UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARLOS ANTONIO DOS SANTOS LIMA

SISTEMA PARA ANÁLISE DE DADOS DE NODOS SENSORES

MOSSORÓ – RN

2014

MARLOS ANTONIO DOS SANTOS LIMA

SISTEMA PARA ANÁLISE DE DADOS DE NODOS SENSORES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação – associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Pedro Fernandes Ribeiro Neto.

MOSSORÓ – RN

2014

Catalogação da Publicação na Fonte.
Universidade do Estado do Rio Grande do Norte.

Lima, Marlos Antonio dos Santos.

Sistema para análise de dados de novos sensores. / Marlos Antonio dos Santos Lima. – Mossoró, RN, 2014

52 f.
Orientador(a): Prof. Dr. Pedro Fernandes Ribeiro Neto.

Dissertação (Mestrado em Ciência da Computação). Universidade do Estado do Rio Grande do Norte. Universidade Federal Rural do Semi-Árido. Programa de Pós-Graduação em Ciência da Computação.

Bibliotecária: Elaine Paiva de Assunção CRB 15 / 492

MARLOS ANTONIO DOS SANTOS LIMA

SISTEMA PARA ANÁLISE DE DADOS DE NODOS SENSORES

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: ___ / ___ / _____.

BANCA EXAMINADORA

_____

Prof. Dr. Pedro Fernandes Ribeiro Neto
Orientador
Universidade do Estado do Rio Grande do Norte – UERN

_____

Prof. Dr. Henrique Jorge Amorim Holanda – UERN
Coorientador
Universidade do Estado do Rio Grande do Norte – UERN

_____

Prof. Dr. Silvio Roberto Fernandes de Araújo
Membro Interno
Universidade Federal Rural do Semi-Árido – UFERSA

_____

Prof. Dr. Antônio Alfredo Ferreira Loureiro
Membro Externo
Universidade Federal de Minas Gerais – UFMG

Dedico este trabalho aos meus pais
Maria do Socorro e Antônio Neto, com
todo amor e gratidão, pelo esforço que
fizeram para a realização desta
grande conquista.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me protegido, dado forças e sabedoria para continuar a caminhada e acreditar que podemos ser pessoas cada vez melhores.

Em especial ao meu Pai e minha Mãe, pelo grande amor, por seus grandiosos ensinamentos, educação, e diante de todas as dificuldades por terem me incentivado aos estudos, desde o início de minha vida. “Devo tudo a vocês, Mãe e Pai, obrigado! Amo muito vocês”.

Aos meus irmãos, Mauricio e Marcia, pelo companheirismo e amizade que sempre tivemos. “Me orgulho de vocês por serem essas pessoas tão boas”.

À minha família em geral, tios, tias, primos, primas, avôs e avós pela grande ajuda que sempre me deram.

À uma pessoa que conheci e que me fez ver o mundo com outros olhos. Lailsa, agradeço imensamente por você ter surgido em minha vida. Sua companhia me trouxe mais felicidade, harmonia, alegria, compreensão e amor. Espero viver ao seu lado por todos os anos de nossas vidas e se possível todo o sempre. És minha fonte de inspiração, caixinha de surpresas felizes e soluções para tudo.

À Leila e Dona Lourdes, irmã e mãe do meu amor, pois são duas pessoas que considero parte da minha família.

Ao André (da Marcia, minha irmã), grande amigo de computação, pela atenção que me deu nos momentos de necessidade.

Não poderia me esquecer de agradecer aos amigos que me acompanharam nessa jornada: Marlon, Anderson, Jomar, Ronnison, Davi, Denilson, Marianny, Ismael, Rodrigo, Kayo, Aristóteles e os demais, não menos importantes, que me acompanharam ao longo da jornada.

Aos meus amigos de república, Josiel e Elvis, e amiga Clarice, todos com quem partilhei momentos de dificuldade.

Ao meu orientador, Pedro Fernandes, pessoa que considero um grande amigo, pela sua contribuição em todos os sentidos, pelas oportunidades cedidas, e por todo o conhecimento de mundo repassado a mim.

Aos amigos que tive a oportunidade de conhecer no GES (Grupo de Engenharia de Software - UERN) que tornam este grupo uma família.

E aos professores do PPGCC-UERN/UFERSA pela determinação em repassar seus ensinamentos tanto dentro como fora de sala de aula.

"Obstáculos são aqueles
perigos que você vê quando tira
os olhos de seu objetivo."
(Henry Ford)

RESUMO

As Redes de Sensores sem Fio (RSSF) têm sido alvo de pesquisas em que cada vez mais tecnologias são agregadas para proporcionar uma melhor usabilidade de sua programação e implantação. Neste tipo de rede, o sensoriamento é uma de suas tarefas primordiais, caracterizado pela aquisição de sinais de fenômenos físicos do ambiente e conversão em dados que possam ser interpretados e processados. No entanto, trata-se de um processo sujeito a falhas e para incrementar a confiabilidade desse, o diagnóstico e detecção precoce de desvios em dados reportados pelos nodos é pertinente, possibilitando a utilização de técnicas de correção ou tolerância. Uma forma de prover tal diagnóstico é analisando os dados reportados pelos nodos antes de sua inserção no meio, pois permite o conhecimento à respeito do comportamento de variações numéricas no processo de aquisição de sinais físicos do ambiente. Assim sendo, um sistema que possibilita a análise de dados de uma RSSF foi desenvolvido. Este, é capaz de controlar o nível de energia no nodo sensor, simulando o decaimento de tensão elétrica da bateria, já que tal fenômeno é evidente em aplicações reais dessa tecnologia. Dessa forma, o sistema permite a verificação da existência de desvios em leituras realizadas pelos sensores, antes da implantação da rede em ambiente real. Como resultados, foi possível identificar a ocorrência de desvios em dados de nodos sensores reais avaliados no estudo de caso e aplicar um método de correção para reduzir a inconsistência observada na leitura de variáveis físicas.

Palavras-Chave: Rede de Sensores sem Fio; nível de energia; desvios em dados.

ABSTRACT

Wireless Sensor Networks have been subject of research in which more and more technologies are aggregated to provide better usability within its programming and deployment. In this type of network, the sensing is one of your primary tasks, characterized by the acquisition of signs of physical phenomena in the atmosphere and conversion data that can be interpreted and processed. However, it is a process prone to failure and to increase the reliability of this, the diagnosis and detection early of deviations in data reported by the nodes is relevant, allowing the use of techniques for correction or tolerance. One way to provide such diagnosis is analyzing the data reported by the nodes before their insertion in the environment because allow knowledge about the behavior of numerical variations in the acquisition of physical signs of the environment process. Therefore, a system that enables data analysis of a WSN has been developed. This is able to control the energy level of the sensor node, simulating the decay of the voltage of the battery, as this phenomenon is evident in real-world applications of this technology. Thus, the system allows the determination of deviations in readings taken by the sensors before network deployment in real environment. As a result, it was possible to identify the occurrence of deviations in data evaluated in the case study of real sensor nodes and apply a correction method to reduce the inconsistency observed in readings of physical variables.

