DESENVOLVIMENTO E IMPLEMENTAÇÃO DE SISTEMA DE MONITORAMENTO REMOTO UTILIZANDO REDES DE SENSORES SEM FIO (RSSF)

JESSICA F. LOPES, PAULO ROGÉRIO SCALASSARA, CARLA MARIA M. DOS SANTOS,

ARTHUR HIRATA BERTACHI

Centro Integrado de Pesquisa em Controle e Automação (CIPECA), Universidade Tecnológica Federal do Paraná, câmpus Cornélio Procópio
Avenida Alberto Carazzai,1640 – Cornélio/PR, Brasil – CEP 86300-000
E-mails: je.fernaandes@gmail.com,prscalassara@utfpr.edu.br, carla_martins91@hotmail.com, arthurltda@hotmail.com

Abstract— A Wireless Sensor Network consists in monitoring environmental conditions, which has many applications. This paper presents the initial part of the project that involves a RSSF projected to monitor the temperature about a determined place whose test program with nesC language in the TinyOS software is modified. Because the conversion of analog data to digital one, it needs to modify the Java application to be coded the temperature on Celsius scale, facilitating the observation. Therefore, the data was collected in a given environment at the desired scale.

Keywords— Wireless Sensor Network, TinyOS, nesC language, Embedded Systems

Resumo— Redes de Sensores sem fio têm como objetivo monitorar algum fenômeno decorrente no ambiente, possuindo, assim, diversas aplicações. Este artigo apresenta a parte inicial do projeto que envolve uma RSSF projetada para monitoramento de temperatura de um determinado espaço através da modificação do programa de teste apresentado em linguagem nesC próprio do software TinyOS. Devido a sua conversão de dados coletados analógicos para digitais, necessita-se de modificação na implementação do aplicativo Java para que seja codificada a temperatura em escala Celsius, facilitando a sua observação. Diante disso, coletou-se em um determinado ambiente dados de temperatura na escala desejada.

Palavras-chave— Redes de Sensores sem Fio, TinyOS, linguagem nesC, Sistemas Embarcados

1 Introdução

As redes de sensores sem fio (RSSF) são formadas por um grande número de pequenos sensores que têm como objetivo detectar e transmitir alguma característica do meio. Esta tecnologia, cada vez mais promissora, está tornando-se uma ferramenta indispensável para diversas atividades. Assim, segundo (VIEIRA et al., 2004), vários fabricantes de sensores passam a utilizar deste meio em suas linhas de produção. Suas peculiaridades mais importantes vão desde a tecnologia de transmissão de dados, arquitetura, topologia, segurança e proteção de dados. Com isso, dentre estes diversos aspectos avaliados, utilizou-se o kit de sensores da CrossBow, que constitui-se de: três processadores, dois sensores e uma interface USB-PC.

Estes dispositivos apresentam importantes recursos nos ambientes corporativos. Seu diferencial está presente no baixo custo de infraestrutura e no suporte a aplicações móveis, além de serem utilizadas por profissionais das mais diversas áreas. Cada vez mais, a tecnologia está buscando trazer conforto ao usuário e diminuir a necessidade de infraestrutura física para a implantação de sistemas. As redes de sensores sem fio vêm colaborando com estes aspectos no que diz respeito à mobilidade ao usuário, além de reduzir ainda mais as necessidades de cabos para sua comunicação. Diante disso, buscou-se estudar a aplicação da ferramenta Oscilloscope para a coleta de dados em um determinado ambiente (MONTEZ, 2011).

2 TinyOS/NesC

2.1 Programa Tinyos

TinyOS é um sistema operacional *open-source*, escrito na linguagem de programação nesC, com baixa complexidade, ou seja, projetado para dispositivos de baixa potência sem fio. Esse sistema possui ferramentas complementares como: aplicativos em Java, *Python* e *Shell Scripts*; bibliotecas escritas em 'C'. Baseado em componentes de *software*, ele promove a abstração de *hardware* utilizando um mecanismo que perime processamentos adicionais antes do término da transmissão (LEMOS, 2010).

Em relação ao modelo e linguagem de programação, o TinyOS inclui conceitos de eventos, comandos e tarefas. Uma configuração completa do sistema consiste em um pequeno programa composto de uma aplicação e de seus componentes do TinyOS, ou seja, um conjunto de componentes agrupados (LEMOS, 2010).

2.1.1 Estrutura da Aplicação

Como fora mencionado anteriormente, uma rede de sensores sem fios tem por aplicação coletar

DOI: 10.5540/03.2013.001.01.0154 © 2013 SBMAC

informações e enviá-las para um ou mais *motes data skins*. Estes fornecem dados ao computados através de uma porta serial, o qual os apresenta através de uma interface.

