem-máquina) e conteúdo multimídia (WRIGHT, BAILEY, 2003).

Geralmente um sistema SCADA possui características particulares, intrinsicamente ligadas aos requisitos de sua aplicação. Não obstante, é possível identificar alguns elementos comuns a todos eles.

3.1.1 Arquitetura

Quanto à arquitetura de hardware, é possível distinguir duas camadas em um sistema SCADA: a camada cliente, responsável pela interação humano computador e a camada de servidor de dados, responsável pelas atividades de controle de processo (DANEELS, 1999).

A camada de servidor adquire os dados através de PLC´s (Programmable Logic Controllers) e RTU´s (Remote Terminal Units), com a leitura dos valores atuais dos dispositivos a que estão associados e seu respectivo controle. Os PLCs e RTUs são unidades computacionais específicas, utilizadas nas instalações fabris (ou qualquer outro tipo de instalação que se deseje monitorar) para a funcionalidade de ler entradas, realizar cálculos ou controles, e atualizar saídas. A diferença entre os PLCs e as RTUs é que os primeiros possuem mais flexibilidade na linguagem de programação e controle de entradas e saídas, enquanto as RTUs possuem uma arquitetura mais distribuída entre sua unidade de processamento central e os cartões de entradas e saídas, com maior precisão e seqüenciamento de eventos (WRIGHT, BAILEY, 2003).

Os servidores de dados podem ligar-se entre si ou a clientes através de uma rede de computadores (ethernet, wireless, fibras óticas, entre outras). A Figura 7 mostra uma arquitetura típica de hardware em um sistema SCADA:

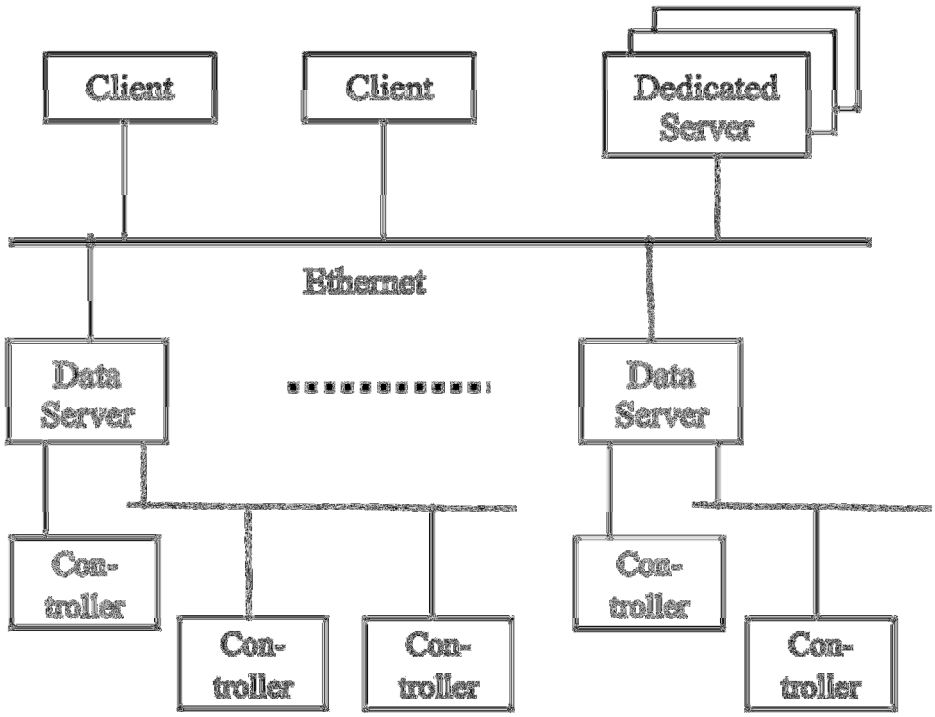


Figura 7 – Arquitetura de hardware SCADA (DANEELS, 1999).

A arquitetura de software baseia-se em um banco de dados em tempo real, localizado em um ou mais servidores. Os servidores são responsáveis pela aquisição de dados e na manipulação de um conjunto de parâmetros, como por exemplo, controle de polling, checagem de alarmes, cáculos, registros e arquivamento.

3.1.2 Comunicação

Um sistema SCADA tem na troca de informações a sua principal funcionalidade. Basicamente, temos os seguintes tipos de comunicação:

- Comunicação com os PLCs/RTUs;
- Comunicação com outras estações SCADA;
- Comunicação com outros sistemas.

A comunicação com os equipamentos de campo geralmente ocorrem por polling ou por interrupção. Em ambos os casos, é necessário um protocolo de comunicação, que pode ser tanto de domínio público quanto de acesso restrito.

A comunicação por polling (ou Master/Slave) faz com que a estação central (Master) tenha controle absoluto das comunicações, efetuando seqüencialmente o polling aos dados de cada estação remota (Slave), que apenas responde à estação central após a recepção de um pedido, ou seja, em half-duplex. Isto traz simplicidade no processo de coleta de dados, inexistência de colisões no tráfego da rede, facilidade na detecção de falhas de ligação e uso de estações remotas não inteligentes. No entanto, traz incapacidade de comunicar situações à estação central por iniciativa das estações remotas (WRIGHT, BAILEY, 2003).

Já a comunicação por interrupção ocorre quando o PLC ou o RTU monitora os seus valores de entrada e, ao detectar alterações significativas ou valores que ultrapassem os limites definidos, envia as informações para a estação central. Isto evita a transferência de informação desnecessária, diminuindo o tráfego na rede, além de permitir uma rápida detecção de informação urgente e a comunicação entre estações remotas (slave-to-slave). As desvantagens desta comunicação são que a estação central consegue detectar as falhas na ligação apenas depois de um determinado período (ou seja, quando efetua polling ao sistema) e são necessários outros métodos (ou mesmo ação por parte do operador) para obter os valores atualizados (WRIGHT, BAILEY, 2003).

A comunicação com outras estações SCADA pode ocorrer através de um protocolo desenvolvido pelo próprio fabricante do sistema SCADA, ou através de um protocolo conhecido via rede Ethernet TCP/IP, linhas privativas ou discadas. A Internet é cada vez mais utilizada como meio de comunicação para os sistemas SCADA. Através do uso de tecnologias relacionadas com a Internet, e padrões como Ethernet, TCP/IP, HTTP e HTML, é possível acessar e compartilhar dados entre áreas de produção e áreas de supervisão e controle de várias estações fabris. Através do uso de um browser de Internet, é possível controlar em tempo real, uma máquina localizada em qualquer parte do mundo. O browser comunica com o servidor web através do protocolo http, e após o envio do pedido referente à operação pretendida, recebe a resposta na forma de uma página HTML. Algumas das vantagens da utilização da Internet e do browser como interface de visualização SCADA são o modo simples de interação, ao qual a maioria das pessoas já está habituada, e a facilidade de manutenção do sistema, que precisa ocorrer somente no servidor (DANEELS, 1998).

Já a comunicação com outros sistemas, como os de ordem corporativa, ou simplesmente outros coletores ou fornecedores de dados, pode se dar através da implementação de módulos específicos, via Bancos de Dados, ou outras tecnologias como o XML e o OPC.

3.2 Outros Trabalhos Sobre Redes de Sensores Sem Fio em Ambientes Industriais.

Esta seção descreve alguns trabalhos já desenvolvidos focando na aplicação de redes de sensores sem fio em ambiente industrial. Salvadori et al. propuseram um sistema digital para avaliação de uso de energia, diagnóstico, controle e supervisão de sistemas elétricos aplicando redes de sensores sem fio com baixo consumo de energia (SALVADORI et al, 2009). O sistema implementa um protocolo de gerenciamento dinâmico de energia e possui fácil instalação, baixo custo, fácil implementação de rotinas redundantes, portabilidade, versatilidade e tempo de vida estendido. Esse trabalho foca no consumo de energia e não traz estudos detalhados sobre erros de transmissão e condições do canal de comunicação.

Bin Lu et al. identifica em seu trabalho as sinergias entre redes de sensores sem fio e análise de motores baseado nos sinais elétricos de forma não invasiva e propõe um esquema para aplicar redes de sensores sem fio no monitoramento online e remoto de energia e diagnóstico de faltas em sistemas de motores industriais (LU, 2009).

O artigo também apresenta algumas análises de desempenho e expõe desafios reais no desenvolvimento e implantação de redes de sensores na prática, incluindo qualidade dinâmica do enlace sem fio, ruído e interferência, e o impacto do ambiente no alcance de comunicação e confiabilidade. São implementados dois métodos não invasivos para diagnóstico de motores para provar sua aplicabilidade e os desafios em aplicar o esquema proposto em um ambiente industrial real são analisados experimentalmente usando resultados de ensaios em campo.

O trabalho de Lu et al. possui grande relevância visto que ele implementa métodos de monitoramento de eficiência em motores utilizando redes de sensores sem fio (LU, 2009).

Entretanto, apesar de apresentar estudos experimentais de desempenho do sistema, não inclui uma análise suficientemente detalhada da relação entre o desempenho da rede e a utilização espectral. Além disso, não foi explorada a capacidade de processamento local e a topologia utilizada foi uma simples topologia em estrela com apenas um nó final; ou seja, o desempenho de uma rede desse tipo funcionando com uma maior quantidade de nós e de forma ad hoc não foi investigado.

4. MATERIAIS E MÉTODOS

Neste capítulo serão apresentadas todas as ferramentas utilizadas no desenvolvimento do software objeto deste trabalho.

4.1 Java

Java é uma linguagem de programação orientada a objetos desenvolvida pela Sun Microsystems (atualmente comprada pela Oracle). Baseada no C++, a linguagem Java foi projetada para ser pequena, simples e portável a todas as plataformas e sistemas operacionais, tanto o código fonte como os binários. Esta portabilidade é obtida pelo fato da linguagem ser interpretada, ou seja, o compilador gera um código independente de máquina chamado byte-code. No momento da execução este byte-code é interpretado por uma máquina virtual instalado na máquina. Para portar Java para uma arquitetura de hardware específica, basta instalar a máquina virtual (interpretador). Além de ser integrada à Internet, Java também é uma excelente linguagem para desenvolvimento de aplicações em geral (DEITEL, 2003).

Várias foram as motivações para a escolha desta linguagem no desenvolvimento do sistema proposto por este trabalho. A primeira diz respeito à presença da API Swing, uma biblioteca para criação de interfaces gráficas que, como a própria Java, é independente da plataforma utilizada. Somado a isto, existe uma API para geração de gráficos (JFreeChart), compatível com a API Swing, que é de fácil uso e de excelente desempenho.

Como segundo fator, existe um extenso suporte à programação concorrente em Java, com a presença de classes que implementam threads e estruturas que facilitam a exclusão mútua em regiões críticas, como semáforos, monitores, entre outros. Todas essas características são vitais para o sistema proposto.

Como ponto negativo, Java não é uma linguagem que apresenta uma excelente performance frente à linguagens como C/C++, pelo fato de ser interpretada e não compilada. O consumo de memória de programas Java comparados à programas de outras linguagens também é relativamente alto. Porém, estudos sugerem que o ponto mais determinante no desempenho de um programa está na qualidade do seu código. Programas bem escritos em

Java podem ser tão ou mais eficientes que programas de qualidade mediana em C/C++ (PRECHELT, 1999). É importante frisar também que o aumento de performance da máquina virtual Java a cada versão lançada, bem como a evolução do poder de processamento dos computadores modernos, reduz cada vez mais a diferença de performance entre programas compilados e interpretados.

4.2 UML

A UML é uma linguagem de modelagem não proprietária desenvolvida pela OMG, um consórcio de empresas de computação internacional, de adesão livre e sem fins lucrativos. Ela permite que o desenvolvedor de software especifique, visualize e documente modelos de software, incluindo sua estrutura e design, de modo que se adeque a todos seus requerimentos (OMG, 2011).

A UML 2.0 define 13 tipos diferentes de diagramas, divididos em três categorias. Seis tipos de diagramas representam estruturas estáticas da aplicação; três representam tipos genéricos de comportamento e quatro representam diferentes aspectos de interação. Segue abaixo a lista de todos os diagramas presentes na UML:

- Diagramas de estruturas incluem: classe, objeto, componentes, estrutura composta, pacotes e desenvolvimento (deployment).
- Diagramas de comportamento incluem: casos de uso, atividade e máquina de estados.
- Diagramas de interação incluem: sequência, comunicação, tempo e interação (OMG, 2011).