Keywords: Wireless Sensor Networks; energy level; deviations in data.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

DARPA	*Defense Advanced Research Projects Agency*
DC	*Direct Current*
FPGA	*Field Programmable Gate Array*
IEEE	*Institute of Electric and Electronic Engineers*
MCU	*Microcontroller Unit*
PPM	Partes por Milhão
PWM	*Pulse Width Modulation*
RSSF	Rede de Sensores sem Fio
USB	*Universal Serial Bus*

SUMÁRIO

1. INTRODUÇÃO

1.1. Contextualização

As Redes de Sensores sem Fio (RSSF) têm atraído cada vez mais interesse e recursos advindos de pesquisas desde seu surgimento (MENDES e RODRIGUES, 2011). Isto devido aspectos particulares de sua comunicação, processamento de dados e gerenciamento de energia. Além disso, dificuldades ligadas a essas redes são potencializadas quando aplicada em ambiente real.

Recentes avanços em áreas como microeletrônica, eletrônica digital e comunicação sem fio possibilitaram às RSSF o desenvolvimento de unidades de processamento e sensoriamento de baixo custo, baixo consumo de energia e de pequeno tamanho denominados nodos [1] sensores (MAHAPATRO e KHILAR, 2013). Estes dispositivos são portados de funcionalidades para aquisição de sinais físicos do ambiente e comunicação por meio de rede *ad hoc*. São exemplo de variáveis físicas coletadas por sensores: humidade, temperatura, luminosidade, pressão atmosférica, aceleração.

Necessidades dos nodos sensores como, por exemplo, mobilidade, assim como, de muitas de suas aplicações destinadas à áreas remotas, que não permitem o fácil o acesso a esses elementos (LOUREIRO *et al.*, 2003), requisitam unidades de armazenamento de energia como baterias elétricas. No entanto este, componente faz com que os hardwares que compõem o nodo atuem diante de esgotamento de energia. Além disso, fenômenos climáticos obrigatoriamente influenciam o funcionamento do sensor, fazendo com que possam apresentar alterações indesejadas nos dados coletados, no processo de aquisição.

Um dos aspectos que merece atenção é que a qualidade dos dados repassados pelos nodos variam de acordo com a qualidade dos sensores utilizados. O custo financeiro e computacional está ligado diretamente à escolha da tecnologia a ser aplicada e à quantidade dispositivos necessários em uma RSSF. Desse modo, a fim de reduzir o custo, hardwares com menor robustez são embutidos em nodos sensores, o que torna necessário conhecer melhor o seu funcionamento.

[1] Neste texto, o termo "nodo" será adotado em todo o seu corpo como sinônimos de "nó" que vem de "nó sensor". No sentido formal o termo nodo sensor de uma Rede de Sensores sem Fio se aplica à um sistema de computação com capacidade de processamento, memória e comunicação sem fio que possui um ou mais sensores.

1.2. Problemática

A detecção de eventos e comportamentos anômalos em Redes de Sensores sem Fio é um desafio importante para tarefas tais como aplicações de monitoramento, diagnóstico de falhas e detecção de intrusão (RAJASEGARAR *et al.*, 2007).

O processo de detecção de falha em dados se dá pela identificação de desvios de padrão em valores repassados pelos nodos. Agentes climáticos do ambiente e insuficiência energética de baterias utilizadas nesses dispositivos são causadores de falhas. Desse modo, uma vez que a rede se encontra em ambiente não controlado, a possibilidade de ocorrer o diagnóstico errado na identificação e definição de possíveis falhas de hardware é evidente, e ainda mais no caso de ausência de valores que poderiam ser coletados antes da implantação da rede, para comparação. Todavia, não é comum que o hardware possua avaria antes de sua utilização.

Na literatura é possível encontrar pesquisas que analisam dados de RSSF. Em Luo *et al.* (2004) é realizada a coleta de dados de uma rede composta por nodos MICA2. Nesta, variáveis físicas como temperatura e humidade podem ser comparados entre si em relação a integridade.

Já Anastasi *et al*. (2004), Ni *et al*. (2009) e Sharma *et al.* (2010) analisam dados advindos de aplicações de RSSF a fim de identificar e categorizar causas e consequência de falhas no processo de aquisição de dados pelos nodos sensores.

Porém, um aspecto em comum observado é que tais pesquisas utilizam dados de aplicações de redes em ambientes dinâmicos, sujeitos a fenômenos climáticos imprevisíveis. Portanto, a probabilidade de falhas passarem despercebidas é evidente se variações em leituras forem atribuídas à fenômenos climáticos, ou o contrário, se fenômenos climáticos forem interpretados como falhas. Outro aspecto também observado é que analisam dados *versus* tempo e dados *versus* quantidade de amostras, havendo pouco ou nenhuma relação mais detalhada sobre dados *versus* nível de energia. Dessa forma, uma avaliação antecipada dos sensores que compõem o nodo, em ambiente controlado (onde são conhecidas suas variáveis físicas), com base no nível de energia, se faz de grande importância para verificar a corretude do processo de aquisição de sinais físicos e garantir um diagnóstico mais preciso dos fatores que afetam dados monitorados pela RSSF, possibilitando a ação antecipada de medidas que visem incrementar a confiabilidade o processo de leitura dos sensores.

1.3. Objetivos

1.3.1. Geral

O objetivo desta pesquisa proporcionar a avaliação de corretude do processo de sensoriamento de nodos sensores, com base no decaimento do nível de energia, uma vez que o consumo de bateria é evidente em aplicações de RSSF. Um fator importante é a realização da análise em ambiente controlado, onde são conhecidos os valores de suas variáveis físicas.

1.3.2. Específico

Como objetivos específicos deste trabalho podemos destacar:

- Prover o controle ajustável de nível de energia no nodo sensor;
- Desenvolver um sistema em software e hardware para desempenhar a análise proposta.

1.4. Organização da Dissertação

Esse trabalho está organizado e distribuído da seguinte forma.

O segundo capítulo trata de uma visão geral sobre RSSF, onde são descritos histórico, conceitos, características, arquitetura, e categorização de falhas de dados de nodos sensores que compõem essa tecnologia.

