Memorias del XVI Congreso Latinoamericano de Control Automático, CLCA 2014
Octubre 14-17, 2014. Cancún, Quintana Roo, México

Desenvolvimento de um Framework para Aquisição, Distribuição, Visualização e Armazenamento de Dados em uma Rede de Sensores sem Fio

Filipe S. Santana, Thiago C. Jesus

Laboratório de Sistemas Embarcados (LSE/UEFS)
Departamento de Tecnologia
Universidade Estadual de Feira de Santana
Feira de Santana, BA, Brasil.
(e-mail: filipe.sbr@gmail.com; jesus@ecomp.uefs.br).

Resumo: Várias aplicações compartilham a necessidade da utilização de um sistema integrado de sensoriamento, desde a aquisição de dados, até o armazenamento e exibição dos mesmos. Assim, este trabalho apresenta a modelagem de um *framework* multicamadas e genérico voltado para aquisição de dados de uma rede de sensores sem fio. Com esse *framework* é possível agilizar o processo de desenvolvimento de uma aplicação *hardware/software* para aquisição, distribuição, visualização e armazenamento de dados. A implementação do *framework* passa por três camadas: a de aquisição, que é a camada de *hardware* responsável pela leitura de sensores; a de aplicação, que recebe dados provenientes da camada de aquisição e pode exibi-los e armazená-los; e a camada de comunicação, que garante a troca de mensagens entre as camadas de aquisição e aplicação através do protocolo *ZigBee*(IEEE 802.15.4).

Keywords: Aquisição de Dados, framework, ZigBee, Redes de Sensores sem Fio.

1. INTRODUÇÃO

Automatização de processos e produtos é uma necessidade cada vez mais urgente em uma sociedade que vem conciliando tecnologia e inovação. Para tanto, é natural o desenvolvimento de uma metodologia na qual seja realizada uma obtenção de dados acerca do objeto de automação, para então tomar-se decisões baseadas nesses dados. Isso ocorre em sistemas de controle, estações meteorológicas, na indústrias, em sistemas inteligentes de tranporte e sistemas de segurança, por exemplo (Mahjoubi et al., 2011; Benghanem, 2010; Vellingiri et al., 2013; Wang et al., 2014; Nguyen and Khan, 2013).

Todas essas aplicações compartilham a necessidade da utilização de um sistema integrado de sensoriamento. Essa funcionalidade comum a várias aplicações motiva o desenvolvimento de *frameworks* de sensoriamento.

Um *framework* para aquisição de dados em rede, focado na camada de aplicação, é apresentado por Nguyen and Khan (2013). Esse *framework* fornece um mecanismo sensível ao contexto para controlar a aquisição de dados em redes de sensores. Assim é possível aumentar a qualidade da informação para toda a rede.

Nagasaka and Motoyama (2007) também apresentam um *framework* para aquisição de dados em rede, voltado a aplicações de propósito geral, porém focado apenas na camada de comunicação, fornecendo um espaço de dados compartilhado que pode ser acessado por comandos simples como *PUT*, *GET*, *DELETE*, *NOTIFY* e *LINK*. Mershad and Artail (2013) também apresentam um *framework* para resolver problemas na camada de comunicação de tais sistemas. Nesse caso, é proposta uma solução que garante privacidade e segurança dos dados em uma rede *Ad Hoc* veicular.

A camada de comunicação de um sistema de aquisição de dados desempenha um papel crucial, pois é ela a encarregada de determinar a escalabilidade do sistema (quantidade de elementos sensores), a velocidade e qualidade de transmissão, além do roteamento dos dados até seu destino. No contexto de transmissão de dados surge o conceito de Redes de Sensores sem Fio (RSSF).

RSSF vem introduzindo uma revolução na forma de monitorar e controlar diversos ambientes, possuindo aplicações nas mais diversas áreas econômicas e militares: monitoramento de tráfico, diagnósticos em indústrias, distribuição de água, automação residencial, dentre outras. Os avanços nas áreas de redes sem fio, microcontroladores e sensores abriram um leque de possibilidades para desenvolvimento de RSSF, tornando comum a implementação de tecnologias para agilizar a concepção de RSSF (Guibas and Zhao, 2004).

Pensando nas abordagens tradicionais de projeto de RSSF, Kiepert and Loo (2012) propõem e descrevem os componentes necessários de um *framework* unificado para RSSF, como portabilidade de *hardware*, sistema operacional embarcado, execução multi-tarefa, processamento paralelo, processamento em rede, sincronia de mensagens, busca em rede, comunicação tolerante a falhas, armazenamento de

dados, etc. Esse *framework* unificado dá suporte à maioria das atuais aplicações, bem como provê suporte para novos avanços na área.