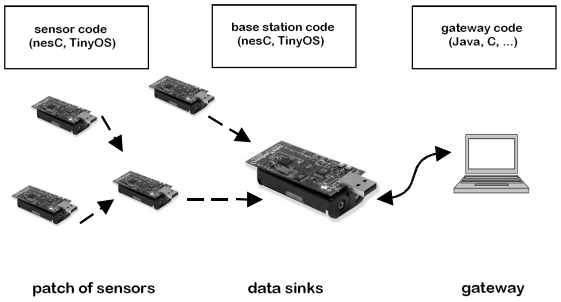

Figura 1. Modelo esquemático da aplicação (BERNARDO, 2011)

Conforme Figura 1, nota-se que uma aplicação em RSSF envolve três partições distintas, quais sejam:

- Nós sensores (*mote sensor*): Utilizou-se o sensor IRIS, conforme Figura 2, que possui 8 k*bytes* de RAM, 128 k*bytes* de memória *flash* para código de programas e 512 k*bytes* de memória *flash* para dados; composto de um microcontrolador ATMEGA1281 e um módulo de rádio-frequência AT86RF230 da Atmel. Para obtenção dos dados de temperatura, fez-se o uso da placa sensorial MDA100CB, de acordo com a Figura 3, caracterizada por uma placa de aquisição de dados que tem um termistor de precisão, sensor de luz/fotocélula e área para prototipagem.

Figura 2. Sensor IRIS (BERNARDO, 2001)

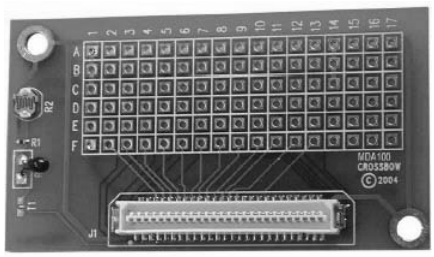

Figura 3. Placa sensorial MDA100CB

- *Mote* raiz: Formado pela junção do sensor IRIS com o adaptador MIB520CA, como se pode visualizar na Figura 4, que permite o carregamento de aplicações e a comunicação através da porta serial.

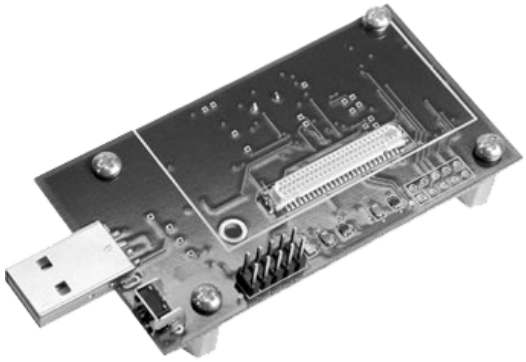

Figura 4. Adaptador MIB520CA (BERNARDO, 2011)

- *Gateway*: Composto pelo computador que permite a visualização dos dados coletados.

2.2 Linguagem NesC

NesC é uma linguagem de programação orientada a componentes com um modelo de execução baseado em eventos. Uma característica importante é o fato das aplicações serem capazes de responder a eventos do ambiente. Esta consiste em um dialeto de 'C', e foi projetada para incluir os conceitos estruturais e modelos de execução do TinyOS (AMÉRICO, 2010).

Os aplicativos escritos em nesC são compostos por componentes, que podem ser construídos e combinados para formar uma aplicação, aumentando, assim, a modularidade e usabilidade do código. Nessa linguagem, o comportamento desses componentes é especificado em termos e conjuntos de interfaces bidirecionais, que informam o que se usa e fornece, conforme Figura 5, a qual indica que o componente Timer fornece as interfaces *StdControl* e *Timer,* usando interface *Clock* (BREWER, 2004).