O terceiro capítulo destina-se ao desenvolvimento de um sistema de análise de dados de RSSF, juntamente com seus componentes principais e auxiliares, programação, linguagens utilizadas e fluxo de execução. Além disso, é apresentado o estudo de caso no qual foram analisados nodos sensores, fabricados pela empresa Memsic *Tecnology*[2], identificados desvios em suas leituras e aplicado um método de correção.

Por fim, no quarto capítulo são feitas as considerações finais sobre os resultados alcançados, e apresentados trabalhos futuros.

[2] http://www.memsic.com/

2. REDES DE SENSORES SEM FIO

Este capítulo apresenta um breve histórico e definição sobre Redes de Sensores sem Fio (RSSF). Sobre o nodo sensor, são caracterizados os subsistemas que o compõe, como também abordadas questões relacionadas à falhas a que o subsistema de sensoriamento está sujeito.

2.1. Breve Histórico

As Redes de Sensores sem Fio tiveram seu surgimento a partir de aplicações militares (DARGIE; POELLABAUER, 2010). Seu desenvolvimento foi impulsionado por pesquisas financiadas pela agência norte americana DARPA (*Defense Advanced Research Projects Agency*) que Organizou o *Distributed Sensor Nets Workshop* (Proceedings of the Distributed Sensor Nets Workshop, 1978) concentrando-se em desafios de pesquisas em redes de sensores, tais como tecnologias de rede, técnicas de processamento de sinal e algoritmos distribuídos.

Com a ocorrência de avanços na computação, comunicação e micro eletrônica na década de 1990, a DARPA pôde atuar como pioneira em uma nova vertente de pesquisas em Redes de Sensores sem Fio no programa SensIT (WANG e BALASINGHAM, 2010), provendo capacidades tais como redes *ad hoc*, consulta dinâmica, *tasking*, reprogramação e *multi-tasking*.

Ao mesmo tempo, o IEEE (*Institute of Electric and Electronic Engineers*) notou o baixo custo e alta capacidade que redes de sensores ofereciam. Com isso, a organização definiu o padrão IEEE 802.15.4[3] para baixa transferência de dados em redes sem fio de área pessoal. Baseado no IEEE 802.15.4, a ZigBee Alliance[4] publicou o padrão ZigBee, que especifica um conjunto de protocolos de comunicação de alto nível que pode ser usado por RSSF.

Um exemplo é a China que tem usado RSSF em seus programas de investigação estratégicos nacionais (NI *et al.*, 2007) e atualmente a comercialização dessa tecnologia também têm sido acelerada por companhias como Linear *Tecnology*[5] e Memsic *Tecnology*[6].

[3] http://www.ieee802.org/15/pub/TG4.html/
[4] http://www.zigbee.org/
[5] http://www.linear.com/

2.2. Definição

Por definição, uma RSSF é representada por um conjunto de dispositivos autônomos chamados nodos sensores cuja finalidade é monitorar e ou atuar sobre o ambiente (YICK, MUKHERJEE, e GHOSAL, 2008; DARGIE; POELLABAUER, 2010).

Em aplicações reais os nodos são implantados em áreas geográficas para monitorar fenômenos físicos tais como temperatura, umidade, vibrações, abalos sísmicos, dentre outros. Possuem aplicações em diversas áreas, em exemplo: na área médica (para monitoramento remoto de pacientes) (JANANI *et al.*, 2011), na área militar (no monitoramento de forças inimigas), residencial, industrial, ambiental, em infraestruturas (LOUREIRO *et al.*, 2003), bem como em aplicações civis incluindo monitoramento de habitats (REZAEI; MOBININEJAD, 2012).

Uma forte característica dessas redes é que em certas aplicações podem ser inviável a intervenção humana, em especial quando o cenário se caracteriza com hostil ou impraticável financeiramente.

Uma RSSF pode ser composta por dois tipos de nodos: sensores e sorvedouros (*sink*). A Figura 1 apresenta, segundo Mendes e Rodrigues (2011), o cenário que caracteriza essas redes.

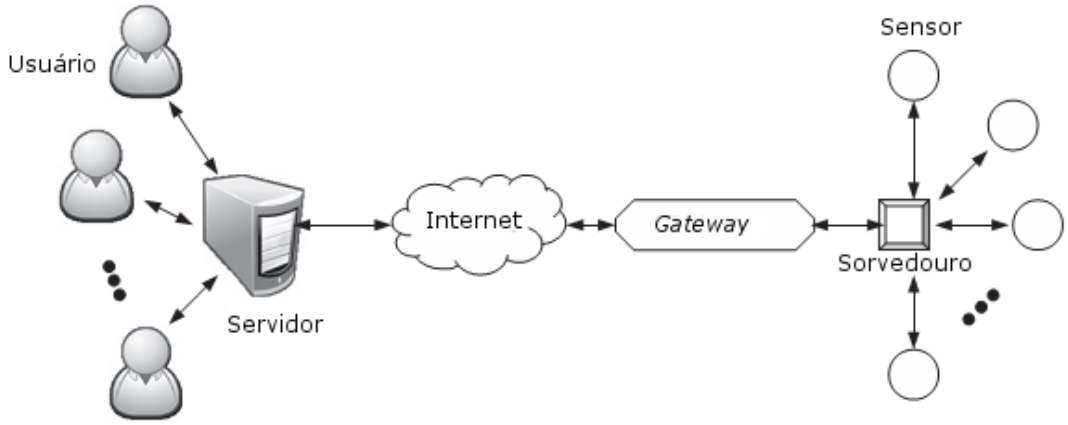


Figura 1: Exemplo de RSSF. Adaptado de Mendes e Rodrigues (2011).

Os nodos sensores, espalhados em uma região de interesse, coletam dados e os repassam para o sorvedouro através de comunicação sem fio direta ou por múltiplos saltos

[6] http://www.memsic.com/

(PAK e BAHK, 2012). Por sua vez, o sorvedouro se conecta a um *gateway* para envio dos dados à uma rede externa.

Em se tratando de aplicações onde é impossível ou impraticável financeiramente intervir manualmente no nodos, a rede deve ter um tempo de vida suficientemente longo para satisfazer os requisitos de aplicação. Esse tempo pode se configurar na ordem de dias, meses ou até anos. Além disso, a corretude dos dados repassados pelos nodos dever ser garantida.