Nesse contexto, este trabalho propõe a implementação de um *framework* para RSSF contendo os principais elementos sugeridos por Kiepert and Loo (2012), preenchendo uma lacuna da literatura, abordando aspectos práticos do desenvolvimento de tal sistema. Esse *framework* fornece um conjunto de funcionalidades para facilitar e agilizar o desenvolvimento de uma aplicação de aquisição, armazenamento, tratamento e visualização de dados em um RSSF. É apresentada uma arquitetura baseada em uma camada de *hardware* microcontrolada com interface para sensores, uma camada de aplicação que facilite o armazenamento, tratamento e visualização de dados, bem como uma camada de comunicação sem fio entre as duas primeiras camadas. Esse *framework* é implementado de forma genérica para que possa ser utilizado em aplicações de aquisição de dados de propósito geral.

Este trabalho está estruturado da seguinte forma: na seção 2 é apresentada uma formulação do problema de aquisição de dados tratado e uma breve descrição do *framework*, esclarecendo nomenclatura e estabelecendo algumas noções preliminares. Na seção 3 é apresentada a metodologia de desenvolvimento do sistema, enfatizado na modelagem da arquitetura do *framework*. Uma discussão sobre o desenvolmiento do sistema é apresentada na seção 4. Finalmente, na seção 5, as conclusões acerca da implementação são apresentadas

2. NOÇÕES PRELIMINARES E REQUISITOS

Um sistema de aquisição de dados, disposto em uma RSSF, é composto por diversos nós sensores espalhados em um ambiente, próximo do fenômeno a ser observado por estes sensores. Estes sensores são componentes que traduzem uma informação do "mundo real" (temperatura, pressão, umidade, dentre outras informações) em um sinal que pode ser processado e analisado (tensão elétrica, por exemplo). É necessário uma camada de *hardware* para adquirir os dados de interesse (microcontroladores, FPGAs, Controladores Lógico Programáveis ou DSPs), componentes de energia (baterias), de transmissão de dados (rádio), e eventualmente pode existir uma memória para armazenar dados coletados. Uma vez que se tenha adquirido os dados, é natural que se queira usá-los para gerar estatíticas e auxiliar na tomada de decisões. Neste caso, um sistema de aquisição pode fornecer uma camada de aplicação para armazenamento, tratamento e visualização desses dados em um ponto remoto. Por consequência, é preciso estabelecer a troca de dados entre a camada de *hardware* e a camada de aplicação, que é garantida por uma terceira camada - a de comunicação.

RSSF possuem características peculiares, como operação em baixa potência, grande número de elementos (nós) e roteamento dinâmico de mensagens (diversos caminhos para chegar ao mesmo destino). Devido à essas características, se faz necessário o uso de um padrão específico e diferenciado para a comunicação; assim, definindo as camadas físicas e de controle de acesso (modelo OSI), surge o padrão IEEE 802.15.4, sendo este uma tecnologia de relativo curto alcance, baixa complexidade, baixo custo, baixo consumo de energia, e a possibilidade de transmissão de taxas de dados pequenas (Buratti, 2011). Após o surgimento desse padrão, foi criado pela ZigBee Alliance o padrão ZigBee que define as camadas de rede e de aplicação, protocolando ainda topologias de rede, mecanismos de entrada e saída de dispositivos da rede, segurança e roteamento.

Com os diversos sistemas de sensoriamento nas áreas de agricultura, transportes, meteorologia e segurança; e, atualmente, com todos estudos, avanços e grandes possibilidades providas pelas Redes de Sensores Sem Fio, motiva-se a criação de um *framework* para agilizar o desenvolvimento e implantação de tais sistemas.

Um *framework* consiste de uma coleção abstrata de classes, interfaces e padrões dedicada a resolver uma classe de problemas através de uma arquitetura flexível e extensível (Govoni, 1999). Uma das vantagens do desenvolvimento de um *framework* é a criação de um repositório de modelo de funcionalidades que guiará o processo de produção de sistemas inteiros, nesse caso, sistemas de aquisição de dados em uma RSSF (Zhao et al., 2013). Desta forma, este trabalho se propõe a desenvolver o *firmware* para os nós da rede. Cada nó é baseado em um microcontrolador que garante a obtenção de dados de até oito sensores analógicos, além da transmissão desses dados para uma aplicação remota em *software*. O *framework* desenvolvido provê funcionalidades para coleta de dados por um microcontrolador, bem como configurações do mesmo; e envio de dados remotamente através do protocolo ZigBee. Também fornece um *software* desenvolvido em linguagem JAVA para visualização, tratamento e armazenamento dos dados.