A ferramenta utilizada para a modelagem UML foi o astah community 6.3, um software de licença gratuita (CHANGEVISION, 2011).

4.3 NetBeans

O NetBeans é um ambiente de desenvolvimento integrado (IDE) gratuito e de código aberto, desenvolvido em Java e com suporte à várias linguagens, como Java, C/C++, PHP, JavaScript, entre outras. Este IDE possui várias ferramentas que permitem a criação de software profissionais para desktops, empresas, web e dispositivos móveis (ORACLE, 2011).

Além das características dos melhores IDE´s existentes, como complementação de código, refatoração, inserção de código, fácil integração com banco de dados, controle de versão integrado, entre outros, o NetBeans apresenta um construtor de interfaces gráficas para o usuário (GUI) bastante poderoso, composto de componentes da biblioteca Swing e AWT Java. Este construtor abrevia bastante o tempo de desenvolvimento de interfaces gráficas para sistemas desktop. A figura 8 apresenta a tela principal do NetBeans, com destaque para o construtor de GUI.

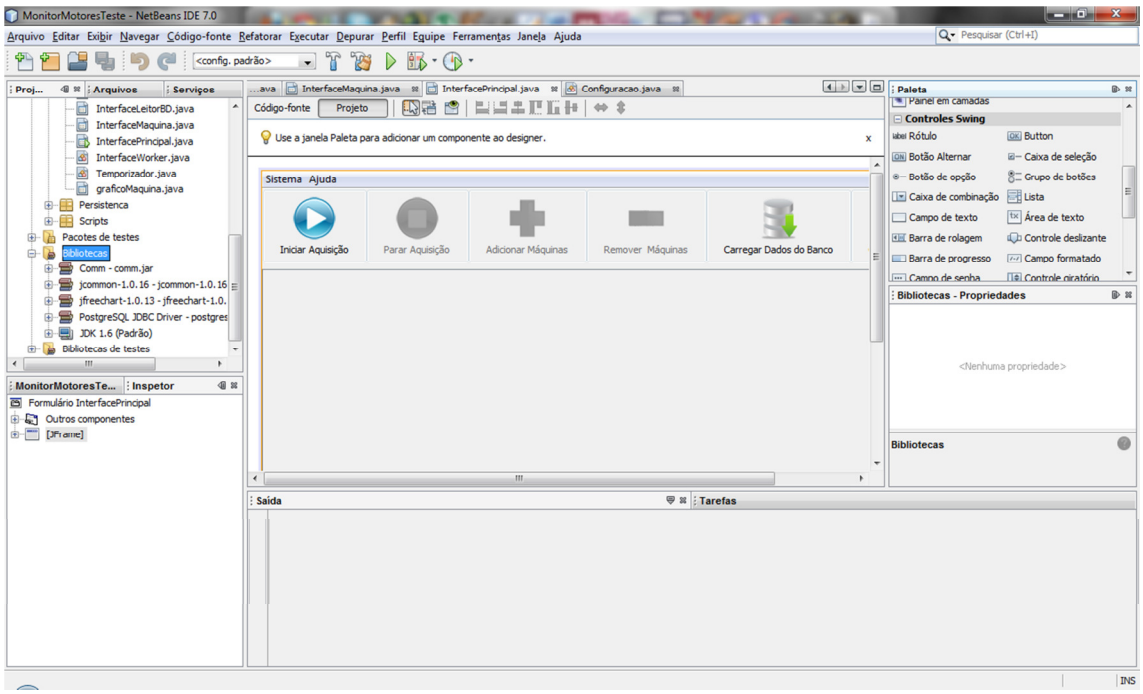


Figura 8 - Tela principal do NetBeans, com destaque para o construtor GUI.

4.4 PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados (SGBD) objeto-relacional de código aberto, disponível para a maioria dos sistemas operacionais, como Windows, Linux e UNIX. Ele inclui a maioria dos tipos de dados presentes no SQL 2008, como INTEGER, NUMERIC, BOOLEAN, CHAR, DATE, INTERVAL, e TIMESTAMP. Também oferece suporte ao armazenamento de grandes objetos binários, incluindo imagens, sons ou vídeo. Possui interfaces de programação nativas para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre outras, além de possuir extensa documentação (POSTRGRESQL, 2011).

As características de integridade de dados do PostgreSQL incluem chaves primárias (compostas), chaves estrangeiras com restrição e updates/deletes em cascata, checagem de restrições, restrições únicas e restrições not null. Outas características avançadas do SGBD incluem: herança de tabelas, sistema de regras e eventos de banco de dados (POSTGRESQL, 2011).

O PostgreSQL é altamente escalável, seja na quantidade de dados que ele pode manipular ou seja na quantidade simultânea de usuários que ele suporta. Existem sistemas PostgreSQL ativos em ambientes de produção que gerenciam mais de 4 terabytes de dados. A Tabela 3 mostra os limites do sistema:

Tabela 3 - Limites de capacidade do PostgreSQL.

Limite	Valor
Tamanho máximo do banco de dados	Sem limite
Tamanho máximo de um tabela	32 Terabytes
Tamanho máximo de uma linha	1.6 Terabytes
Tamanho máximo de um campo	1 Gigabyte
Número máximo de linhas por tabela	Sem limite
Número máximo de colunas por tabela	250 – 1600 (depende dos tipos das colunas)
Número máximo de index por tabela	Sem limite

Quando instalado, o PostgreSQL permite o gerenciamento do banco de dados através da de uma ferramenta gráfica chamada pgAdmin III, facilitando bastante o uso do sistema. Também está presente o gerenciamento tradicional, através de linha de comando.

4.5 JFreeChart

JFreeChart é uma biblioteca de criação de gráficos gratuita para a plataforma Java. Ela é distribuída com o código-fonte completo sujeito aos termos da licença GNU LGPL, o que permite que seja usada em aplicações proprietárias ou gratuitas. A JFreeChart pode ser usada em aplicações Java, applets, servlets e JSP (GILBERT, 2007).

Com o uso da biblioteca JFreeCart é possível criar gráficos dos mais variados formatos, como pizza, barra, linha, gráficos de dispersão, Gantt, entre outros. Além disso, a biblioteca possui algumas características úteis, como:

- Capacidade de exportar os gráficos como imagens nos formatos PNG e JPEG;
- Tool tips para informar valores nos gráficos;
- Zoom interativo;
- Eventos de mouse nos gráficos;
- Suporte à annotations Java;
- Ajuste automático do gráfico quando atualizado com novos valores em tempo real.

Neste trabalho, a biblioteca JFreeChart foi utilizada para gerar gráficos do tipo TimeSeries. Trata-se de um gráfico de linha onde a abscissa é a variável tempo e a ordenada é outra variável qualquer. A Figura 9 mostra um exemplo deste tipo de gráfico, utilizando o JFreeChart.

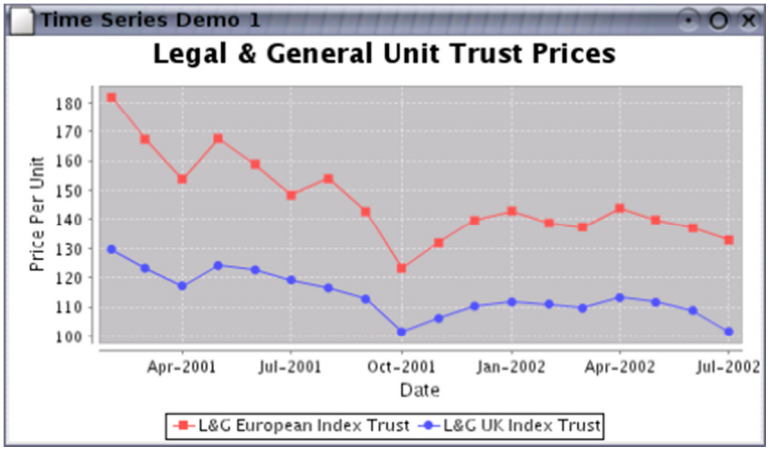


Figura 9 - Exemplo de um gráfico do tipo TimeSeries.

5. DESCRIÇÃO DO SISTEMA

5.1 Requerimentos do sistema

O sistema proposto neste trabalho tem por objetivo capturar os dados enviados pelo sistema descrito em 2.2, exibi-los para o usuário e realizar a persistência dos mesmos, caso o usuário assim deseje.

A captura dos dados é realizada através de uma porta serial (RS-232). Os dados são enviados em pacotes, obedecendo a um protocolo simples de comunicação. O sistema deve ser robusto e eficiente o bastante para que a perda de pacotes lidos em tempo real seja mínima.

A Figura 10 representa as ações que o usuário será capaz de desempenhar no sistema.

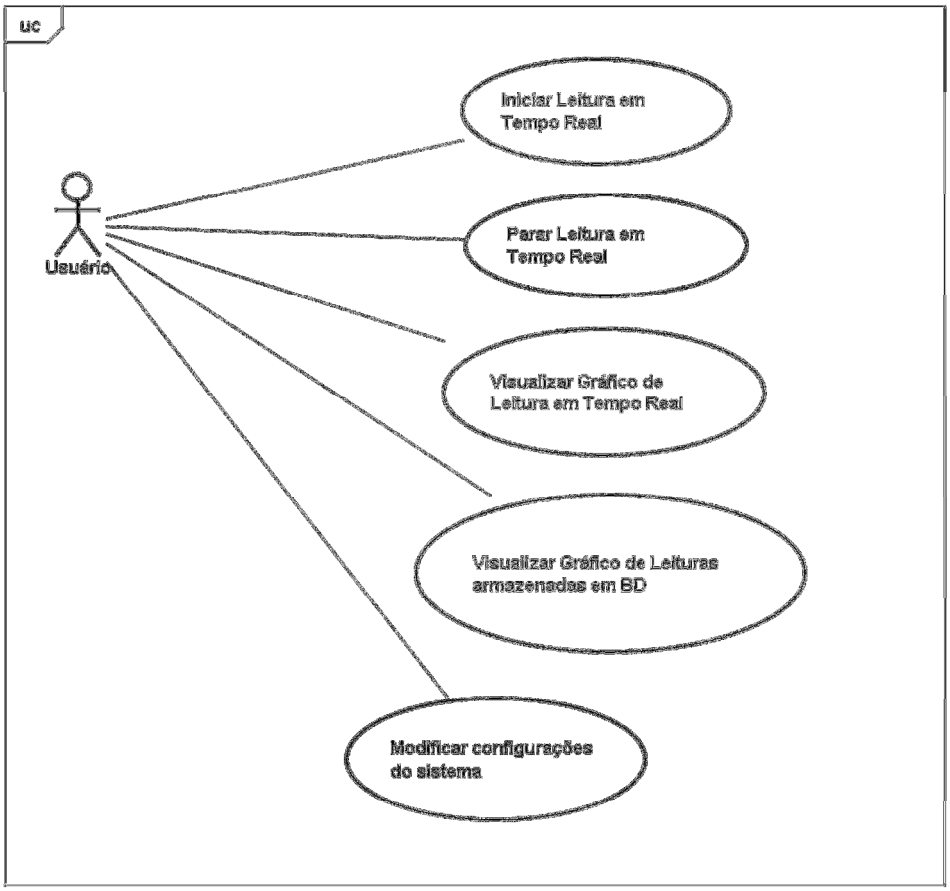


Figura 10 – Diagrama de caso de uso: sistema

Como se pode deduzir pelo diagrama acima, as ações “parar leitura” e “visualizar gráficos de leitura em tempo real” necessitam que, para serem usadas, alguma leitura em tempo real já esteja ocorrendo.

A ação “Modificar configurações do sistema” pode ser desdobrada em um novo diagrama de caso de uso, como apresentado na Figura 11.