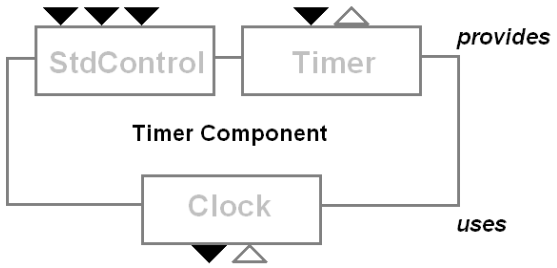

Figura 5. Interfaces de um componente (GROUP, 2006)

Componentes são ligados uns aos outros via interfaces. O fluxo de informação pode ocorrer por meio de comandos ou de eventos. Sendo assim, seus pontos principais são:

- Tarefas: realizam conjuntos de comandos;

 © 2013 SBMAC

- Eventos: sinalizam interrupções externas, geram um sinal, recebe/aceita um eventos;
- Comandos: funções de procedimentos para outros componentes, não sinalizam eventos

2.4 Programa Oscilloscope

Oscilloscope é um aplicativo que permite a visualização das leituras que os sensores fazem através do computados (PIERI, 2008). Cada *mote* possui um instalado e, periodicamente, amostras de suas respectivas coletas são transmitidas a um *BaseStation* e então, apresentadas, graficamente, na tela por meio de um aplicativo Java. Na Figura 6, pode-se observar o exemplo de um dos gráficos gerados através do monitoramento de um laboratório.

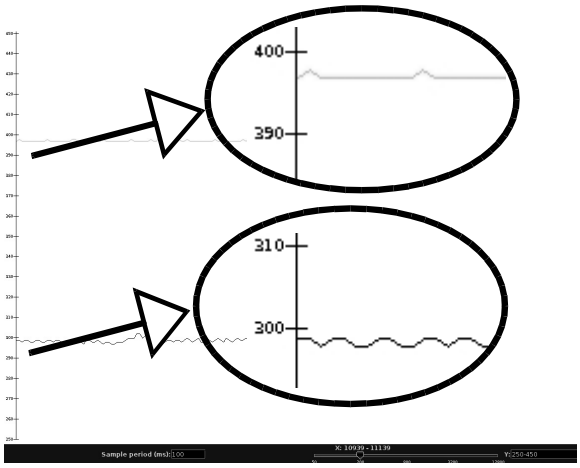

Figura 6. Monitoramento de dados com dois sensores em escala ADC

Na Figura 6, nota-se que com a utilização de dois sensores para coleta de dados, houve uma grande diferença entre seus respectivos dados, com o sensor 1 obtendo uma oscilação maior que o sensor 2, estes se encontravam a uma distância de, aproximadamente, 10 metros. Esta diferença ocorreu devido aos ambientes que estes se encontravam, pois o sensor 1 encontrava-se em um local aberto com ocorrência de rajadas de vento, enquanto o sensor 2 permanecia-se em um local fechado. Este monitoramento foi executado num intervalo de 100 [ms] para recebimento de cada pacote, tendo assim, recepção de dados praticamente contínua. A escala apresentada na Figura 6, está na escala ADC (*Analog to Digital Converters).*

2.5 Comunicação entre motes

A comunicação entre *motes* é realizada através da troca de mensagens. Nestas mensagens são contidas em variáveis do tipo message_t, que contêm um inteiro de 8 *bits* que define o tipo de pacote e um campo com os dados. Cada pacote pode guardar uma certa quantidade dada em *bytes*, a qual é definida em TOSH_DATA_LENGTH. Este é, por determinação, igual a 28 *bytes*, porém pode ser modificado durante a compilação do código para até 9 vezes o valor inicial. O conteúdo de uma mensagem é definido através da definição de uma estrutura (a qual é feita em um arquivo com terminação .h) que utiliza dados de redes portáteis identificados por iniciar com *nx_struct*. Na Figura 7, pode-se observar como se utiliza essa estrutura usando, por exemplo, *nx_uint16_t* (BERNARDO, 2011).

```
#ifndef ANTITHEFT_H
#define ANTITHEFT_H

typedef nx_struct theft {
        nx_uint16_t who;
} theft_t;
...

#endif
```

Figura 7. Estrutura *nx_uint16_t*

Mensagens ativas (AM) designam-se por permitirem associar emissores e receptores distintos a cada tipo de mensagem. O envio de mensagens é suportado pela interface *AMSend*, o qual o comando *send* e o evento *sendDone* realizam o envio de pacotes. Para isso ocorrer, existem dois endereços que são definidos ao iniciar o sistema: endereço de *broadcast*, definido por TOS_BCAST_ADDR, e o endereço do próprio nó, TOS_NODE_ID, definido no instante da compilação e instalação do código do *mote*. Com isso, a interface é realizada através do componente de configuração *AMSenderC*, que recebe o número do tipo de mensagem trocada (BERNARDO, 2011).

3 *Oscilloscope*

3.1 Modificação do programa

As placas de sensores MTS101CA, as quais são utilizadas no *kit* da Crossbow de sensoriamento, possui um termistor de precisão, captando precisão de até 0,2°C de variação. Este possui resistência, a 25°C de 10 kΩ, que varia conforme a temperatura do ambiente, e está conectado ao canal do conversor analógico-digital, conforme dados do fabricante disponíveis no MTS/MDA *Sensor Board Users Manual*. Para utilizar o termistor, este sensor dever ser habilitado através da criação de uma linha de controle digital, conforme Figura 8.