2.3. Características

Um dos grandes desafios nas RSSF é encontrar mecanismos que sejam suficientes à determinadas aplicações para suportarem qualidade de serviço, tempo de vida e facilidade de manutenção (KARL e WILLIG, 2005). Desse modo, mecanismos típicos de redes de sensores são:

- **Comunicação em múltiplos saltos (*multi-hop*):** utilização de nodos intermediários como retransmissores para reduzir altas potências em comunicação a longas distâncias. Para muitas aplicações de RSSF, a comunicação por múltiplos saltos se torna necessário na propagação de informações;

- **Execução eficiente em energia:** para atingir longos tempos de atuação, operações eficientes em energia devem ser visadas. No transporte de dados entre dois nodos, a energia gasta é medida em *joule* por *bit*, assim evitar a formação de *hotspots,* para proporcionar um consumo homogêneo de energia, é um técnica válida;

- **Auto configuração:** uma RSSF precisa configurar a maioria do seus parâmetros operacionais de forma autônoma, independente da configuração externa. O grande número de nodos e uma implantação simplificada é requerido na maioria das aplicações. Por exemplo, a rede deve ser capaz de tolerar falhas nos nodos, bem como a integração (reposição) novos nodos;

- **Processamento em rede e colaboração:** em algumas aplicações, um único sensor não é capaz de decidir se um evento aconteceu, mas vários sensores podem colaborar para detectá-lo. Assim, a informação deve ser processada na própria rede para alcançar esta colaboração, em vez de cada nodo transmitir todos os dados a uma rede externa;

- **Localização:** o princípio de localidade deve ser extensivamente visado para assegurar, em particular, a escalabilidade. Nodos, que não dispõem de recursos computacionais robustos, devem tentar evitar sobrecargas em processamento e armazenamento e limitar-se apenas a informações de seus vizinhos diretos. Com isso, o objetivo é permitir que a rede escale um grande número de dispositivos, sem ter que depender de alto processamento em um único nodo;

- **Exploração de *trade-off*:** RSSF devem contar com exploração de *trade-offs* dos objetivos geralmente contraditórios, (por exemplo: maior precisão em resultados exige maior gasto de energia reduzindo o tempo de vida da rede), tanto durante o projeto do sistema e/ou protocolo, como em tempo de execução.

- ***Data centric*:** redes tradicionais são geralmente centradas em torno da transferência de dados entre dois dispositivos específicos, em que cada um é equipado com, pelo menos, um endereço de rede. O funcionamento dessas redes é, portanto, centrado no endereço (*address-centric*). Em uma RSSF, onde os nodos são normalmente implantados de forma redundante para tolerar falhas ou para compensar a baixa qualidade dos equipamentos, a identidade do nodo, no fornecimento de dados, em particular, torna-se irrelevante. O importante são as respostas e valores, e não qual nodo as oferece. Por isso, a mudança de paradigma centrado no endereço para um paradigma centrado em dados (*data-centric*) no projeto de arquitetura e protocolos de comunicação é promissor.

2.4. Nodo Sensor

Caracteristicamente, um nodo sensor é um pequeno dispositivo que inclui três componentes essenciais e um auxiliar: um **subsistema de sensoriamento** para aquisição de sinais a partir do ambiente físico e conversão em dados; um **subsistema de processamento** e armazenamento de dados, um **subsistema de comunicação** sem fio para transmissão dos dados (ANASTASI *et al*., 2009) e um (auxiliar) **subsistema de energia** como fonte geradora de tensão elétrica aos outros subsistemas, esboçados na Figura 2.

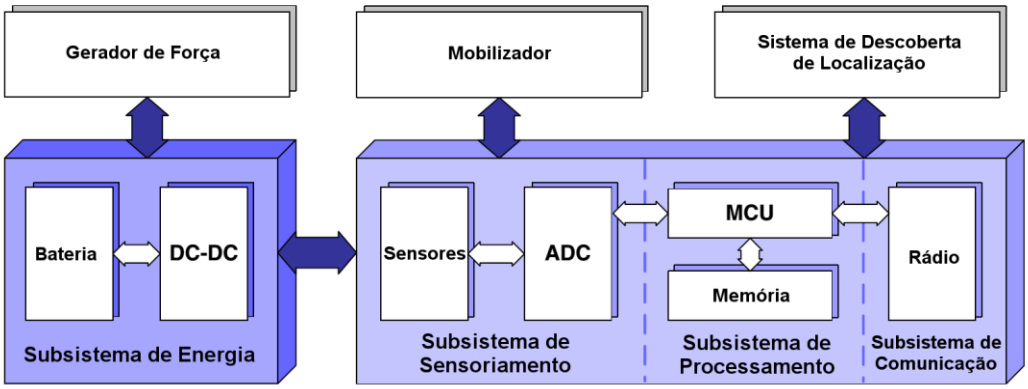


Figura 2: Arquitetura de nodo sensor. Adaptado de Anastasi *et al*. (2009).

Nesta, podemos visualizar componentes como: DC-DC (*Direct Current to Direct Current*) – para conversão de tensão contínua em outra com valor diferente; ADC (*Analog-to-Digital Converter*) – para conversão de sinal analógico em digital, MCU (*Microcontroller Unit*) – para processamento de instruções de programa.

2.4.1. Subsistema de Comunicação

O subsistema de comunicação proporciona uma interface extensível para acomodar diferentes tipos de rádios. Um exemplo é o transceptor *Chipcon* (CC2420), que implementa a especificação rádio IEEE 802.15.4 e fornece uma taxa de transmissão de 250 kbps com mais de 16 canais na banda de 2,4 GHz (DARGIE; POELLABAUER, 2010).

2.4.2. Subsistema de Processamento

O subsistema de processamento agrupa todos os outros subsistemas e alguns periféricos adicionais. Seu principal objetivo é executar instruções referentes à detecção, comunicação e auto-organização. É constituído por um *chip* processador, uma memória não volátil (normalmente memória flash interna) para armazenar instruções de programa, uma memória ativa para armazenar temporariamente os dados monitorados, e um relógio interno. Nesse sistema, podem ser utilizados hardwares mais eficientes em energia como processadores digitais de sinal e FPGA (*Field Programmable Gate Array*) (STELTE, 2010), no entanto a maioria dos nodos sensores existentes utilizam microcontroladores (ROEDIG, *et al*., 2010) pela dinamicidade na instalação de código e atualização.

2.4.3. Subsistema de Sensoriamento

Sensores que podem compor o subsistema de sensoriamento têm como objetivo interagir o mundo virtual com o mundo físico. Na Figura 3 é possível visualizar os componentes típicos de um sensor, bem como o fluxo de dados desde a ocorrência do fenômeno de interesse até sua representação em valores digitais numéricos.