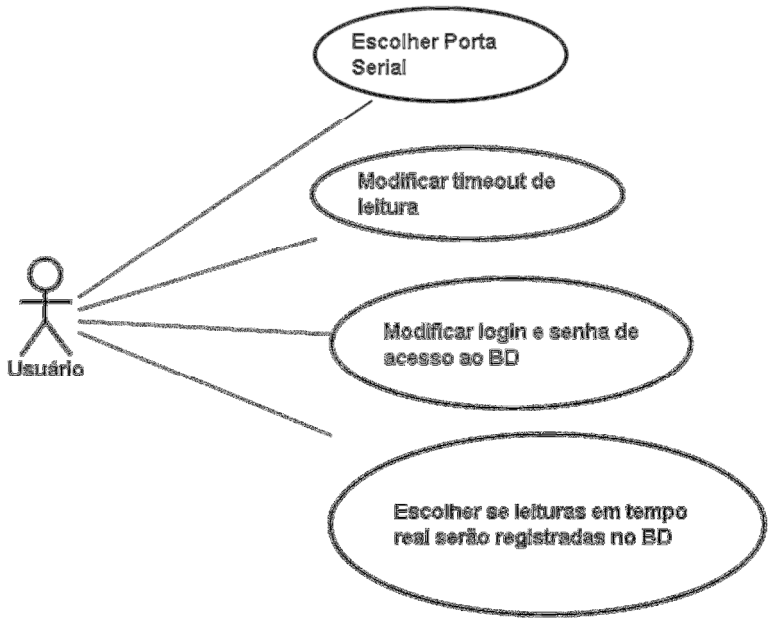


Figura 11 – Diagrama de caso de uso: configurações

As configurações do sistema estão relacionadas com a leitura de dados em tempo real. A ação “escolher porta serial” permite selecionar qual porta será utilizada pelo sistema para ler os dados (usualmente COM1 ou COM2).

A opção “modificar timeout de leitura” permite definir quanto tempo, em segundos, o sistema deverá esperar antes de emitir um sinal ao usuário indicando que não se recebeu mais dados de determinada máquina.

O usuário poderá indicar ao sistema se este deve armazenar os valores lidos em tempo real no banco de dados do sistema, bem como alterar o nome de usuário (login) e senha utilizados para acessar este mesmo banco.

A implementação do sistema foi dividida em três módulos: aquisição, interface e persistência, cada um com tarefas específicas no sistema. O módulo de aquisição é uma camada que provê serviços para o módulo de interface. Sua tarefa é ler corretamente os dados

do meio físico e repassá-los, já interpretados, para o módulo no qual ele está interligado (no caso, o módulo de interface).

O módulo de interface é responsável por toda a comunicação entre o usuário e o sistema. É a partir dele que todas as operações do sistema, descritas nas Figuras 10 e 11, são executadas. Portanto, é tarefa do módulo de interface instanciar os demais módulos do sistema e requisitar os serviços destes. É também neste módulo que está implementada toda a interface gráfica do programa.

O módulo de persistência provê uma ponte entre o banco de dados e o sistema. É através deste módulo que o sistema salva dados de aquisições em tempo real e pode, posteriormente, recuperar estes mesmos dados. O módulo de persistência pode ser considerado como um conjunto de classes utilitárias utilizadas pelo módulo de interface. Porém, sua finalidade específica permite que ele seja descrito como um módulo em separado no sistema.

As ferramentas utilizadas para a implementação, já descritas no capítulo 4, foram: diagramas UML para modelagem do sistema, Java, NetBeans e JFreeChart para escrita do código fonte e PostgreSQL para geração do banco de dados.

5.2 Módulo de aquisição

O módulo de aquisição é responsável por ler corretamente os dados enviados pelas máquinas e repassar estas informações para o módulo de interface. Este módulo foi implementado seguindo uma arquitetura de camada: o módulo provê uma interface com operações restritas para os outros módulos do sistema. A interface contém as operações de iniciaLeitura, paraLeitura e lePacote. Para se utilizar os serviços desta camada, basta instanciar a classe AquisicaoMotores. A leitura é realizada, então, utilizando o método lePacote, que retorna um pacote com os valores de endereço da máquina, eficiência e torque, além da hora da aquisição.

Os dados são obtidos através de uma conexão serial (RS-232), ligada diretamente ao sistema. O módulo de aquisição lê os dados pela porta serial, através da classe

LeitorPortaSerial, e envia cada byte lido para um buffer circular sincronizado por semáforos (LEA, 1999), implementado na classe BufferBytes.

Uma classe denominada ConstrutorPacotes retira os bytes do buffer e os analisa para determinar se estes constituem um pacote válido. O formato do pacote reconhecido pelo sistema é exibido na Tabela 4:

Tabela 4 – Formato do pacote de dados

Cabeçalho (4 bytes)				Dados (7 bytes)		
				Endereço	Eficiência	Torque
0xFF	0xFF	0xFF	0xEF	1 byte	2 bytes	4 bytes

O algoritmo para reconhecimento de um pacote válido encontra-se implementado na classe ConstrutorPacotes. Este algoritmo é baseado na máquina de estados exibida na Figura 12:

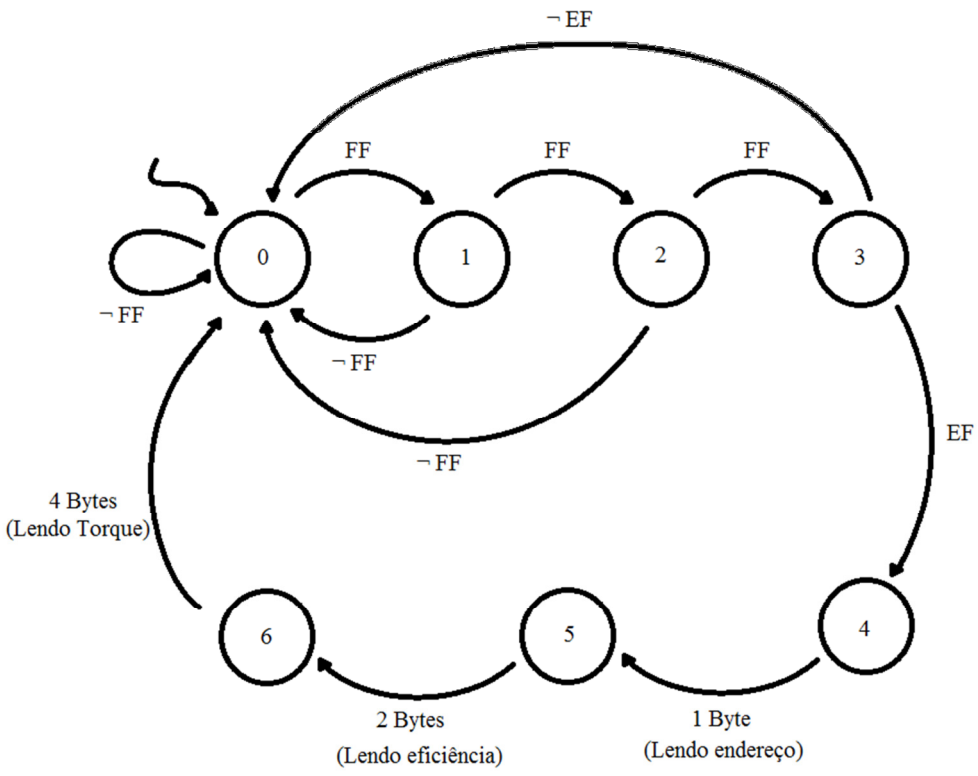


Figura 12 – Máquina de Estados para reconhecimento de um pacote válido

O algoritmo de reconhecimento implementado pela máquina de estados acima lê continuamente os bytes do buffer, procurando por um início de cabeçalho, conforme indicado no estado 0. Apenas quando o início do cabeçalho é encontrado (byte de valor FF em hexa), o algoritmo passa para o estado 1 e verifica se o próximo byte lido possui valor FF conforme o segundo byte do cabeçalho. Se são iguais, passa-se para o estado 2, onde o próximo byte lido é comparado com o terceiro byte do cabeçalho (valor FF). Finalmente, outro byte é lido e comparado com o valor EF, passando para o estado 4 em caso positivo. Se, durante a leitura de bytes nos estados de 0 ao 3, o byte lido e o byte esperado forem diferentes, o algoritmo volta ao estado 0, onde procura o início de cabeçalho.

Quando a sequência de bytes do cabeçalho é lida, ou seja, o algoritmo encontra-se no estado 4, ele lê mais 7 bytes, associando aos seus respectivos valores de endereço da máquina, eficiência e torque (indicados pelos estados 4, 5 e 6). Neste momento o algoritmo detectou um pacote completo. Então, um objeto Pacote é instanciado com todos os valores capturados e mais um atributo incluindo a data completa de aquisição, com precisão de milissegundos.

Este objeto Pacote recém-instanciado é colocado em outro buffer sincronizado com semáforos para ser finalmente consumido pelos outros módulos do sistema (LEA, 1999). Feito isto, o algoritmo de reconhecimento de pacotes volta a procurar um novo início de cabeçalho, até que a leitura seja interrompida pelo usuário.

O diagrama de sequência da Figura 13 mostra todo o processo de leitura de um pacote:

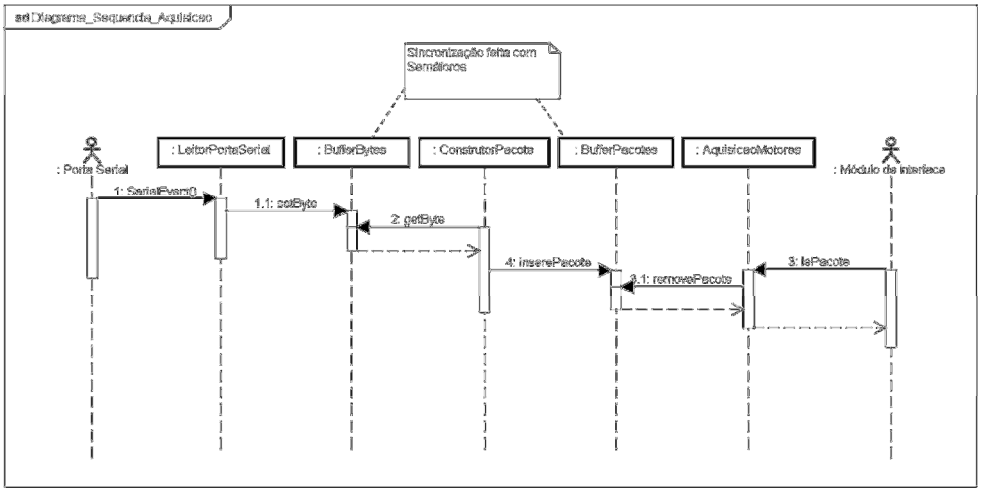


Figura 13 – Diagrama de Sequência mostrando a leitura de um pacote.

5.3 Módulo de Interface

O módulo de interface é responsável por toda a comunicação entre o usuário e o sistema. O usuário pode, inicialmente, escolher entre realizar a aquisição, modificar configurações do sistema e visualizar dados armazenados anteriormente. A Figura 14 mostra a tela inicial do sistema:

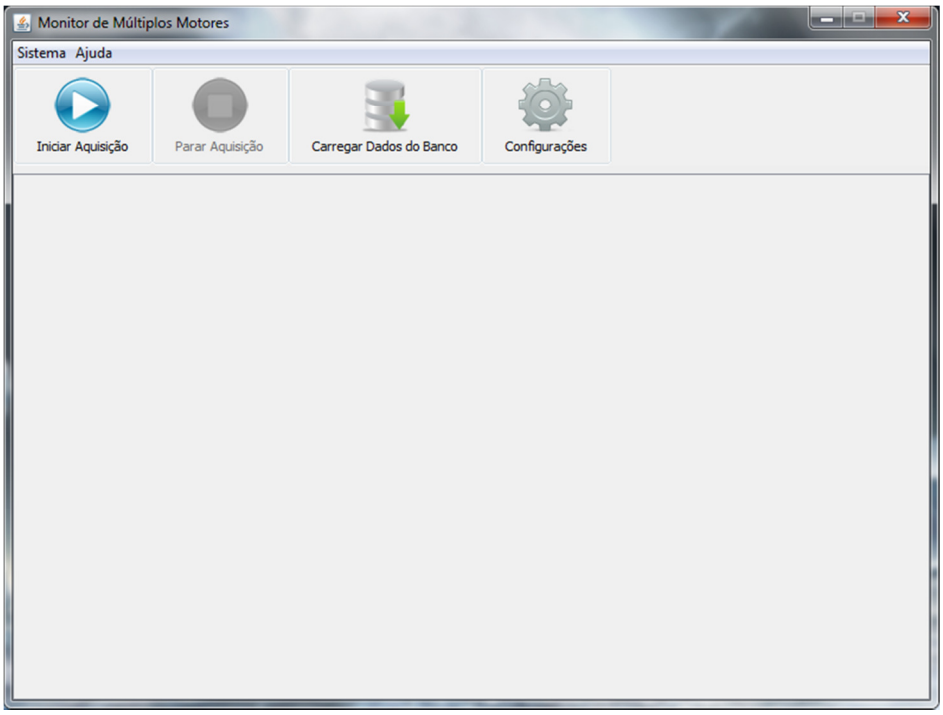


Figura 14 – Tela inicial do sistema.