Esse *framework* é multicamadas, simples e flexível. Assim, ele provê uma interface de aquisição de dados indepentende dos sensores, uma aplicação para gerência da aquisição de dados e visualização desses dados, e uma interface de comunicação entre a interface de aquisição e a aplicação. Desta forma é possível agilizar o processo de aquisição de dados em aplicações diversas e de forma genérica. A figura 1 apresenta o diagrama de estados do comportamento do *framework*, evidenciando o fluxo de execução em cada camada. De forma geral, a camada de aquisição, após inicializada, fica em um estado ocioso esperando receber através da camada de comunicação um comando da aplicação. A camada de aquisição envia uma confirmação de recebimento de comando. Se por alguma falha de comunicação a aplicação não receber essa confirmação dentro de um intervalo de tempo, o comando é reenviado, garantindo a tolerância a falhas na comunicação prevista em (Kiepert and Loo, 2012). Após receber os comandos, a camada de aquisição o decodifica, executa e gera mensagens de retorno, que pode ser dados de leitura de sensor, valores de configuração ou confirmações de operações realizadas com sucesso. Esse retorno é enviado à camada de aplicação, que deve enviar de volta uma confirmação de recebimento. Mais uma vez, pensando na tolerância a falhas, se a camada de aquisição não receber essa confirmação dentro de um intervalo de tempo, a mensagem de retorno é reenviada para a aplicação. No caso desse retorno ser uma leitura de sensor, esse dado pode ser armazenado e exibido em um gráfico.

É importante ressaltar que a execução da camada de aquisição é paralela à execução da camada de aplicação, sendo sincronizadas pelas mensagens enviadas pela camada de comunicação.

Na seção 3 é apresentada a modelagem do *framework*, enfatizando seus componentes e a interação entre eles.

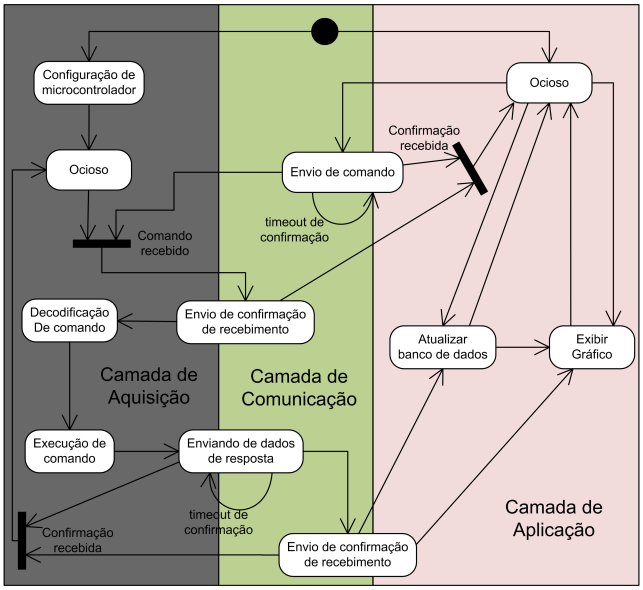

Figura 1. Diagrama de estados do *framework*.

3. ARQUITETURA

A figura 2 apresenta a arquitetura do *framework* em relação às classes que compõe cada camada.

A camada de aquisição é composta por apenas uma classe, a ***DAQ***. Essa classe se encarrega de todas as interações em *hardware* para realizar a aquisição de dados, o que inclui a configuração desse *hardware* para seu funcionamento, como inicialização e ajuste de parâmetros para medições. Essas operações são realizadas mediante comandos provenientes da camada de aplicação, os quais são enviados pela camada de comunicação, bem como as respostas a esses comandos.

A camada de comunicação possui quatro classes: ***HardwareComm***, ***DataReceiver***, ***DataReceiverZigBee***, e ***ZigBeeUtils***. A classe ***HardwareComm*** trata do envio e recebimento das mensagens pelo lado da camada de aquisição. A classe ***DataReceiver*** se encarrega de receber dados provindos da camada de aquisição. A classe ***DataReceiverZigBee*** é somente uma implementação mais específica da classe anterior, pois assim é possível facilmente mudar-se de protocolo no *software*. Por fim, a classe ***ZigBeeUtils*** é responsável por enviar mensagens da camada de aplicação para a camada de aquisição através do protocolo ZigBee.