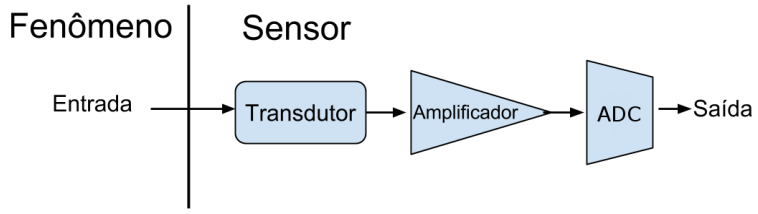


Figura 3: Componentes de hardware sensor. Adaptado de Ni *et al*. (2009).

Esses componentes são: a) transdutor, responsável por converter uma forma de energia em outra (tipicamente em elétrica), b) amplificador, que altera a amplitude do sinal para corresponder melhor à faixa de conversão de analógico para digital, c) ADC, necessário para discretizar os sinais analógicos em dados digitais e fazer interface com um processador.

No processo de conversão realizado pelo componente ADC, dois passos são requeridos (DARGIE; POELLABAUER, 2010):

- 1º O sinal analógico é quantizado (convertido para um sinal discreto a partir de um sinal contínuo). A decisão mais importante nesta fase é determinar o número de valores discretos permitidos.

- 2º A frequência de amostragem deve ser definida. Em RSSF, devido à grande ocorrência de ruído, uma sobre amostragem é requerida.

O que caracteriza a especificação do ADC é a sua resolução, na qual é representada por uma expressão do número de bits que podem ser utilizados para codificar a saída digital.

Para aumentar a precisão do sensor, calibração pode ser necessária, já que condições de calibração de fábrica nem sempre condizem às condições do campo (ambiente externo) (NI *et al*., 2009). A Figura 4 esboça uma curva de calibração de entrada e saída para conversão analógico/digital.

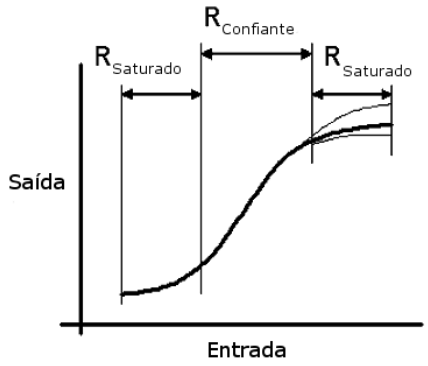


Figura 4: Relação entre entrada e saída de dado de um sensor. Adaptado de Ni *et al*. (2009).

A faixa de detecção total, é constituída pelo intervalo de operação confiante ($R_{confiante}$) e o intervalo saturado ($R_{saturado}$). O intervalo $R_{confiante}$ é geralmente linear e deve consistir de um mapeamento um-para-um de valores de saída para valores de entrada. Dependendo do tipo do sensor, podem existir diferentes graus de variabilidade fora do intervalo de operação confiável.

2.4.4. Subsistema de Energia

O subsistema de energia geralmente consiste de uma bateria que fornece tensão DC (*Direct Current*) a todos os outros subsistemas para manter componentes ativos tais como osciladores, amplificadores, registradores e contadores. Além disso, fornece conversores DC-DC para que cada subsistema possa obter a quantidade certa de tensão.

Em particular, é possível coletar energia a partir do ambiente externo, por exemplo, utilizando células solares como fonte de energia. No entanto, fontes externas geralmente exibem um comportamento não-contínuo, de modo que um acumulador de energia (bateria ou capacitor) se mostra indispensável.

2.5. Falha de Dados em RSSF

De uma perspectiva técnica, o nodo sensor converte sinais do mundo físico em parâmetros (dados) que podem ser medidos e processados por um sistema de computação. Esta é a tarefa primordial de uma RSSF (RENJITH e BABURAJ, 2012), no entanto sujeito a falhas. Em implantações reais de RSSF verifica-se que uma porção substancial dos dados recolhidos são, na verdade, defeituosos (SHARMA *et al.*, 2010).

Segundo Kamal *et al.* (2013), em torno de 50% dos dados coletados em aplicações de

RSSF são defeituosos. Como nodos sensores são normalmente implantados em áreas de difícil acesso, a verificação e manutenção manual se mostra demorado e/ou custosa. Assim, o melhoramento da confiabilidade dos dados recolhidos pela RSSF é um ponto pertinente no aprimoramento dessa tecnologia e formas que proporcionem a identificação rápida de falhas em dados de nodos sensores contribuem para o aumento de tal confiabilidade.

Diante de análises de dados, provenientes de aplicações de RSSF, realizadas em pesquisas como a de Ni *et al*. (2009), uma taxonomia para falhas de dados que podem ocorrer no processo de conversão de sinais foi definida e sistematicamente caracterizada. Estas falhas e suas possíveis causas podem ser classificadas como:

- *Outlier*: leituras isoladas cujos valores possuem distância numérica de padrões esperados. As causas são muitas vezes desconhecidas;

- *Spike*: múltiplos dados com uma distância numérica muito maior que a faixa de mudança esperada. As causas frequentes incluem falha da bateria, de hardware ou de conexão;

- "*Stuck*-at": valores que se mantêm com variação zero durante um período de tempo não esperado. Geralmente causado por mal funcionamento do hardware do sensor;

- Grande quantidade de ruído ou variância: quando a variância dos dados lidos é maior que o esperado. Pode ser devido a falha de hardware, ambiente fora de faixa, ou enfraquecimento da bateria;

- Calibração: o sensor reporta valores muito além do que se consideram dados verdadeiros. A principal causa é a ocorrência de erro no processo calibração (o ganho do sensor pode estar diferente do ideal);

- Conexão ou hardware: mal funcionamento do hardware que causar falta de precisão. Perturbações externas e idade do sensor podem proporcionar maior probabilidade de seu fracasso. Outras causas incluem curto-circuito ou fio(s) solto(s);

- Bateria fraca: voltagem de alimentação abaixo do limiar necessário com que o sensor consegue reportar dados confiáveis;

- Ambiente fora de faixa: o ambiente excede a faixa de sensibilidade do hardware

transdutor do sensor.

- *Clipping*: a leitura dos dados se mantêm em um valor extremo superior ou inferior. Isto é devido ao meio ambiente exceder os limites do conversor analógico/digital.

Um sensor pode apresentar uma ou mais falhas simultaneamente, dependendo do contexto da aplicação. Porém, é importante saber que mesmo na presença de falhas, alguns dados ainda poderão ter valores informativos, enquanto outros serão totalmente sem interpretação. Nesse último caso, como consequência, os dados devem ser descartados.