Durante a aquisição dos dados, o sistema exibe todas as máquinas ativas, bem como os dados do último pacote capturado. Para cada nova máquina que se conecte a rede e passe a enviar pacotes, o sistema adiciona suas informações recebidas na tela principal. Se o usuário clicar sobre um valor qualquer, poderá visualizar um gráfico tempo x valor. O sistema também indica se a máquina ainda está ativa segundo o timeout definido pelo usuário, através de um pequeno LED ao lado da leitura. A Figura 15 mostra o sistema durante a execução de uma leitura.

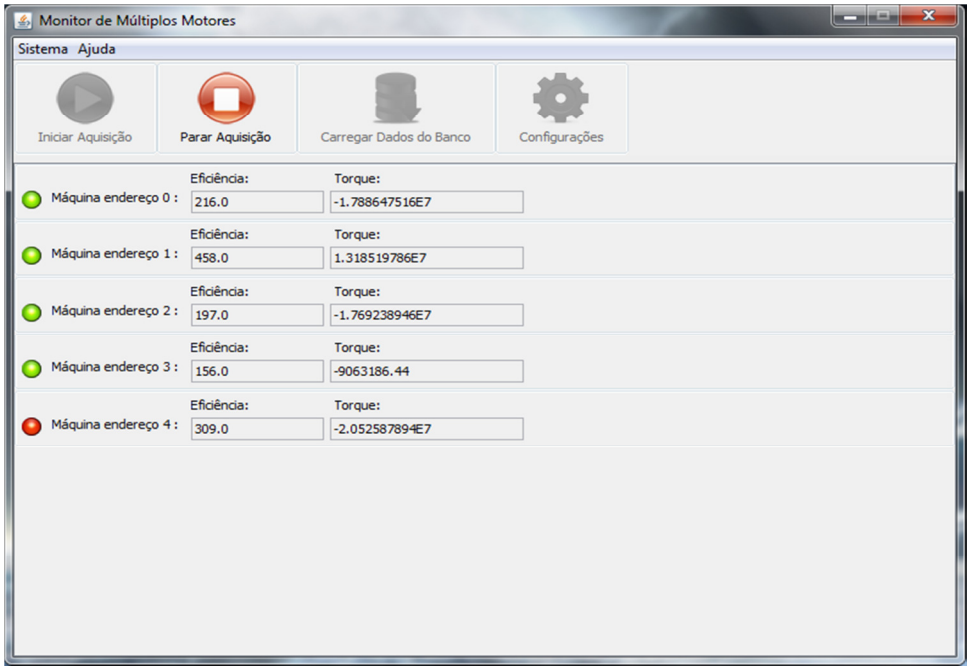


Figura 15 – Tela do sistema durante leitura.

A classe responsável pela leitura em tempo real é a InterfaceWorker. Trata-se de uma thread que herda a classe SwingWorker, permitindo a execução de tarefas de longa duração sem congelar a interface gráfica (ORACLE, 2011).

Quando o usuário clica em “Iniciar Aquisição”, um objeto da InterfaceWorker é instanciado e as configurações atuais do sistema são passadas como parâmetro. Este objeto, de acordo com as configurações, cria uma nova instância do módulo de aquisição e começa a requisitar-lhe os pacotes.

O objeto InterfaceWorker mantém, ainda, uma lista dinâmica de objetos da classe InterfaceMáquina. Os objetos desta classe são responsáveis por exibir os dados de uma determinada máquina, gerar os gráficos em tempo real se o usuário desejar e atualizar os dados de tais gráficos caso existam.

Quando um pacote é lido, o objeto InterfaceWorker verifica o endereço do mesmo. Se aquele endereço pertencer a um dos elementos da lista de objetos InterfaceMáquina, o InterfaceWorker apenas atualiza os valores do mesmo. Se não houver nenhum elemento na lista para aquele endereço, o InterfaceWorker adiciona um novo elemento InterfaceMáquina e atualiza seus valores.

O usuário pode visualizar um gráfico eficiência x tempo ou torque x tempo, bastando apenas clicar em cima do valor desejado para determinada máquina. Os gráficos são gerados com a ajuda da biblioteca JFreeChart. As janelas dos gráficos são independentes entre si. O usuário pode criar uma janela com gráfico para cada grandeza em exibição de cada máquina ativa. As Figuras 16 e 17 mostram telas destes gráficos.

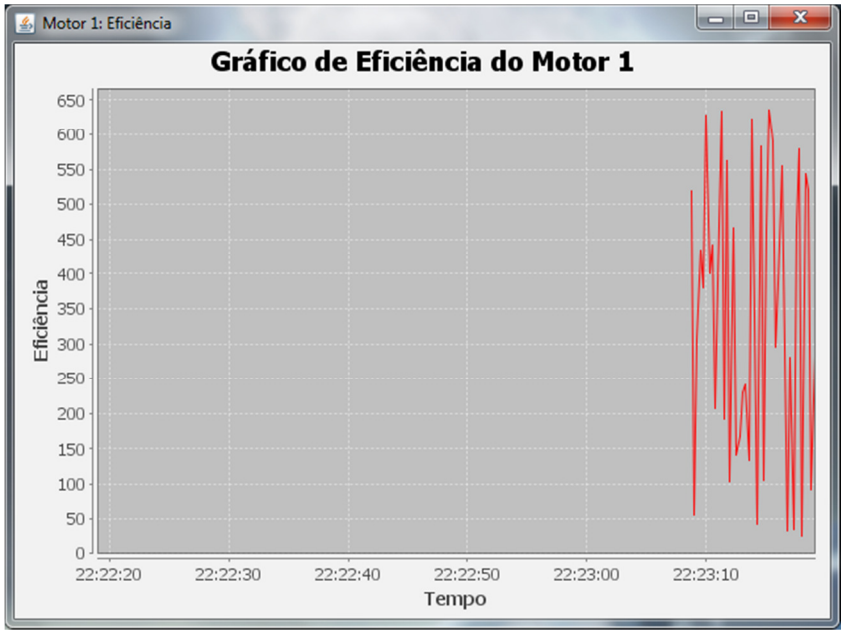


Figura 16 – Gráfico de eficiência de um dos motores em execução.

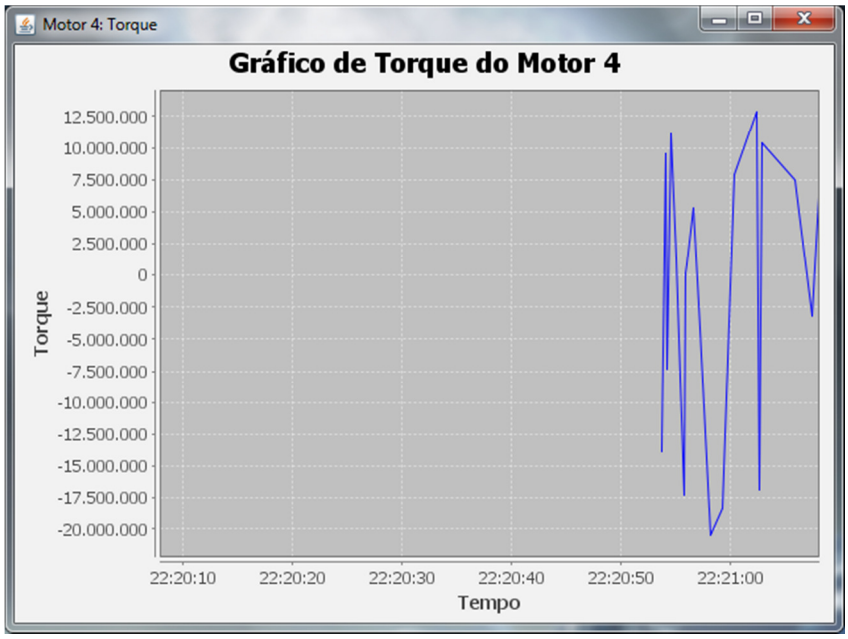


Figura 17 - Gráfico de Torque de um dos motores em execução.

O módulo de Interface permite ainda alterar configurações do sistema, de acordo com o caso de uso descrito na Figura 11. Essas configurações, se alteradas, são salvas em um arquivo criptografado, uma vez que contém informações importantes como login e senha do banco de dados acessado pelo sistema. Desse modo, as opções escolhidas pelo usuário são mantidas mesmo que o sistema seja finalizado. A Figura 18 mostra a tela que permite modificar as configurações do sistema. O usuário acessa esta tela ao pressionar o botão configurações, na tela inicial do sistema.

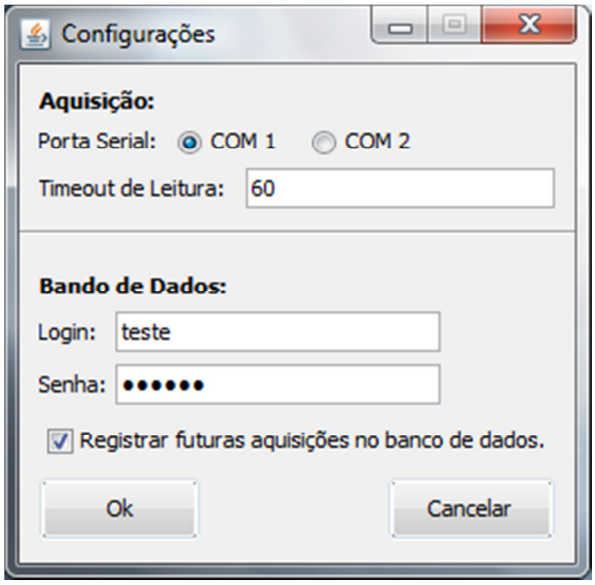


Figura 18 – Tela de configurações do sistema.

Ainda sobre o módulo de Interface, tem-se a opção de resgatar gráficos de leituras passadas do banco de dados. Porém, como esta ação do sistema está intimamente relacionada com o módulo de persistência, ela será tratada em detalhes na próxima seção.

5.4 Módulo de Persistência

Esse módulo é responsável pela persistência dos dados adquiridos, bem como sua recuperação. O módulo não se trata de uma camada na arquitetura do sistema. Ele pode ser melhor compreendido como um conjunto de classes utilitárias que permitem a troca de informações entre o sistema e o banco de dados associado.

Para o banco de dados, o SGBD utilizado foi o PostgreSQL. Nele, foi criado um banco chamado maquinadb, contendo uma única tabela chamada leitura, que armazena os dados obtidos do sistema. A Figura 19 apresenta o modelo entidade-relacionamento do banco criado:

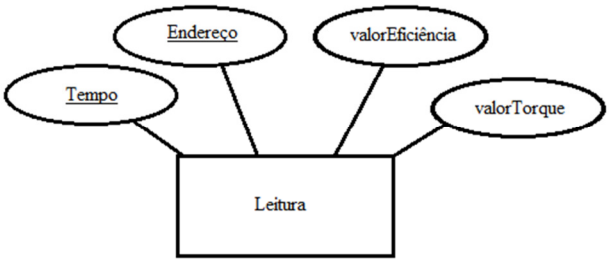


Figura 19 – Modelo E-R do banco de dados.