 © 2013 SBMAC

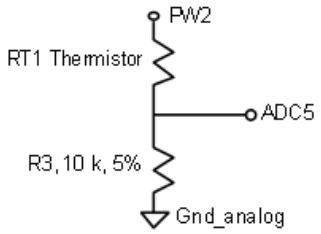

Figura 8. Circuito esquemático do termistor

Os sinais coletados através da transmissão de dados são medidos numa escala de 0 a 1023. Convertendo-os para a escala graus Celsius, este pode alcançar, aproximadamente, de 0°C a 50°C. Para que os dados apresentados no *Oscilloscope* fossem dados à temperatura de conversão, modificou-se o programa com base nas Equações 1 e 2. Nestas equações, os parâmetros são informados pelo fabricante.

$$C = a + b.\ln(R_{thr}) + c.[\ln(R_{thr})]^3 \quad (1)$$

$$R_{thr} = \frac{R1(ADC_{FS} - ADC)}{ADC} \quad (2)$$

a – 0.001010024
b – 0.000242127
c – 0.000000146
R1 – 10 kΩ
ADC_FS – 1023
ADC – valor que o sensor está lendo.

Como a escala do *Oscilloscope* não é dada em graus Celsius, modificou-se o programa conforme a Equação (1) e (2), alterando-o para facilitar a coleta de dados sem precisar realizar conversão de temperatura, conforme Figura 9.

```
uint16_t adc2celcius(uint16_t adc) {

      float Rth = R1(ADC_FS-ADC)/ADC;
      float temp_k = a + b*ln[Rth] + (ln[Rth])^3

      float temp_c = (1/temp_k) - 273;

      return (uint16_t)temp_c;

}
```

Figura 9. Alteração no programa *Oscilloscope*

Após a implementação desta parte no programa, este passou a apresentar os dados diretamente na escala Celsius, conforme a Figura 10 está representada. Assim, com compilação e execução juntamente com o programa alterado, fez-se a mesma mudança para o programa *MultihopOscilloscope* cuja diferença está na disseminação e coleta de dados.

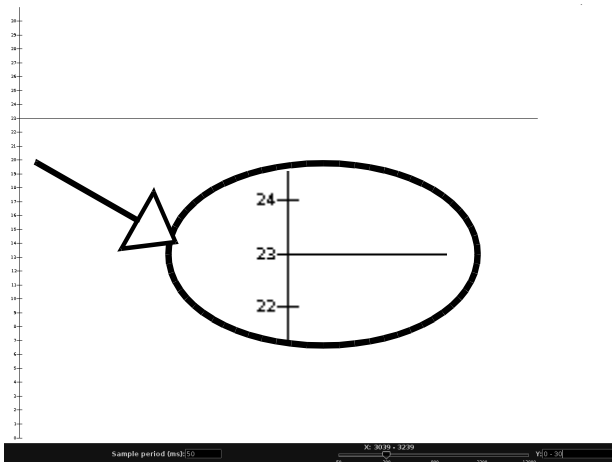

Figura 10. Monitoramento com escala Celsius

4 Programa *MultihopOscilloscope*

4.1 Collection de Mensagens

Para suportar a coleta de mensagens, o sistema operacional TinyOS oferece o serviço *Collection*. Os *motes* sensores usam a interface *Send* para enviar mensagens para o *mote* raiz. Essa interface é extremamente parecida com a interface *AMSend*, exceto que não indica o endereço de destino. Desta forma, pode-se ter várias coletas de dados numa mesma rede, desde que identificadas por número de tipos diferentes (BERNARDO, 2011).

Collection resume-se na operação complementar para a divulgação e o recolhimento de dados gerados na rede em uma estação de base. A abordagem geral utilizada é a construção de uma ou mais "árvores" de recolha, cada uma das quais são enraizadas em uma estação de base. Quando um nó possui dados que devem ser recolhidos, ele os envia até à raiz da "árvore", ou seja, a estação base, encaminhando os dados coletados através de outros nós que formam a estrutura da árvore (GROUP, 2006).