Impactos relacionados a ocorrência dos tipos de falhas definidas anteriormente são (NI *et al*., 2009):

- *Outlier*: pode distorcer significativamente estatísticas como: média, variância, gradiente. Não oferece qualquer informação útil e pode ser descartada;

- *Spike*: geralmente não possuem qualquer valor informativo e deve ser descartado. Isto resulta em uma perda de rendimento de dados do sensor;

- "*Stuck*-at": se o ambiente se encontra fora de faixa do sensor, os dados ainda podem ser interpretados, porém com baixa fidelidade. Caso contrário, os dados podem ser descartados;

- Grande quantidade de ruído ou variância: se os dados ruidosos se aproximam em valor dos dados de outros sensores, então ainda oferecem utilidade e não devem ser descartados;

- Calibração: os dados não devem ser descartados. Estes podem ainda fornecer informações e um método de calibração adequado pode corrigi-los;

- Conexão ou hardware: quando o sensor não está funcionando como projetado, não faz sentido utilizar seus dados, portanto devem ser descartados;

- Bateria fraca: geralmente, uma falha no suprimento de energia resulta em dados inúteis que devem ser descartados. No entanto, se o comportamento do sensor em baixa tensão gera ruído adicional, ainda pode haver valor informativo em suas leituras;

- Ambiente fora de faixa: ainda é possível haver informação útil. No mínimo, indica que o ambiente excede a faixa de sensibilidade do sensor;

- *Clipping*: ainda é possível haver informação útil. Indica que os dados excedem valores superiores ou inferiores do conversor analógico/digital.

A fim de exemplificar a ocorrência de falhas no processo de sensoriamento, a Figura 5 apresenta dados de três sensores de CO_2 (Dióxido de Carbono) aplicados em solo.

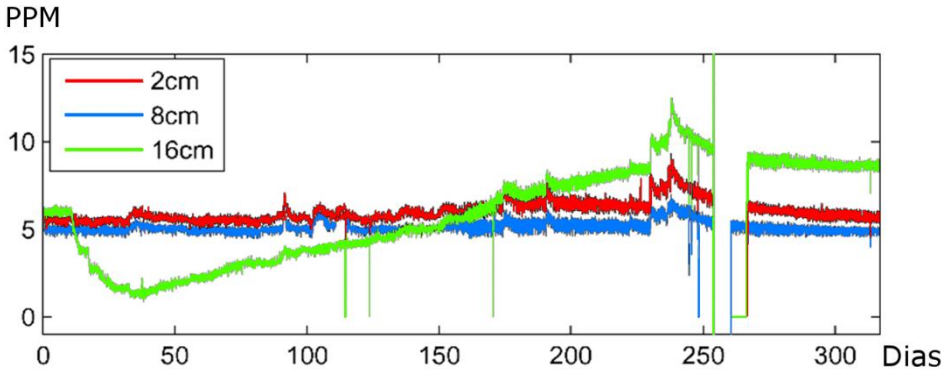


Figura 5: Gráfico de concentração de CO_2 em três profundidades diferentes do solo. Adaptado de Ni *et al*. (2009).

Nesse gráfico, a abscissa horizontal representa o tempo em dias e a ordenada vertical a concentração de CO_2 ppm (partes por milhão). Além disso, três leituras em profundidades de 2 (dois), 8 (oito) e 16 (dezesseis) cm do solo, nas quais duas se assemelham e uma difere em valores. Como pode ser visto, após o dia 250 há a ocorrência de falhas do tipo "*spike*" comum aos três sensores.

Ainda na Figura 5, pode-se observar que o sensor localizado a 16cm possui uma leitura que difere do padrão dos outros dois sensores, caracterizando-se anormal diante da variação considerada acentuada se comparada às outras duas leituras. Assim, tal comportamento pode ser caracterizado como falha de calibração.

Incrementar a confiabilidade do processo de sensoriamento dos nodos pode ser feito por meio do diagnóstico e detecção de falhas. Para isso, técnicas baseadas em regras, estimativas ou aprendizagem podem ser aplicadas (WARRIACH *et al*., 2012), onde estas se utilizam de um conjunto leituras provenientes dos sensores. No entanto, existe a necessidade de se conhecer valores ideais que o sensor deve reportar, para ser possível a identificação de

desvios em dados. Assim, garantir a veracidade do processo de sensoriamento dos nodos antes da implantação da rede é de grande importância.

Uma forma prover tal garantia é analisando os dados reportados pelos nodos antes de sua inserção no meio, em busca de possíveis falhas, pois possibilitará o conhecimento a respeito do comportamento de variações numéricas no processo de aquisição de sinais físicos do ambiente.

Utilizando-se novamente do exemplo citado anteriormente (Figura 5), por se tratar de uma aplicação em ambiente não controlado, não se pode afirmar com exatidão todas as causas de falhas no sensor implantado a 16 cm, se seu hardware sofreu algum dano durante o processo de dispersão no campo ou se já possuía defeito. Levando em consideração que no sensor em específico houve um declive seguido de ascendência na leitura, tal comportamento pode estar relacionado aos níveis decrescentes de energia da bateria.

Assim, sendo requerido o conhecimento prévio sobre o funcionamento dos sensores, no que diz respeito a corretude de seus dados, ou na ausência disto, o conhecimento prévio de variações ou falhas que estes possam apresentar, como um processo que antecede decisões relacionadas à técnicas para prover confiabilidade, desenvolveu-se um sistema para análise de dados de tarefa de sensoriamento de RSSF.

A análise a ser realizada pelo sistema tem como principal cenário a simulação de consumo de energia, pois é um recurso essencial ao funcionamento do sensor, e a falta ou redução deste pode ocasionar a ocorrência de falhas em dados como do tipo: *spike,* grande quantidade de ruído e bateria fraca, assim como, também, há a possibilidade de outros tipos falhas se manifestarem.

3. DESENVOLVIMENTO

Neste capítulo são descritos processos que compõem um sistema desenvolvido para análise de dados de nodos sensores. São estes o controle de tensão no nodo e apresentação dos dados sensoriados. Em sequência, nodos sensores MICAz[7] são analisados no estudo de caso, onde foi identificado um desvio em leituras de uma das variáveis físicas do ambiente, além de um método de correção que foi aplicado.

3.1. Controle de Tensão

Analisar nodos de uma RSSF até níveis baixos de energia tem como requisito alguma forma de se controlar sua fonte de energia, pois em tal análise a utilização de baterias torna essa tarefa muito longa em relação ao tempo.

O controle de energia de um dispositivo eletrônico, no caso de um nodo sensor, pode ser feito por equipamentos especializados, fabricados para tal função. No entanto, fatores como disponibilidade, custo de aquisição, precisão da escala de tensão e forma de controle podem impor barreiras em sua utilização.

Assim um circuito analógico formado por componentes básicos de eletrônica para prover tensão contínua controlável foi desenvolvido, de forma que atendesse a escala de operação e pudesse ser controlado via software executando em um computador. O esquema desse circuito pode ser visualizado na Figura 6.