Nota-se, no modelo, que a hora de aquisição do pacote e o endereço da máquina formam a chave primária da tabela. O atributo tempo é do tipo TIMESTAMP, que permite armazenamento de datas com precisão de milissegundos. O atributo endereço é do tipo VARCHAR, que permite armazenamento de strings. Os atributos valorEficiência e valorTorque são ambos DOUBLE PRECISION, que permite armazenamento de valores reais.

O módulo de persistência possui duas classes, a EscritorBD e a Leitor BD. O processo de escrita de dados é bastante simples e ocorre durante a leitura de dados pelo módulo de Interface, conforme indicado no diagrama de sequência da Figura 20. A tarefa de escrita só é realizada, convém frisar, se o usuário selecionou a opção correta de armazenar os dados de leitura no banco. Desse modo, toda vez que o objeto InterfaceWorker lê um pacote do módulo de aquisição e faz a atualização dos dados na respectiva InterfaceMáquina, ele envia o pacote para um objeto EscritorBD, que traduz os dados em uma consulta SQL e insere os dados no banco.

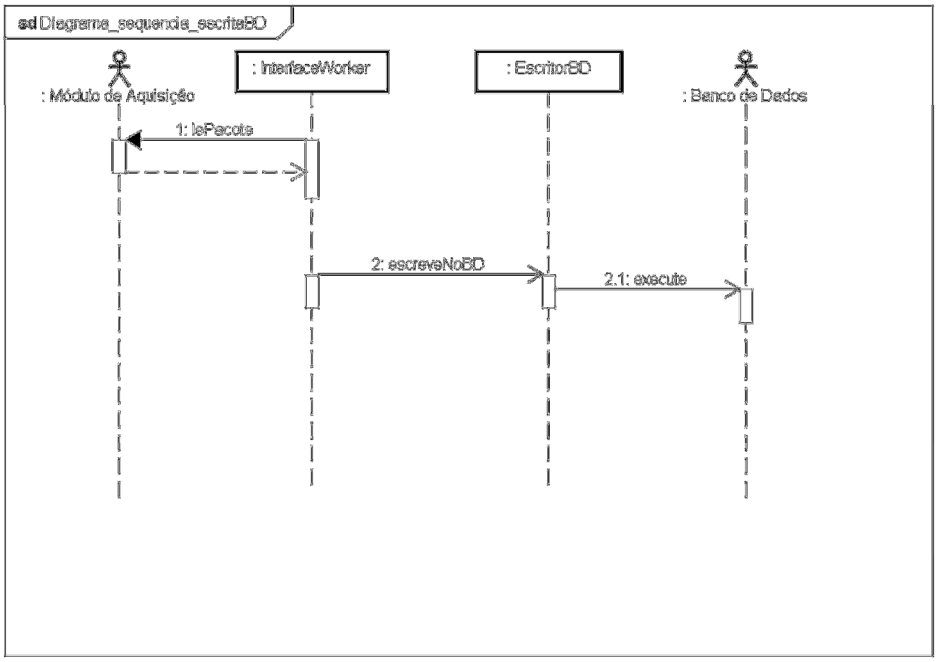


Figura 20 – Diagrama de sequência: escrita de dados no BD.

Já o processo de leitura dos dados do banco começa com o usuário clicando no botão "Carregar dados do banco", exibido na Figura 14. Um objeto da classe InterfaceLeitorBD, do módulo de interface, é instanciado. Este objeto instancia, por sua vez, um objeto da classe LeitorBD. Então, o objeto InterfaceLeitorBD pede ao LeitorBD a lista de endereços das máquinas que contém dados no banco. O LeitorBD repadassa o pedido para o banco, através de uma consulta SQL. Quando o resultado volta para o InterfaceLeitorBD, este mostra uma tela ao usuário, apresentada pela Figura 21. Se não houver registro de nenhuma máquina no banco, uma mensagem de erro é exibida.

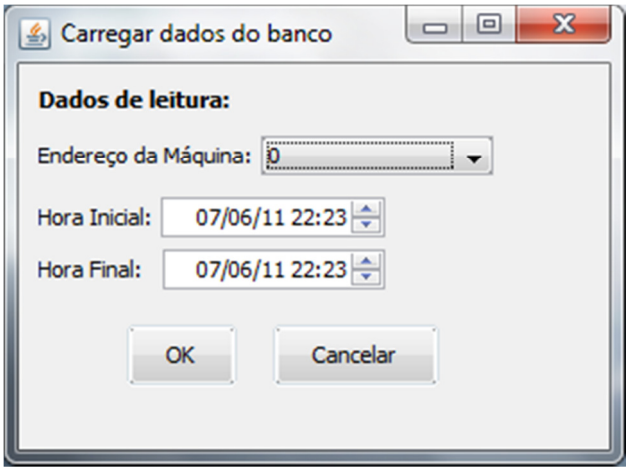


Figura 21 – Tela "Carregar dados do banco"

Desse modo, o usuário poderá escolher uma máquina do banco e os limites iniciais e finais da leitura que deseja recuperar do banco. Ao clicar em OK, o InterfaceLeitorBD manda nova mensagem para o LeitorBD, que realiza uma nova consulta no banco. Se existir valores para o intervalo de tempo inserido pelo usuário, estes são repassados para o InterfaceLeitorBD, que cria um gráfico apresentando os valores lidos, como mostra a Figura 22. Caso contrário, uma mensagem de erro é mostrada.

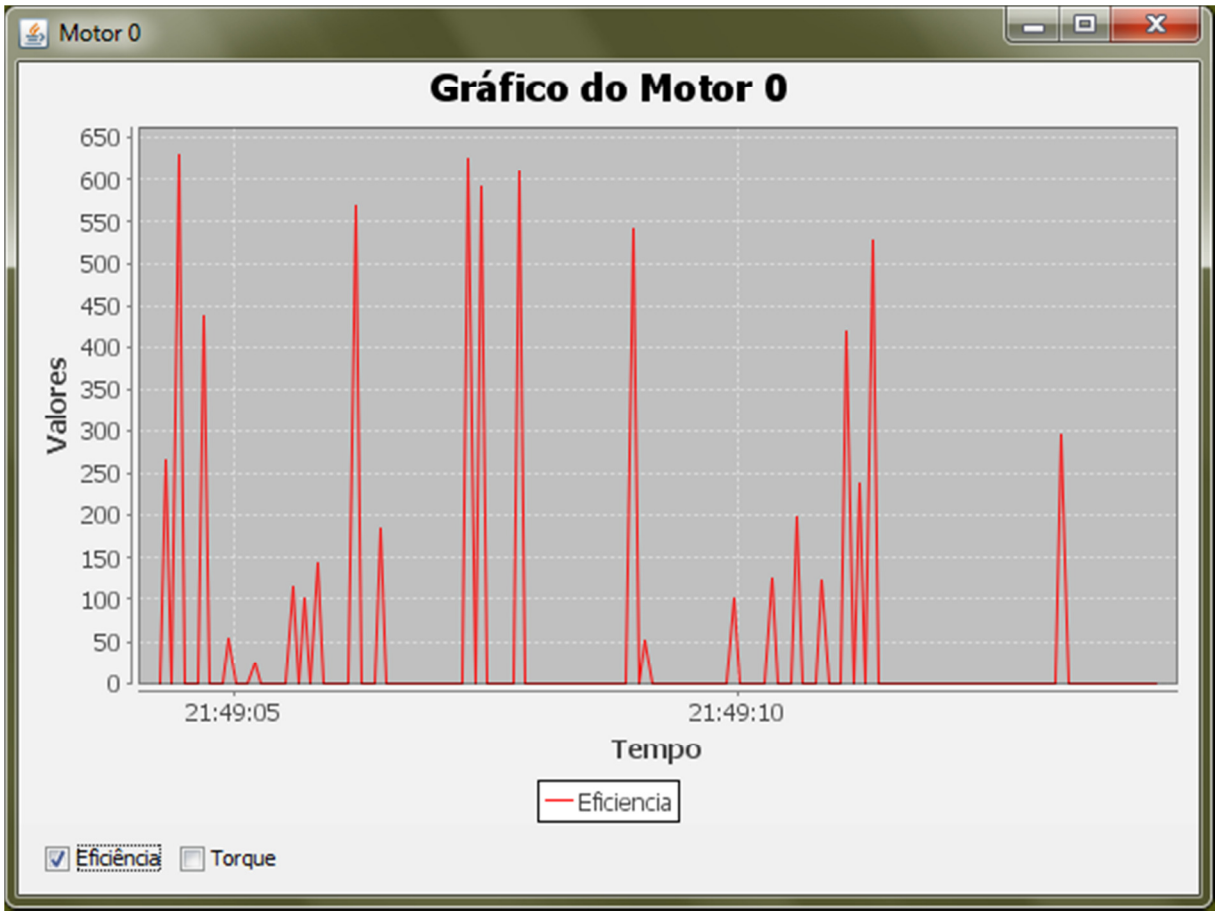


Figura 22 – Gráfico de leitura recuperada do banco de dados.

Os botões do tipo checkbox mostrados abaixo na tela permitem ao usuário selecionar apenas uma das grandezas que ele deseja visualizar. Na Figura 22, por exemplo, temos um teste com uma aquisição de valores aleatórios para Eficiência que haviam sido salvos anteriormente no banco de dados.

O diagrama de sequência apresentado pela Figura 23 mostra o fluxo de execução durante uma bem sucedida leitura de dados do banco:

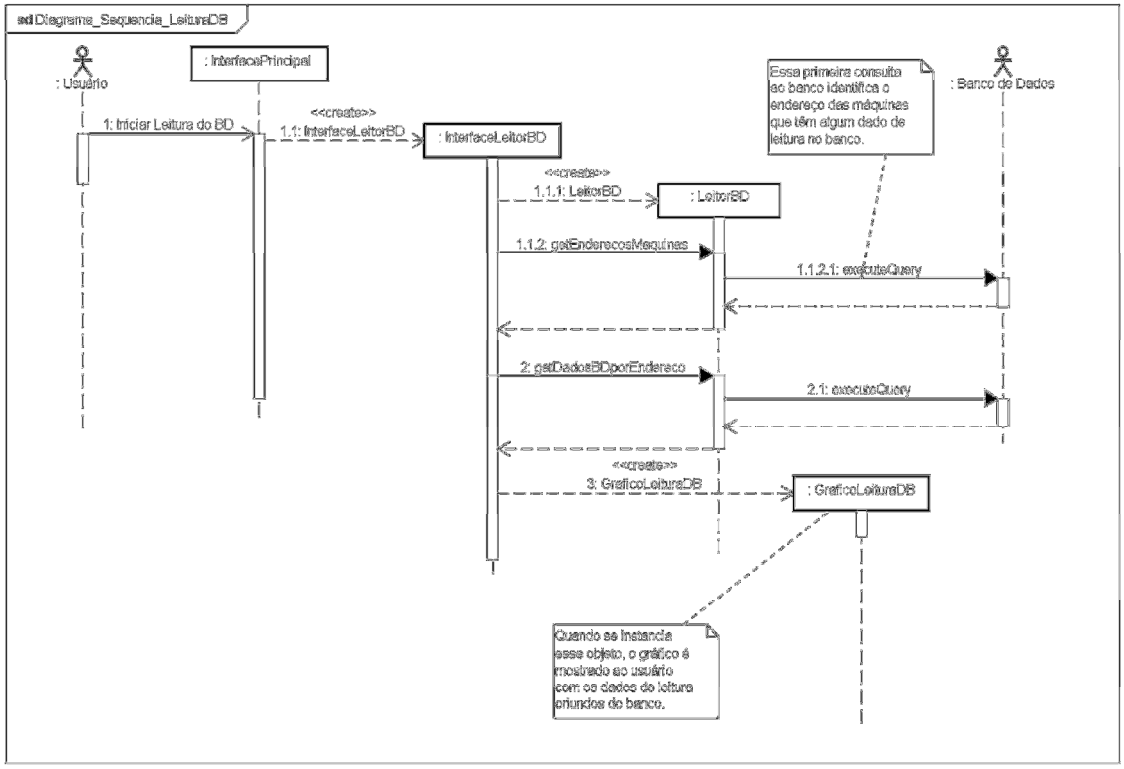


Figura 23 – Diagrama de sequência: Recuperação de dados do banco

6. RESULTADOS

O sistema apresentado passou por vários testes de software durante seu desenvolvimento, com o intuito de comprovar sua eficiência e robustez. As técnicas de teste utilizadas foram as mais comuns, como técnicas de caixa-preta e caixa-branca (MYERS, 2004).