Por fim, tem-se a camada de aplicação, podendo ser vista na figura 2 (para fins de simplificação, somente o modelo é apresentado, pois além das classes *Node*, *Sensor*, *Network*, e *ZigBeeNetwork*; há também classes de controladores e interface gráfica). Assim, pode-se criar uma abstração da rede nestas quatro classes, sendo que o nó da rede (*Node*)

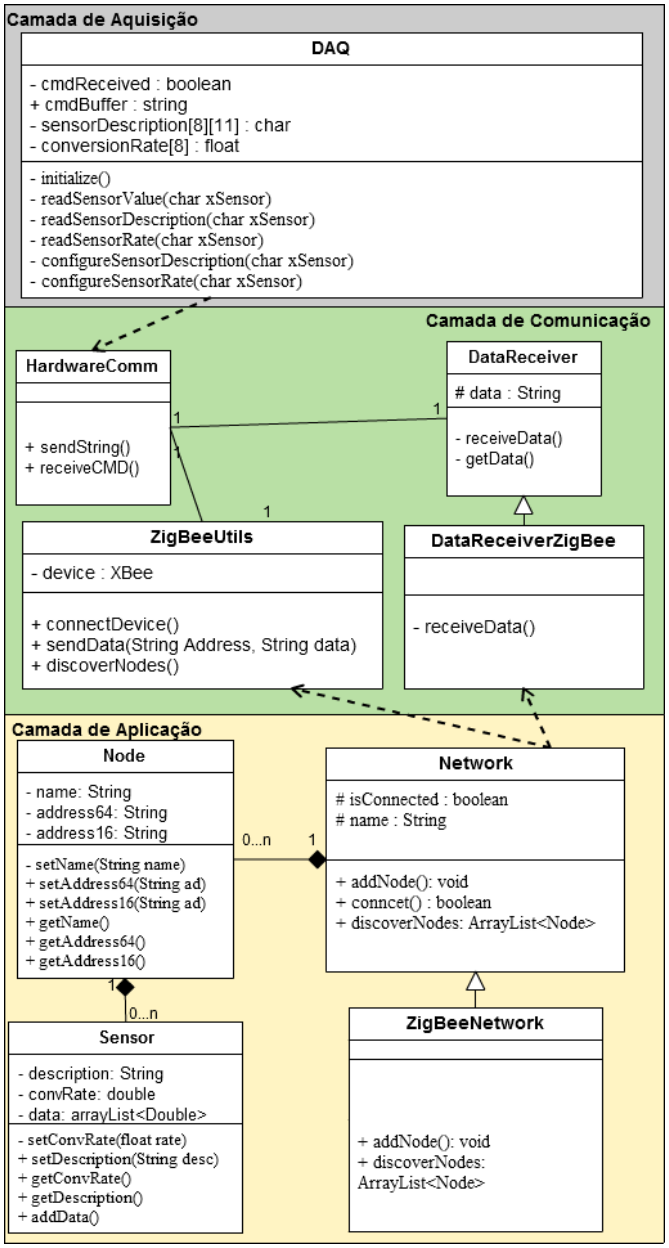

Figura 2. Diagrama de classes do *framework*.

possui vários sensores (*Sensor*) imersos em uma Rede de Sensores Sem Fio ZigBee (*ZigBeeNetwork*). Caso se deseje mudar o tipo de comunicação na rede a ser implementada, basta criar uma nova extensão da classe *Network*. Por exemplo, pensando-se em criar uma rede com comunicação *WiFi*, deveria ser implementada uma classe *WiFiNetwork* que extende a classe *Network*.

Toda a aplicação foi desenvolvida utilizando o padrão de projeto de MVC (*Model-View-Controller*). A abordagem MVC divide as aplicações nessas três faces, em que o *model* consiste nos dados, regras de negócio e lógicas do sistema; o *view* é a apresentação das telas, ou seja, a interface gráfica; e o *controller* define o comportamento de reação desta interface com as entradas fornecidas pelo usuário, e também repassa ao *model* dados provindos de entrada. Assim, o MVC promove a flexibilidade, desacoplamento e reutilização (Erich Gamma, 2000).