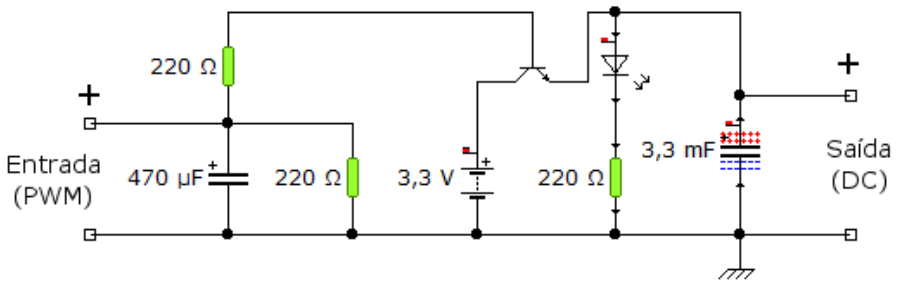


Figura 6: Esquema eletrônico de circuito controlador de tensão.

A entrada recebe um sinal em forma de onda com variação de largura chamado PWM (*Pulse Width Modulation*) e na saída é disponibilizada tensão contínua. O circuito foi projetado para prover o controle de energia a partir 0 (zero) volts até o limite definido por um

[7] http://www.memsic.com/wireless-sensor-networks/MPR2400CB

gerador auxiliar de energia. Neste caso foi utilizado um gerador de 3,3 (três vírgula três) volts no estudo de caso.

Devido ao requisito do circuito ser comandado a partir de uma aplicação em computador, houve a obrigatoriedade de utilizar um dispositivo como interface para intermediar a comunicação. O Arduino por possibilitar tal funcionalidade, fácil programação (BANZI, 2011) e ser de baixo custo, foi escolhido. No entanto dispositivos como FPGAs (KILTS, 2006), beagleboard[8], dentre outros, poderiam desempenhar tal função, porém de forma subutilizada se comparados ao Arduino.

Por meio de portas digitais específicas do Arduino é possível a simulação de variação de tensão utilizando sinal PWM. Um exemplo de como ocorre esse controle pode ser observado na Figura 7.

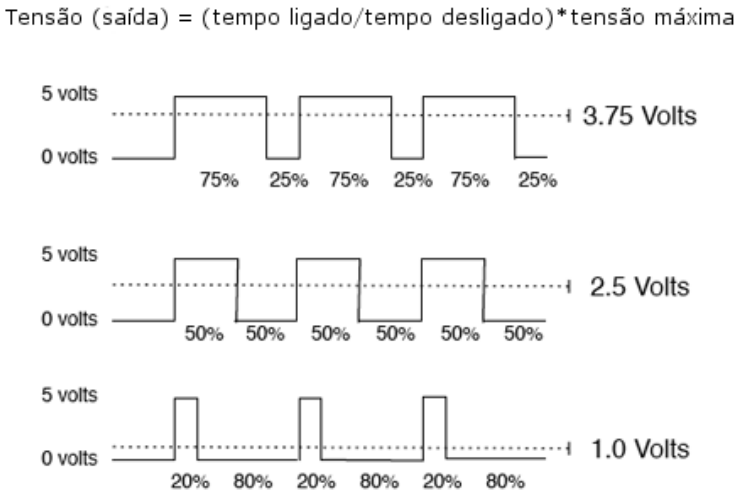


Figura 7: Variação de tensão por sinal PWM.

Variando a largura do pulso é possível controlar a potência média aplicada a uma carga. Assim, quando a largura do pulso varia de zero até o máximo, a tensão média também varia na mesma proporção.

3.2. Apresentação de Dados

Na escolha de como apresentar os dados coletados dos sensores, deve-se ter em mente vantagens e desvantagens a respeito de qual forma ser adotada. Em se tratando de variações mínimas de valores provenientes de sensores com alta precisão, a representação textual, por

[8] http://beagleboard.org/

exemplo, por meio de tabelas, pode prover um maior entendimento a respeito de pequenas mudanças. Por outro lado, em se tratando de variações bruscas, recursos visuais podem proporcionar ao leitor um melhor e mais rápida identificação.

Uma vez que é recorrente a utilização de gráficos para a verificação de desvios em dados coletados por nodos sensores, constatado em trabalhos como de Anastasi *et al*. (2004), Ni *et al*. (2009) e Sharma *et al.* (2010), esta foi a forma escolhida para apresentação.

Portanto, foram utilizados gráficos de dados *versus* tempo e tensão *versus* tempo para a identificação de possíveis desvios em valores lidos pelos nodos, avaliados pelo sistema desenvolvido.

No processo de plotagem em gráficos, o modo como são obtidos os dados varia de acordo com a tecnologia de comunicação entre computador e rede, podendo ocorrer por meio de conexão direta com a RSSF ou até mesmo por consulta a partir de um banco de dados em um servidor. Portanto, a forma com que o sistema obtém os dados da RSSF não é estrita. No estudo de caso em específico, a forma com a qual se obtiveram dados será descrita.

Por conseguinte, para tal apresentação, uma aplicação desktop foi feita em linguagem Java[9]. A determinação das linguagens para implementação de todo o código do sistema foi opcional somente para plataforma desktop, ao contrário da plataforma Arduino. Linguagens como C++[10], python[11], dentre outras, poderiam ser adotadas, no entanto foi escolhida a Java devido possuir propriedades como reutilização de código, orientação objeto, facilidade na criação de interface com janelas, botões, menus, etc, e bibliotecas auxiliares como JFreeChart[12] 1.0.17 para plotagem de informações em gráficos (apresentados na Sessão 3.3) e librxtx-java[13] para comunicação serial USB (*Universal Serial Bus*) com Arduino.

3.3. Estudo de Caso: Análise de Nodos MICAz com Placa Sensora MTS400

Em um curto espaço de tempo as RSSF têm feito pesquisadores e engenheiros empregarem cada vez mais tecnologias em novas aplicações. Principalmente em nodos sensores MICAz, produzidos pela Memsic *Tecnology*[14], uns dos mais comumente usados e

[9] http://docs.oracle.com/javase/7/docs/api/
[10] http://www.cplusplus.com/
[11] http://www.python.org.br/wiki/
[12] http://www.jfree.org/jfreechart/
[13] http://playground.arduino.cc//Linux/Ubuntu/
[14] http://www.memsic.com/

razoavelmente fáceis de implementar (ALI *et al*., 2011). No entanto, adotar apenas manuais técnicos de módulos como esse pode não ser suficiente se informações importantes para determinadas situações não estiverem disponíveis. Assim, a fim de contribuir ainda mais em pesquisas em RSSF, um kit de sensores foi utilizado como alvo de análise.