Os testes foram realizados em todos os módulos, em separado e integrados. Dentre os resultados, convém destacar o do módulo de aquisição. O seu teste de caixa-preta comprovou que este módulo ignora cabeçalhos errados. O módulo de aquisição só começa, de fato, a ler um pacote quando este possui o formato de cabeçalho correto. O teste foi realizado dando como entrada ao módulo de aquisição, vários pacotes com cabeçalhos incorretos.

Além dos testes usuais, foram realizadas duas simulações. A primeira procura demonstrar a robustez do sistema quanto à sua capacidade de leitura de uma grande quantidade de pacotes. Já a segunda procura testar o sistema em campo, simulando a rede de sensores descrita na seção 2.2. A seguir detalharemos cada uma das simulações.

6.1 Simulação 1

Nesta simulação foi necessário fazer uma modificação no módulo de aquisição do sistema. A classe que lê os dados da porta serial foi trocada por outra capaz de se conectar a um servidor através de sockets, bastando informar o endereço IP do mesmo.

Foi também desenvolvido um programa servidor que simula a captura de dados em máquinas e a subsequente geração de pacotes. O usuário pode, inclusive, aumentar ou diminuir o número de máquinas simuladas, conforme a necessidade.

Cada máquina neste programa simulador é uma thread que gera um pacote contendo: um cabeçalho no formato proposto do protocolo, um endereço simples e fixo e valores aleatórios para as grandezas medidas de eficiência e torque. Uma máquina simulada foi programa para enviar, aproximadamente, 10 pacotes por segundo.

O módulo de aquisição do sistema se conecta, então, a este programa simulador, através de uma rede TCP/IP. Com isso, foi possível simular o funcionamento do sistema usando uma rede sem fio comum, usando dois computadores distintos.

A Figura 24 mostra a tela do programa simulador, em funcionamento:

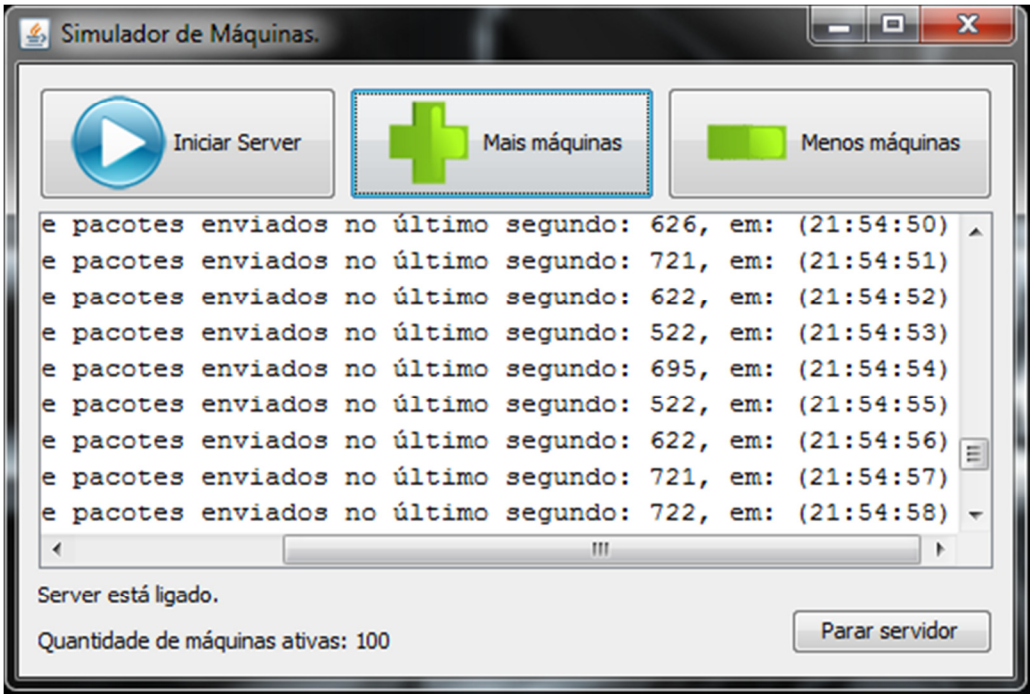


Figura 24 – Tela do Programa de Simulação de Máquinas.

O módulo de aquisição original deve ler dados de uma porta serial, configurada com uma taxa de transmissão de 57600 bps, ou 7200 bytes por segundo, de forma extrapolada. Como cada pacote possui 11 bytes no total, temos uma taxa máxima de 654 pacotes por segundo.

No teste realizado, o programa servidor simulou o funcionamento de 100 máquinas simultâneas, chegando a uma taxa total de 750 pacotes por segundo, em média. O sistema proposto conseguiu a mesma taxa média de leitura de pacotes, mesmo com a escrita no banco de dados habilitada. Este resultado sugere que a perda de pacotes foi praticamente nula.

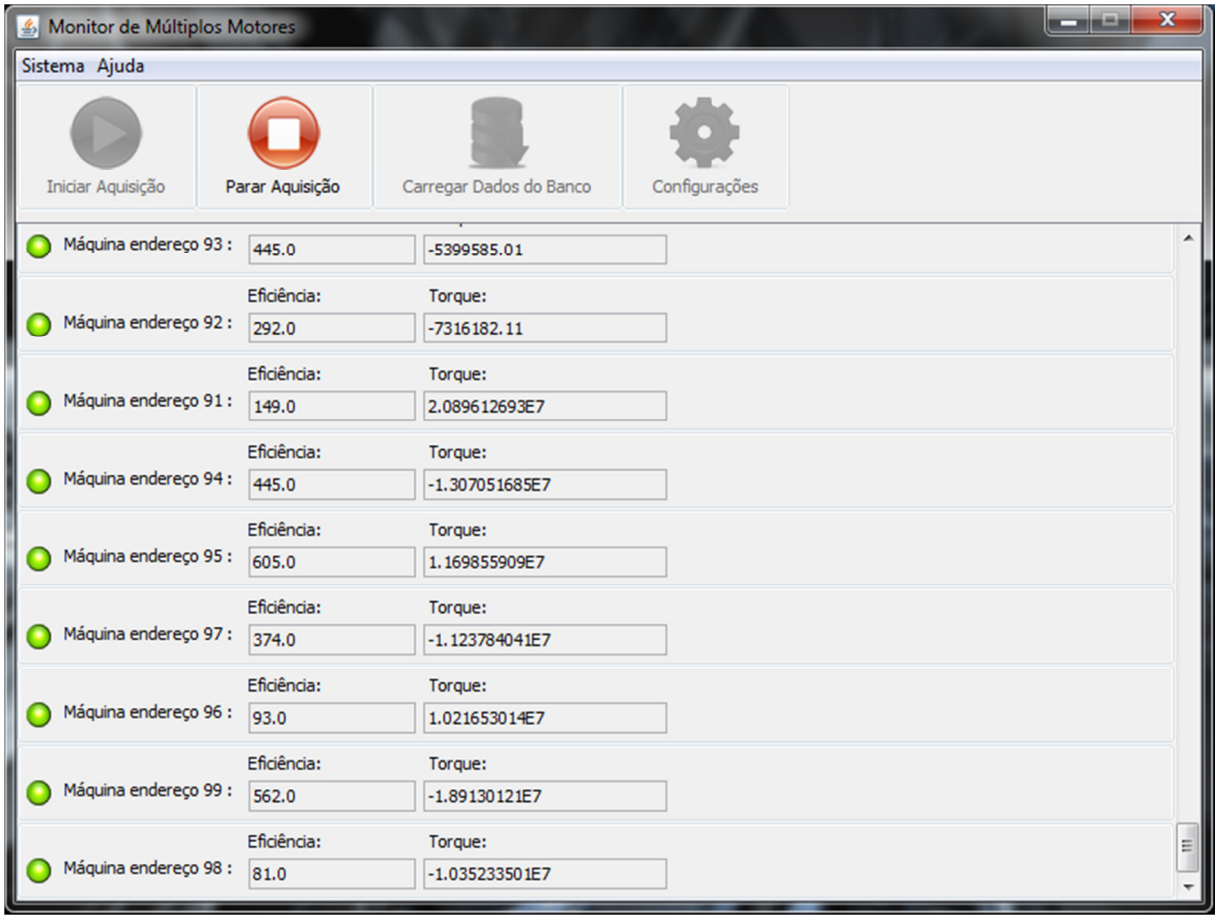


Figura 25 – Simulação de Leitura.

Na Figura 25, pode-se observar o sistema em funcionamento, realizando a leitura de pacotes com valores aleatórios de eficiência e torque das 100 máquinas virtuais geradas pelo simulador descrito acima. As máquinas virtuais possuem endereços na faixa de 0 a 99 neste exemplo.

6.2 Simulação 2

Este outro experimento procurou medir o desempenho do sistema em um ambiente mais próximo daquele para o qual foi projetado. Com a ajuda de 3 dispositivos ZigBee, sendo um nó coordenador e 2 nós finais, foi simulado a leitura de dados de duas máquinas distintas. O experimento se configurou de acordo com a Figura 26:

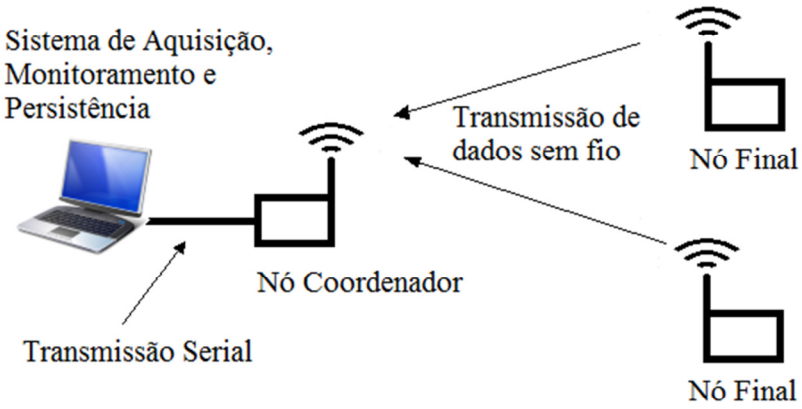


Figura 26 – Simulação Usando Dispositivos ZigBee.

O sistema se comunica com o nó coordenador através de uma conexão serial. Já os nós finais foram configurados para transmitirem, continuamente, pacotes com valores reais de leitura de eficiência e torque. Estes dados de leitura foram previamente obtidos em testes do sistema descrito em 2.2 e são dados conhecidos, tornando possível conferir os dados exibidos e armazenados pelo sistema proposto.

Mais uma vez, o sistema proposto não apresentou falhas na leitura dos dados. A Figura 27 mostra o sistema no momento da aquisição. Nela, temos a tela principal do sistema ao fundo, mostrando a aquisição em tempo real dos dados enviados pelos dois nós finais. Em destaque, temos dois gráficos de leituras em tempo real para o torque e a eficiência de um dos motores (no caso, um dos nós finais). Esta leitura foi comparada com os valores de teste previamente inseridos nos nós finais. Comprovou-se, então, que os dados eram idênticos, garantindo assim que o sistema proposto é capaz de realizar uma leitura de dados de eficiência e torque de forma correta.