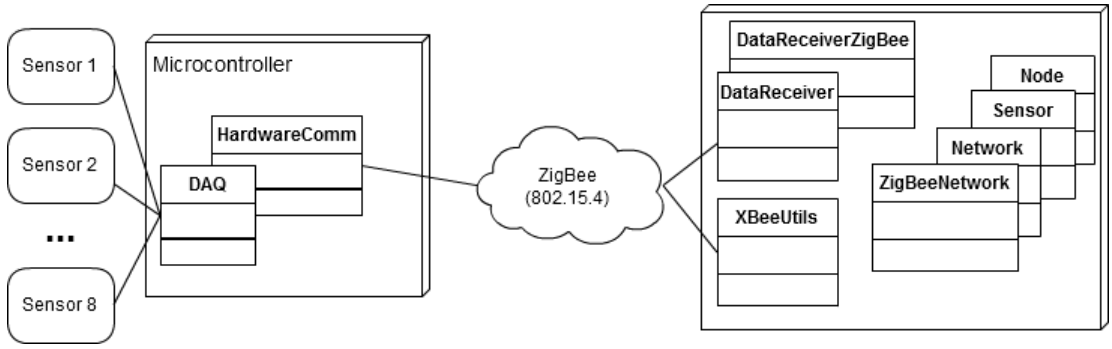

Figura 3. Diagrama de implantação do *framework*.

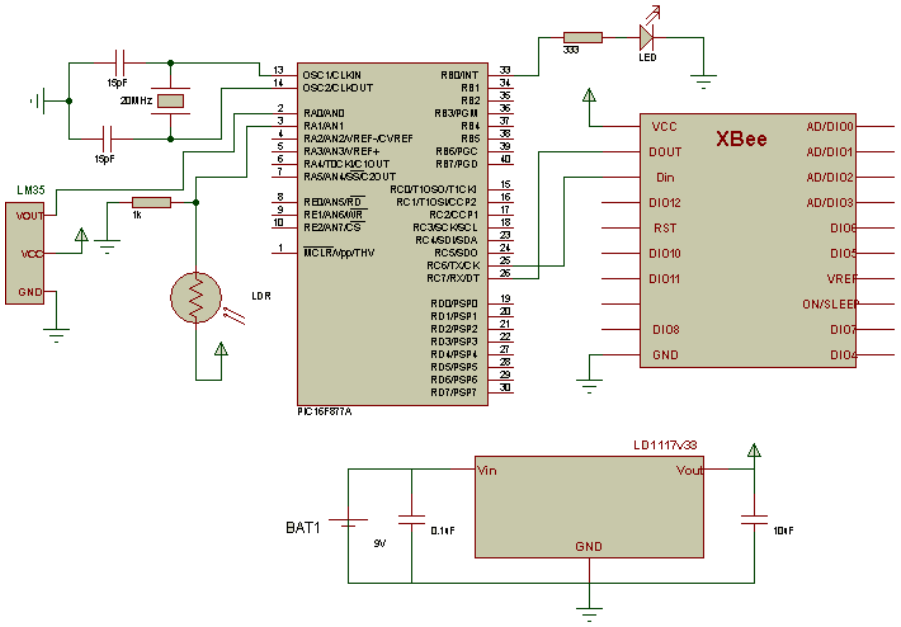

Figura 4. Esquema elétrico dos nós da rede.

Este tabalho implementa o padrão MVC, de forma que todo o projeto é baseado em eventos. Assim, alterações pertinentes no *model* lançam eventos que são capturados e exibidos pelo *view*, gerando as devidas alterações neste último. Ou seja, quando uma mensagem é recebida pela camada de aplicação, os objetos *Node* e *Sensor* sofrem alterações e lançam eventos para a interface gráfica, que é atualizada.

A figura 3 ilustra de forma mais clara a interação dos componentes do *framework*, evidenciando sua disposição física. Essa figura mostra o diagrama de implantação do *framework*, relacionando as classes que executam no sistema.

4. DISCUSSÃO

Analisando a figura 3 fica claro que a classe ***DAQ*** é o núcleo do *firmware* do microcontrolador, realizando todas as interações a nível de hardware. Para tanto, é considerada a arquitetura do microcontrolador PIC16F877A. Foi escolhido um PIC principalmente pelo largo suporte que esse microcontrolador possui em relação a protocolos de comunicação que podem integrar este *framework* em versões futuras, como *Ethernet*, CAN e USB. O PIC16F877A foi escolhido por ser o que apresenta a melhor relação custo vs. benefício: ele é relativamente barato e possui memória de programa (8K), canais analógicos (8 canais), processador (20MHz) e periféricos suficientes para atender os requisitos de *hardware*. Entretanto o *firmware* desenvolvido esgotou os recursos desse microcontrolador, principalmente a memória de programa, inviabilizando a expansão do projeto. Desta forma, alguns nós foram desenvolvidos no PIC18F4550, que possui 32K de memória de programa e possui o protocolo USB nativo.