4.2 Dissemination de Mensagens

Para suportar a disseminação de um valor através de uma rede, a partir do mote raiz, o sistema operacional TinyOS possui o protocolo *Dissemination*. Ele permite que os administradores possam reconfigurar e reprogramar uma rede de coleta de dados, o que é importante, pois torna a operação robusta para desconexões temporárias ou perda por pacotes elevados (GROUP, 2006). Esse processo pode ser realizado através de duas interfaces:

- *DisseminationValue*: interface do tipo parametrizada com o valor que se pretende disseminar, com isso, qualquer mote pode modificar seu valor, embora seja mais eficiente realizar a modificação a partir do mote raiz.

 © 2013 SBMAC

- *DisseminationUpdate*: para se modificar o valor, o mote raiz utiliza-se desta interface, associando-se à raiz de disseminação.

4.3 Programa MultihopOscilloscope

Este programa é baseado no programa *Oscilloscope*, porém possui o protocolo de coleta de dados. Periodicamente, os sensores coletam amostras, inclusive, neste caso, o *mote* raiz. Os dados coletados também são demonstrados através do aplicativo Java *Oscil-loscope*, que podem ter amostragens a partir de 1 Hz, mas podendo ser alterado.

Assim, a modificação feita neste programa para demonstração de dados em escala Celsius foi a mesma que no programa anterior, conforme Figura 11.

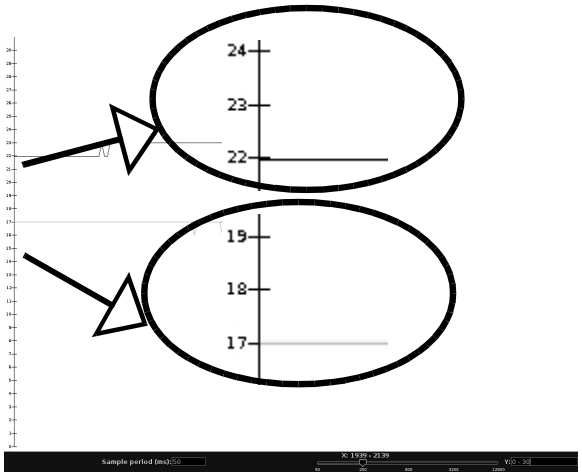

Figura 11. Programa *MultihopOscilloscope* em escala Celsius

5 Conclusão

Redes de Sensores sem Fio são formadas por diversos dispositivos cujo objetivo é a caracterização do meio através de coletas de dados. Com uma tecnologia cada vez mais promissora, estes são ferramentas indispensáveis, além de seu baixo custo. A proposta deste trabalho tem como foco o uso e o melhoramento destes sensores através dos programas criados aos seus próprios *softwares* de utilização, mas com melhor desenvolvimento e aproveitamento em suas aplicações. Este aprimora o desenvolvimento dos nós da rede de sensores, através do uso de protocolos para que a coleta de dados seja mais eficiente, podendo, assim, atingir um alcance maior no raio de monitoramento e comparação na coleta destes.

Agradecimentos

Os autores agradecem a Universidade Tecnológica Federal do Paraná (UTFPR), câmpus Cornélio Procópio, também ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Fundação Araucária (processo número 338/2012) pelo apoio financeiro e bolsas.

Referências Bibliográficas

Américo, L. (2011). Programação para Redes de Sensores sem Fio usando TinyOS.

Bernardo, L. (2011). Redes AD HOC e de Sensores. Dissertação de Mestrado, Faculdade de Ciências e Tecnologia (FCT), Portugal.

Brewer, E. C.; Culler, D.; Gay, D.; Levis, P.; Behren, R. and Welsh, M. (2004). NesC: A Programming Language for Deeply Networked Systems.

Group, TinyOS. (2006). TinyOS Tutorials. Disponível em: <http://docs.tinyos.net/tinywiki/index.oho/TinyOS_Tutorials>. Acesso em: 21 maio 2013.

Lemos, M. V. and Leal, L. B. (2010). Relatório sobre o Sistema Operacional TinyOS e Linguagem NesC, Vol. 1, No. 1, pp. 1-12.

Montez, C. (2011). Redes Locais sem Fio: Conceitos e Aplicações.

Pieri, L. R. B. (2008). Monitoração de Pressão Sonora Utilizando Redes de Sensores Sem Fio – SPLDM. Trabalho apresentado na Universidade Federal de Santa Catarina (UFSC), Florianópolis.

Vieira, L. F. M.; Linnyer, B. R.; Correia, L. H.; Macedo, D. F.; Nakamura, E. F.; Figueiredo, C. M. S.; Vieira, M. A. M. and Silva, D. C. J. (2004). Arquitetura para Redes de Sensores sem Fio. Trabalho apresentado na Universidade Federal de Minas Gerais (UFMG), Belo Horizonte.

 © 2013 SBMAC