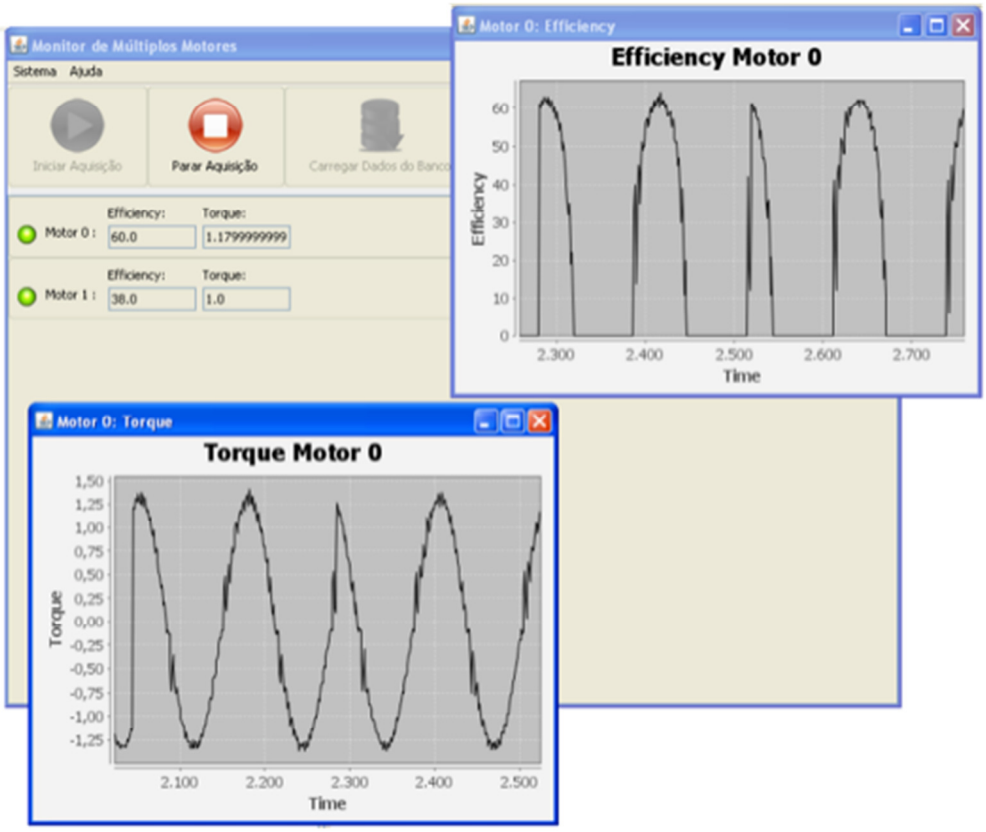


Figura 27 – Aquisição de Dados na Simulação 2.

7. CONCLUSÕES

Este trabalho apresentou um sistema de aquisição, monitoramento e persistência de dados oriundos de uma rede de sensores sem fio. Foi descrito todo o seu funcionamento, com ênfase na divisão das tarefas por módulos.

Os resultados obtidos nos dois testes mostram a eficiência e robustez do software. O primeiro teste deixa claro que o sistema é capaz de ler dados em uma taxa de transmissão de pacotes até mesmo superior ao de uma transmissão serial, utilizada para a aquisição de dados da rede de sensores. Desta forma garante-se que não haverá perdas de pacotes pois a rede de sensores nunca poderá enviar mais dados que o sistema poderá ler.

O segundo teste comprovou que os dados adquiridos pela rede de sensores são lidos corretamente, sem perdas ou modificações. Desta forma podemos concluir que o uso do sistema proposto em um sistema real de monitoramento wireless da eficiência energética e torque em motores industriais é completamente viável.

O sistema, uma vez construído em módulos, permite ser modificado mais facilmente, podendo ser usado em sistemas de monitoramento diversos.

REFERÊNCIAS

AKYILDIZ, I. F., VURAN, M. C. **Wireless Sensor Networks**, 1 ed. Editora Wiley, 2010. 516p.

BAILEY, D. WRIGHT; E. **Pratical SCADA for Industry.** Oxford: Newnes, 2003. 304p.

CAO L., JIANG W., ZANGH Z. **Network wireless Meter Reading System Based on ZigBee Technology.** IEEE Control and Decision Conference, 2008.

CHANGEVISION. **Astah* community – FREE UML Modeling Tool.** Apresenta características da ferramenta. Disponível em: <http://astah.change-vision.com/en/product/astah-community.html> Acesso em 26 de maio de 2011.

DANEELS, A., SALTER, W. **"What is SCADA?".** International Conference on Accelerator and Large Experimental Physics Control Systems, Trieste, Italy. 1999.

DEITEL, H. M. **Java, Como Programar.** Porto Alegre: Bookman, 2003. 1386p.

FETTE, B.; AIELLO, R.; CHANDRA, P.; DOBKIN, D.; BENSKY, D; MIRON, D.; LIDE, D.; DOWLA, F.; OLEXA, R. **RF & Wireless Technologies: Know It All**. Elsevier, 2007.

GOMES, R. D., SPOHN M. A. **Estudo Experimental da Utilização Espectral na Banda ISM de 2.4 GHz**. 9th International Information and Telecommunication Technologies Symposium, 2010.

GILBERT, D. **The JFreeChart Class Library – Developer Guide Version 1.0.4.** Hertfordshire: Object Refinery Limited, 2007. Disponível em: <http://www.jfree.org/jfreechart/devguide.html> Acesso em 25 de maio de 2011.

GUNGOR, V. C, HANCKE G. P. **Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches.** IEEE Transactions on Industrial Eletronics, 2009.

KARL, H., WILLIG, A. **Protocols and Architectures for Wireless Sensor Networks**. Editora Wiley, 2005.

KESTER, W. **"A grounding philosophy for mixed-signal systems".** Electronic Design (special analog issue), June 23, 1999. pp. 29-38.

LEA, D. **Concurrent Programming in Java: Design Principles and Patterns.** 2ª edição, Editora Addison-Wesley, 1999.

LIAW, H.-J., MERKELO, H. **"Signal integrity issues at split ground and power planes"**, *Proc. IEEE EPTC,* May,1996. pp. 752-755.

LOUREIRO, A. A. F., NOGUEIRA, J. M. S., RUIZ, L. B., de FREITAS MINI, R. A., NAKAMURA, E. F., FIGUEIREDO, C. M. S. **Redes de sensores sem fio**. In Simpósio Brasileiro de Redes de Computadores, 2003.

LU, B. GUNGOR, V. C. **Online and Remote Motor Energy Monitoring and Fault Diagnostics Using Wireless Sensor Networks.** IEEE Transactions on Industrial Electronics, 2009.

LU, B., HABETLER, T. G., HARLEY R. G. **A Nonintrusive and In-Service Motor-Efficiency Estimation Method Using Air-Gap Torque With Considerations of Condition Monitoring.** IEEE Transactions on Industry Applications, 2006.

MOONGILAN, D.**"Grounding optimization techniques for controlling radiation and crosstalk in mixed signal PCBs"**, *Proc. IEEE EMC Symp.*, Aug, 1998. vol. 1, pp. 495-500.

MYERS, G. J. **The Art of Software Testing.** 2ª edição. Editora John Wiley & Sons, 2004.

OMG. **Introduction to OMG´s UML.** Apresenta uma introdução à UML. Disponível em: <http://www.omg.org/gettingstarted/what_is_uml.htm> Acesso em 31 de maio de 2011.

ORACLE. **Worker Threads and SwingWorker.** Apresenta informações sobre o uso da classe Swing Worker na execução de tarefas de longa duração. Disponível em: <http://download.oracle.com/javase/tutorial/uiswing/concurrency/worker.html> Acesso em 27 de maio de 2011.

ORACLE CORPORATION. **NetBeans IDE – Features**. Apresenta características do software. Disponível em: <http://netbeans.org/features/index.html> Acesso em 26 de maio de 2011.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreeSQL: About.** Apresenta características do banco de dados relacional. Disponível em: <http://www.postgresql.org/about/> Acesso em 26 de maio de 2011.

PRECHELT, L. **Comparing Java vs. C/C++ efficiency differences to inter-personal differences.** Communications of the ACM, 1999.

SALVADORI, F., CAMPOS, M., SAUSEN, P. S., CAMARGO, R. F., GEHRKE, C., RECH, C., SPOHN, M. A., OLIVEIRA, A. C. **Monitoring in Industrial Systems Using Wireless Sensor Network With Dynamic Power Management.** IEEE Transactions on Instrumentation and Measurement, 2009.

SANTOS, J. L. A. **Sistema telemétrico para monitoramento de trens através de redes de sensores sem fio e processamento em sistema embarcado.** Dissertação de Mestrado, PPGI/UFPB, João Pessoa. 2010.