Essa alteração no projeto foi importante para a comprovação de que o *framework* desenvolvido é genérico e provê a portabilidade de *hardware* prevista em (Kiepert and Loo, 2012), uma vez que o código desenvolvido para o PIC16F877A sofreu pouquíssimas alterações para funcionar no PIC18F4550.

Fisicamente, cada sensor a ser associado ao *framework* deve ser conectado em um dos canais analógicos do microcontrolador, que é previamente configurado para converter sinais analógicos de $0V$ a $3.3V$ em um valor digital codificado em 10 *bits*, ou seja, de "0"a "1023". É importante frizar que o *framework* não fornece condicionamento elétrico do sinal a ser adquirido. Assim, cabe ao usuário

do *framework* garantir a correta interpretação das leituras dos sensores.

Por outro lado, o *framework* permite que o usuário informe uma taxa de conversão para cada sensor. Assim, é possível associar o valor digital lido a um valor de grandeza real, com significado físico associado. Para exemplificar, considere o sensor de temperatura $LM35$, que é comumente utilizado em projetos acadêmicos. Esse sensor fornece um valor de tensão diretamente proporcional à temperatura percebida pelo mesmo, de forma que, para cada grau Celsius medido o $LM35$ fornece $10mV$. Assim, um ambiente a 27^oC vai resultar na emissão de $270mV$, que, em uma escala de $0V$ a $3.3V$ e uma codificação de 10 *bits*, resulta na leitura digital "83". Usando o *framework* é possível configurar a taxa de conversão para esse sensor de "0, 33"e obter assim o valor aproximado em graus Celsius. Esse processo irá gerar muito erro, pois o $LM35$ irá sempre gerar sinais de baixa magnitude. Para leituras digitais mais precisas, é imprescindível um circuito de condicionamento do sinal desse sensor, dando um ganho proporcional em sua saída.

Outro fato que merece atenção é a implementação da classe ***HardwareComm***. Note que a codificação dessa classe também é realizada no microcontrolador. Isso ocorre através do módulo de comunicação USART que implementa uma comunicação serial assíncrona utilizada neste trabalho. Esse mesmo protocolo é utilizado pela classe ***DataReceiver*** com o auxílio da API *RXTX*. A escolha dessa API foi baseada no fato de que a camada de aplicação é desenvolvida em Java, uma linguagem de programação bastante eficiente e segura. Como a comunicação no *framework* é assíncrona, a classe ***HardwareComm*** é implementada baseada em interrupções do microcontrolador, e a classe ***DataReceiver*** é implementada baseada em *threads*. Note que a classe *DataReceiverZigBee* é uma especialização da classe *DataReceiver*, permitindo que o sistema receba dados através de um dispositivo de rádio que implemente ZigBee, e que esteja conectado ao computador.

Para a implementação do protocolo ZigBee foi escolhido o *XBee Series 2* da fabricante *DIGI*, empresa consolidada no mercado de comunicação sem fio. Assim, um nó da rede de sensores sem fio proposto neste trabalho constituise por um microcontrolador, um dispositivo de energia (bateria) e um rádio que implemente o protocolo ZigBee. A figura 4 mostra o esquema elétrico de um nó dessa rede. O microcontrolador se encarrega da aquisição e tratamento dos dados, e da interpretaçãos dos comandos solicitados pelo usuário. O microcontrolador ainda é responsável pela comunicação com o dispositivo XBee, que envia mensagens para a aplicação. No computador em que a aplicação estiver sendo executada deve possuir também um XBee conectado à este. Neste projeto utilizou-se um adaptador Xbee-USB. Através da aplicação é possível solicitar ações a serem executadas por nós específicos da rede, como pode-se ver nas figuras 5 e 6.

Os comandos de ação enviados através da aplicação para um determinado nó da rede permitem: Leitura de dados, configuração e leitura da descrição de sensor, configuração e leitura da taxa de conversão da leitura de sensor; também há o comando para teste de conectividade com determi-

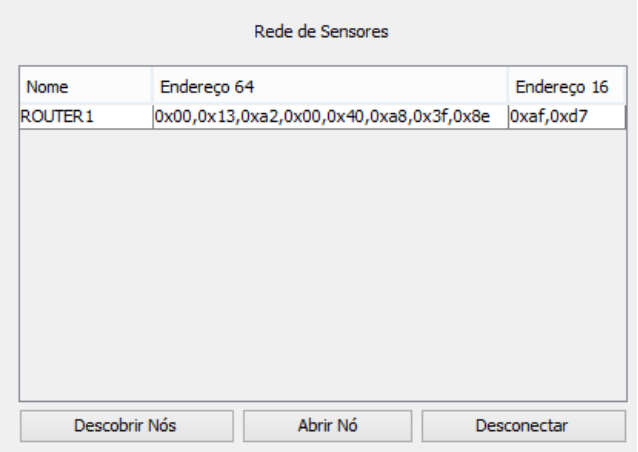

Figura 5. Tela de listagem dos nós da rede.

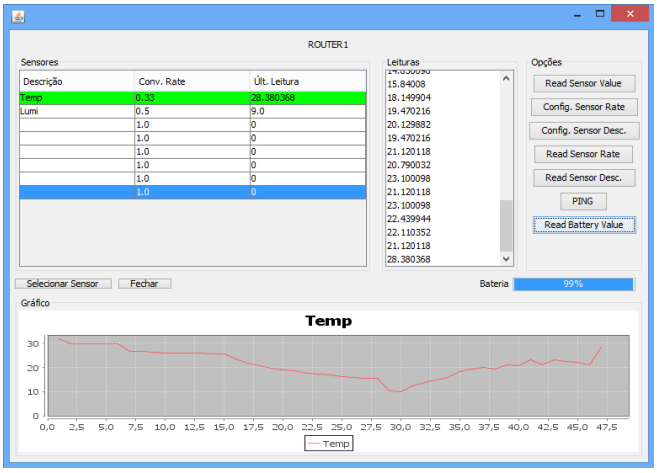

Figura 6. Tela de controle de um nó da rede.

nado nó e leitura da tensão fornecida pela alimentação (verificação da energia fornecida pela bateria). Há ainda as mensagens enviadas no sentido nó-aplicação, sendo: resposta de comunicação, dado lido de sensor, descrição de sensor, taxa de conversão de sensor. Todos estes comandos podem ser vistos na tabela 1.

Tabela 1. Comandos do framework

Comando	Resposta	Descrição
PING	PONG	Teste de comunicação
RSV&X&	DATA&X&NNN.NNNN	Requisição de leitura de sensor X
CSD&X&yyyyyyyyyy&	--	Configurar descrição do sensor X
RSD&X	DESC&X&yyyyyyyyyy&	Ler descrição do sensor X
CSR&X&NNN.NNN&	--	Configurar taxa de conversão do sensor X
RSR&X	RATE&X&NNN.NNNN	Ler taxa de conversão do sensor X
RBV	BAT&NNN	Leitura de porcentagem de carga da bateria

Cada nó da rede possui um endereço de 64 bits e um de 16 bits, análogos ao endereço físico MAC (*Media Access Control*) e endereço IP (protocolo *Ethernet/IP*). O computador que executa a camada de aplicação também é considerado um nó, sendo este o nó principal, chamado de coordenador, responsável pela configuração da rede e recepção de dados dos demais nós sensores. Estes demais podem ser configurados como *Routers* ou *End Devices*, sendo a diferença que o primeiro permite ser uma "ponte" até outros nós, realizando roteamento de mensagens até seu destino; o segundo somente recebe e envia mensagens,

não oferecendo nenhum papel de roteamento da rede. A disposição da rede pode ser observada na figura 7. A utilização de *Routers* permite estender o alcance da rede.

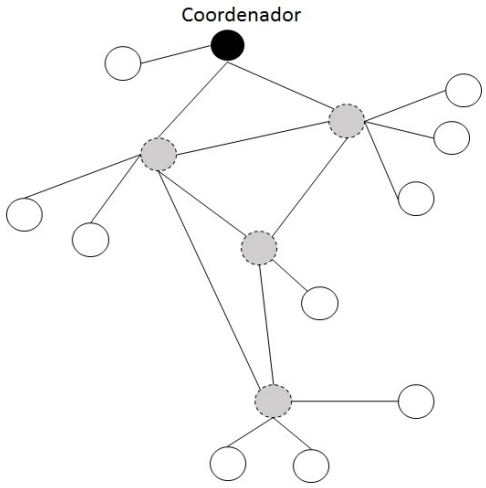

Figura 7. Disposição dos elementos na rede. *Routers* e *End Devices* em cinza e branco, respectivamente.

5. CONCLUSAO

Neste trabalho foi apresentada a modelagem e implementação de um *framework* multicamadas e genérico, voltado para aquisição de dados em uma rede de sensores sem fio *ZigBee*. Este *framework* contém os principais elementos sugeridos por Kiepert and Loo (2012), como portabilidade de *hardware*, sistema operacional embarcado, processamento paralelo, sincronia de mensagens, comunicação tolerante a falhas e suporte a armazenamento de dados, preenchendo assim uma lacuna da literatura, abordando aspectos práticos do desenvolvimento de tal sistema.