ANEXO A – DIAGRAMAS UML DO SISTEMA

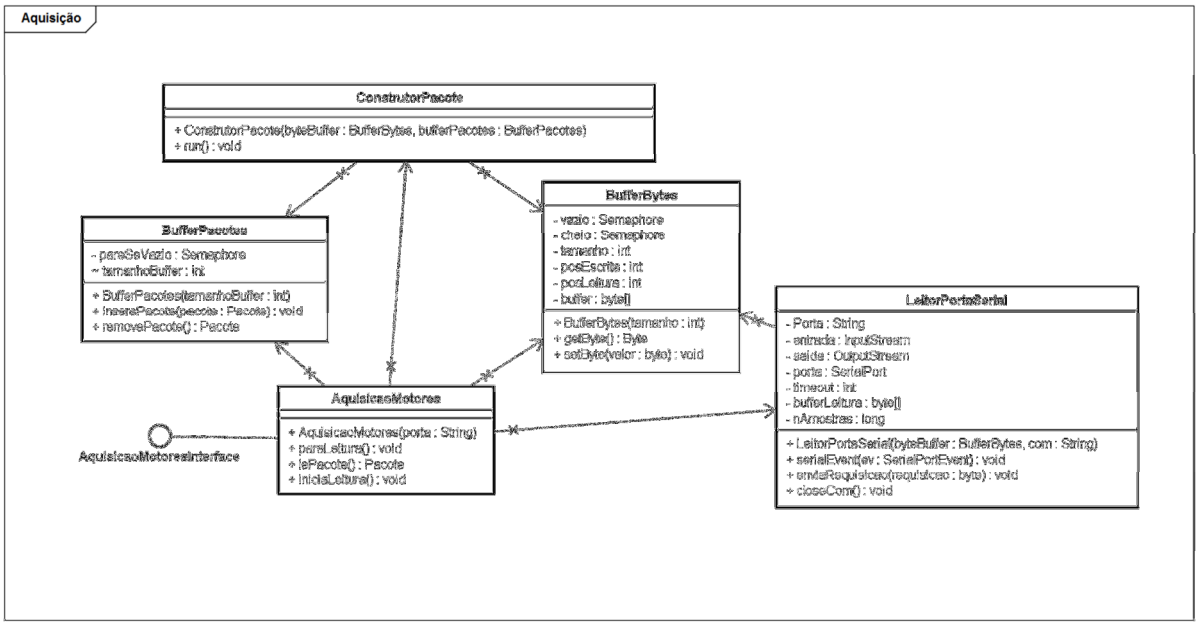


Diagrama de Classes do Módulo de Aquisição

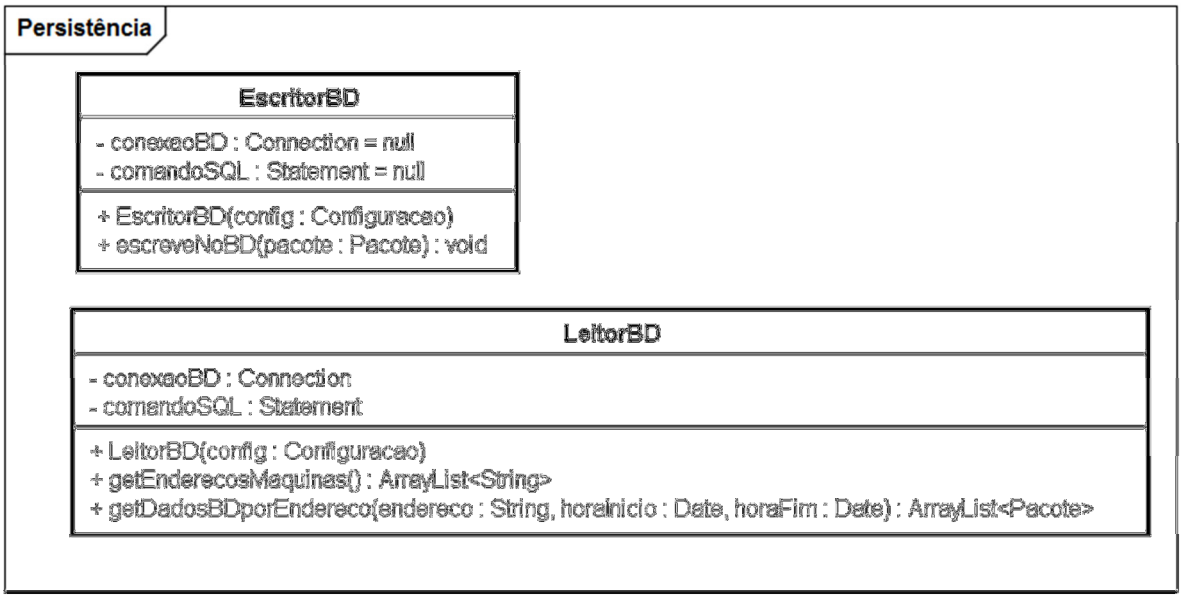


Diagrama de Classes do Módulo de Persistência

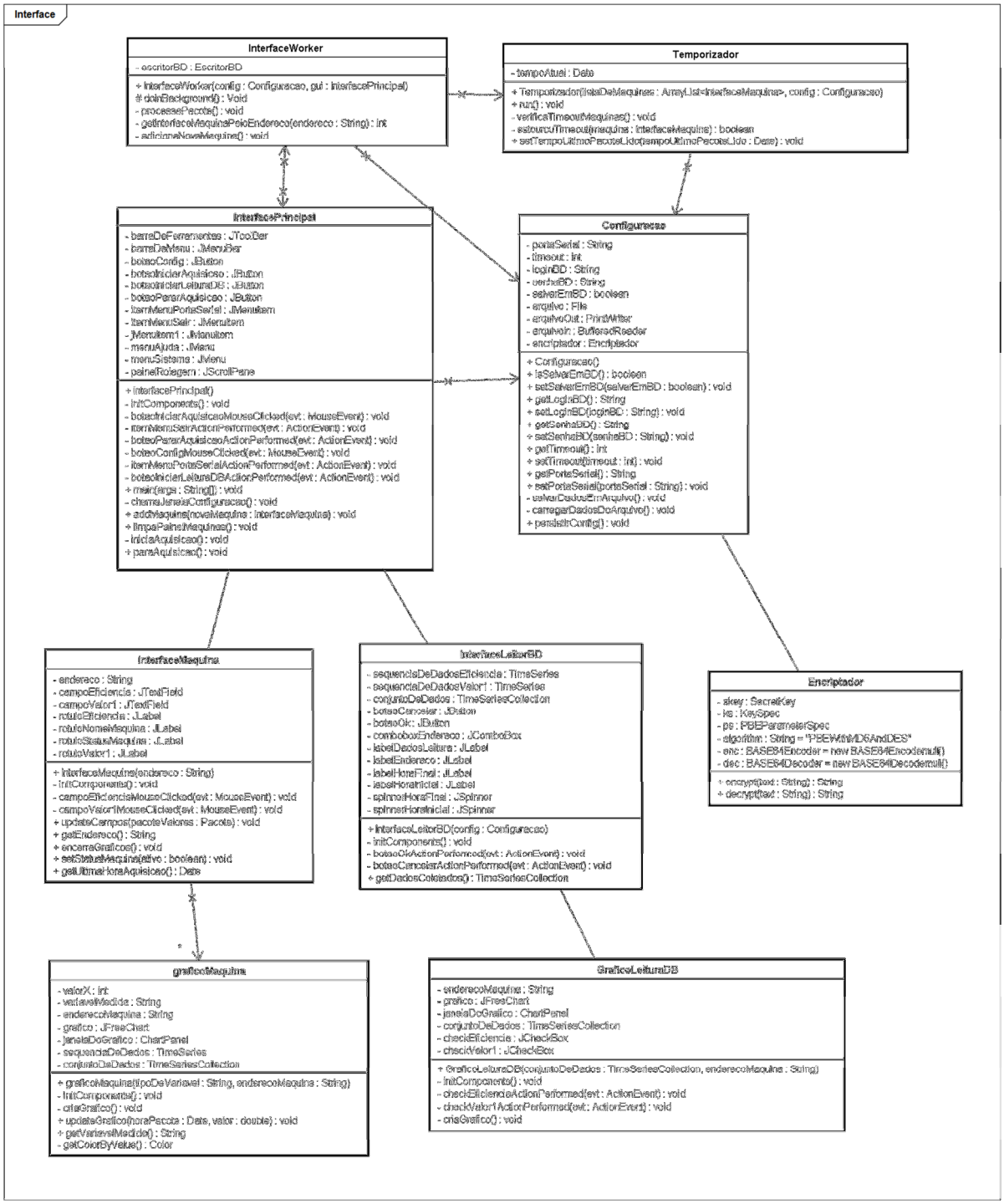
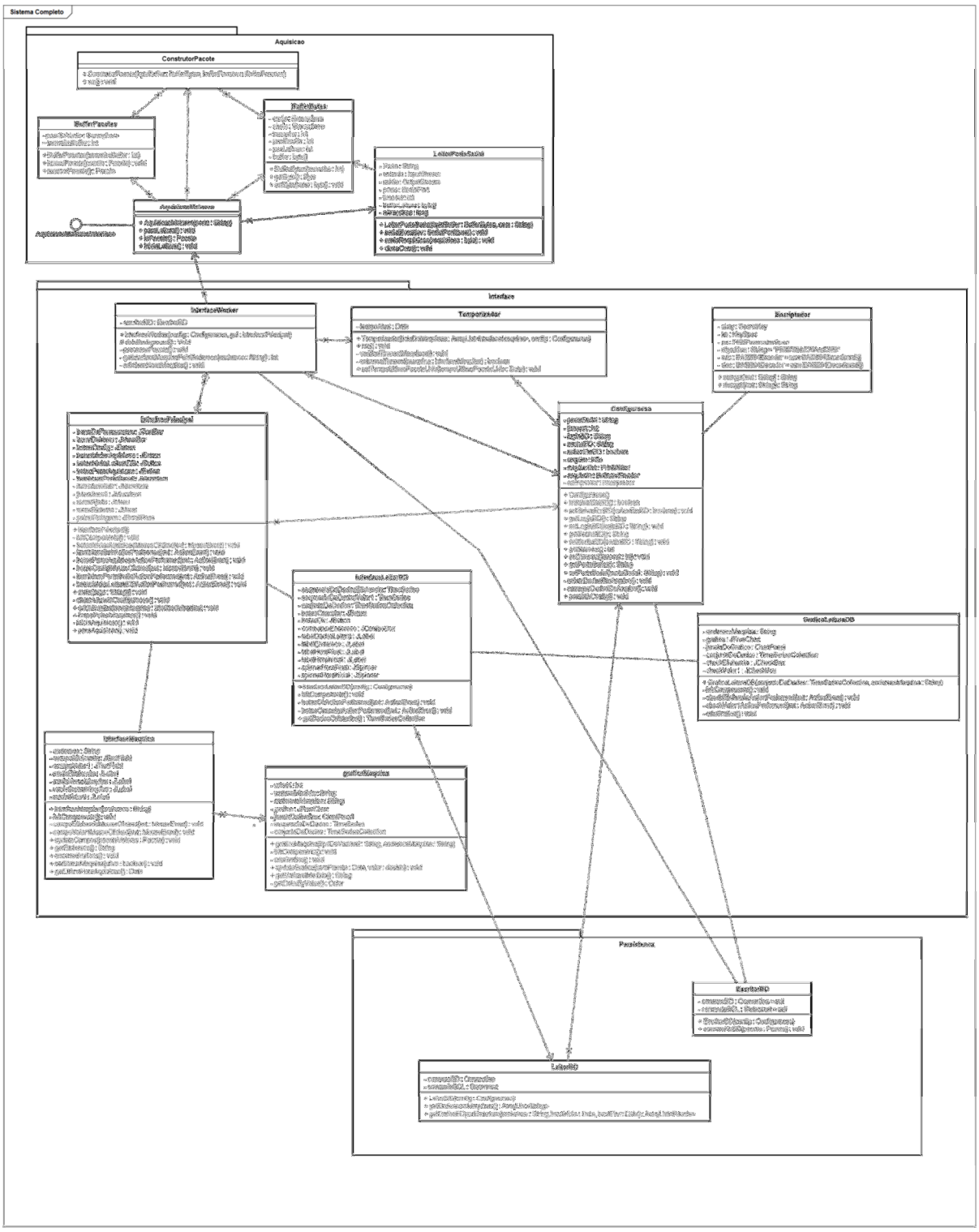


Diagrama de Classes do Módulo de Interface



Diagramas de Classes do Sistema

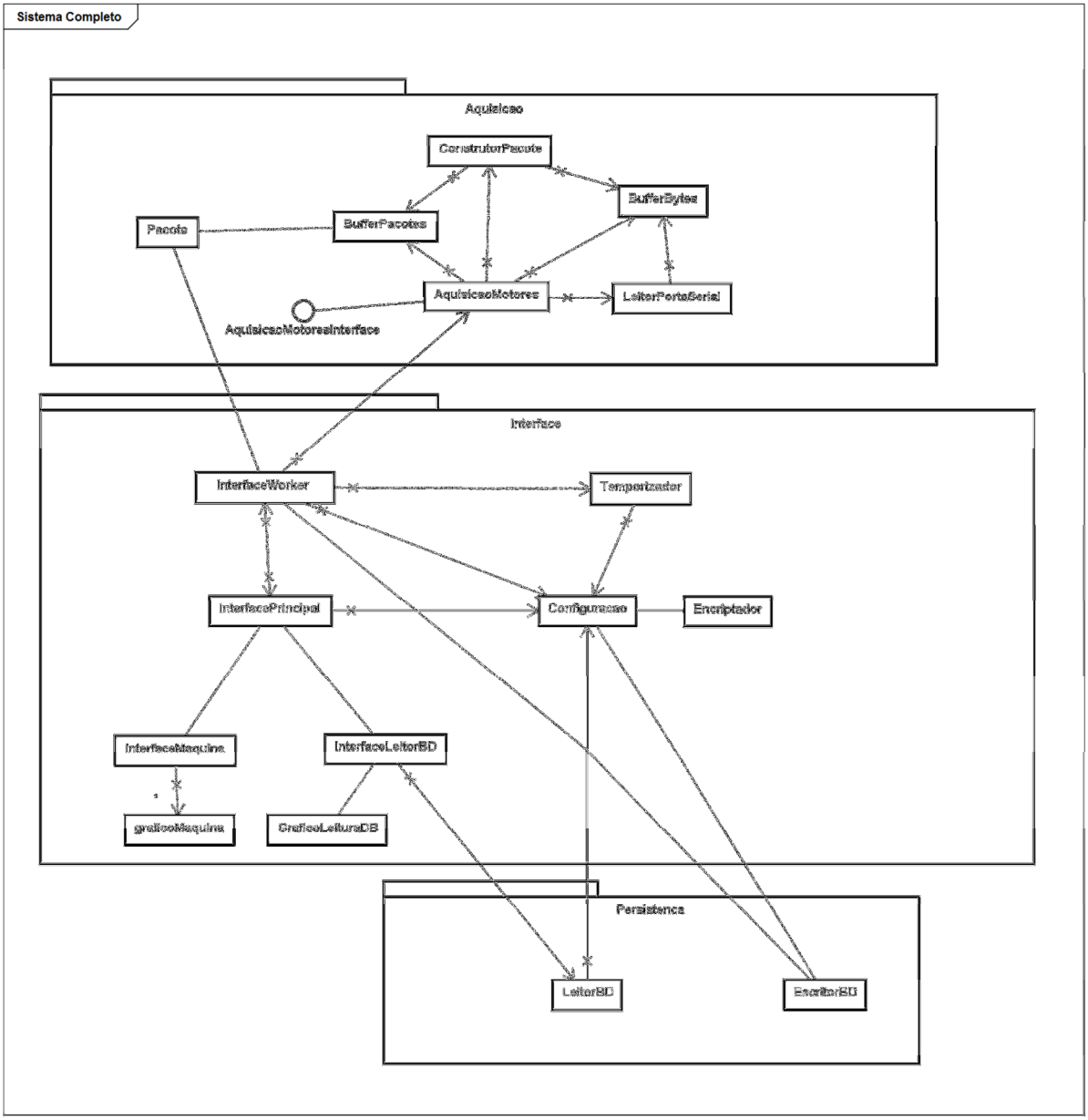


Diagrama de Classes do Sistema (Simplificado)