Com esse *framework* é possível agilizar o processo de desenvolvimento de uma aplicação *hardware/software* para aquisição, distribuição, visualização e armazenamento de dados, uma vez que essas funcionalidade são previamente implementadas neste trabalho. A implementação do *framework* passa por três camadas: a de aquisição, que é a camada de *hardware* responsável pela leitura de sensores; a de aplicação, que recebe dados provenientes da camada de aquisição e pode exibi-los e armazená-los; e a camada de comunicação, que garante a troca de mensagens entre as camadas de aquisição e aplicação através do protocolo *ZigBee (802.15.4)*.

O desenvolvimento do *framework* utilizou o PIC16F877A como principal plataforma de *hardware*, gerando uma interface de monitoramento de até oito sensores analógicos por nó da rede, e uma interface de comunicação via protocolo serial assíncrono com o dispositivo XBee. Alguns nós foram implementados com o microcontrolador PIC18F4550, requerendo poucas alterações no código, uma vez que o *firmware* desenvolvido prima pela portabilidade de *hardware*. Já a camada de aplicação foi desenvolvida em Java, provendo uma interface de armazenamento dos dados adquiridos em um banco de dados, além de prover mecanismos para a visualização gráfica desses dados, seja em tempo real ou uma exibição de séries históricas.

Para trabalhos futuros serão ainda implementadas novas funcionalidades ao framework, como por exemplo, cada leitura de sensor será acompanhada da informação de data e hora em que essa medida for realizada. Para tanto pretende-se utilizar o componente $DS1307$, que funciona como um relógio de tempo real e se comunica com o microcontrolador através do protocolo de comunicação serial I^2C. Também será incluída a leitura de força do sinal de cada nó. Pode ser feito o estudo de economia de energia (já que se tratam de nós remotos) através de algoritmos gerenciadores de modo *Sleep* do microcontrolador e do XBee; ou acoplamento de uma célula solar e realizar o chaveamento entre a bateria e esta célula.

REFERÊNCIAS

Benghanem, M. (2010). A low cost wireless data acquisition system for weather station monitoring. *Renewable Energy*, 35(4), 862 – 872.

Buratti, C. (2011). *Sensor Networks with IEEE 802.15.4 Systems: Distributed Processing, MAC, and Connectivity*. Springer Berlin Heidelberg, Heidelberg.

Erich Gamma, Richard Helm, R.J.J.V. (2000). *Padrões de Projeto*. Bookman, São Paulo.

Govoni, D. (1999). *Java application frameworks*. Wiley.

Guibas, L.J. and Zhao, F. (2004). *Wireless Sensor Networks: An Information Processing Approach*. Elsevier.

Kiepert, J. and Loo, S.M. (2012). A unified wireless sensor network framework. In *Systems Conference (SysCon), 2012 IEEE International*, 1–6.

Mahjoubi, A., Fethi Mechlouch, R., and Ben Brahim, A. (2011). A low cost wireless data acquisition system for a remote photovoltaic (pv) water pumping system. *Energies*, 1(4), 68–89.

Mershad, K. and Artail, H. (2013). A framework for secure and efficient data acquisition in vehicular ad hoc networks. *IEEE Transactions on VehicularTechnology*, 62(2), 536–551.

Nagasaka, Y. and Motoyama, H. (2007). A shared dataspace communication framework for data acquisition system. In *Real-Time Conference, 2007 15th IEEE-NPSS*, 1–4.

Nguyen, N. and Khan, M. (2013). Context aware data acquisition framework for dynamic data driven applications systems (dddas). In *Military Communications Conference, MILCOM 2013 - 2013 IEEE*, 334–341.

Vellingiri, S., Ray, A., and Kande, M. (2013). Wireless infrastructure for oil and gas inventory management. *IECON Proceedings (Industrial Electronics Conference)*, 5461–5466.

Wang, H., Quan, W., Wang, Y., and Miller, G.R. (2014). Dual roadside seismic sensor for moving road vehicle detection and characterization. *Sensors*, 14, 2892–2910.

Zhao, L., He, L., Jin, X., and Yu, W. (2013). Design of wireless sensor network middleware for agricultural applications. In *Computer and Computing Technologies in Agriculture VI*, volume 393 of *IFIP Advances in Information and Communication Technology*, 270–279. Springer Berlin Heidelberg.