Tal kit denominado WSN-PRO2400CA[15] possui 6 (seis) nodos compostos por módulo (de processamento e rádio) MICAz e placa sensora MTS400, além de uma estação base (sorvedouro) para transferência de dados entre computador e rede. A programação dos nodos se dá por meio de linguagem NesC/TinyOS[16], e as principais especificações técnicas de hardware relacionadas ao kit podem ser verificadas no Apêndice A.

O processo de análise foi realizado sequencialmente em cada um do nodos disponíveis devido características de tensão e corrente do circuito controlador não suportarem mais de um dispositivo por vez. Na Figura 8 é possível visualizar os componentes envolvidos no processo de análise, onde são necessários: nodo sensor, estação base, Arduino, circuito controlador, conexões USB, computador e aplicação desktop para controle de tensão e apresentação de dados em gráficos.

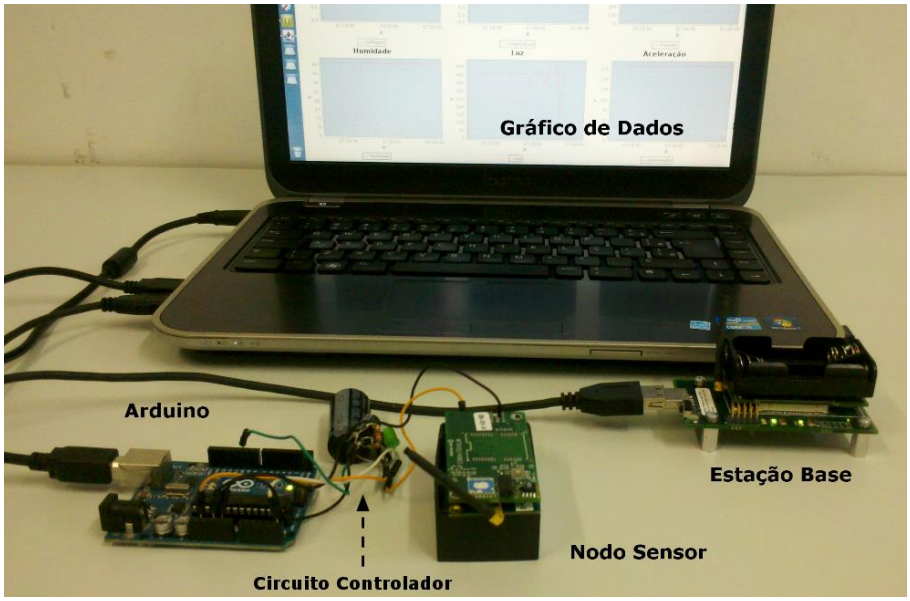


Figura 8: Sistema em execução de análise de nodo sensor.

Exceto entre o circuito de controle de tensão e nodo sensor, cuja interligação transporta somente corrente elétrica contínua, o sistema como um todo é interligado via

[15] http://www.memsic.com/wireless-sensor-networks/WSN-PRO2400CA/
[16] http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page

conexão de dados por fios ou pelo ar. A Figura 9 esboça de forma mais clara as interligações entre componentes envolvidos na análise.

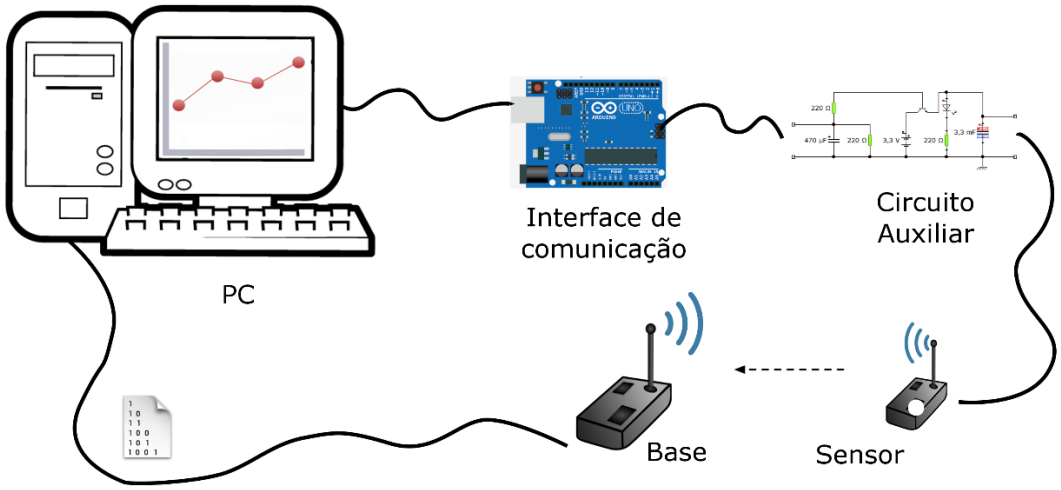


Figura 9: Conexão entre dispositivos envolvidos na análise de dados.

Na execução do sistema, o fluxo de mensagens é iniciado a partir da aplicação desktop que envia constantemente um valor numérico inteiro de 0 (zero) à 255 (duzentos e cinquenta e cinco) para a aplicação no Arduino, cujo código fonte pode ser verificado no Apêndice B. Este valor representa a variação da largura do pulso da porta PWM que é enviado ao circuito eletrônico controlando a tensão de entrada no nodo sensor de 0 (zero) volts até o limite de 3,3 (três vírgula três) volts.

Por sua vez, o sensor, uma vez em funcionamento, coleta os dados físicos do ambiente e os envia à estação base através de conexão sem fio (a estrutura de dados utilizada pode ser vista no Apêndice C.1). A estação base, ao receber os dados, os encaminha por comunicação serial USB para a aplicação desktop. Por fim, esta converte os dados em informações com unidades desejadas e as apresenta em gráficos em relação ao tempo. Também, em conjunto, é plotado o nível de energia do sensor no instante em que foi realizada cada leitura. A partir daí, qualquer variação nas informações pode ser percebida de forma visual.

Na Figura 10 pode ser visualiza a interface da aplicação desktop que contêm os gráficos dos dados coletados de um dos nodos sensores utilizados, relacionados ao seu nível de energia (em volts), umidade relativa do ar em porcentagem (%HR), pressão atmosférica em milibar (mbar), luminosidade do ambiente em lux, temperatura em graus célsius (°C) e aceleração no eixo Y com valor bruto, ou seja, sem conversão, devido ausência (nos manuais da Memsic) de fórmula específica para tal.

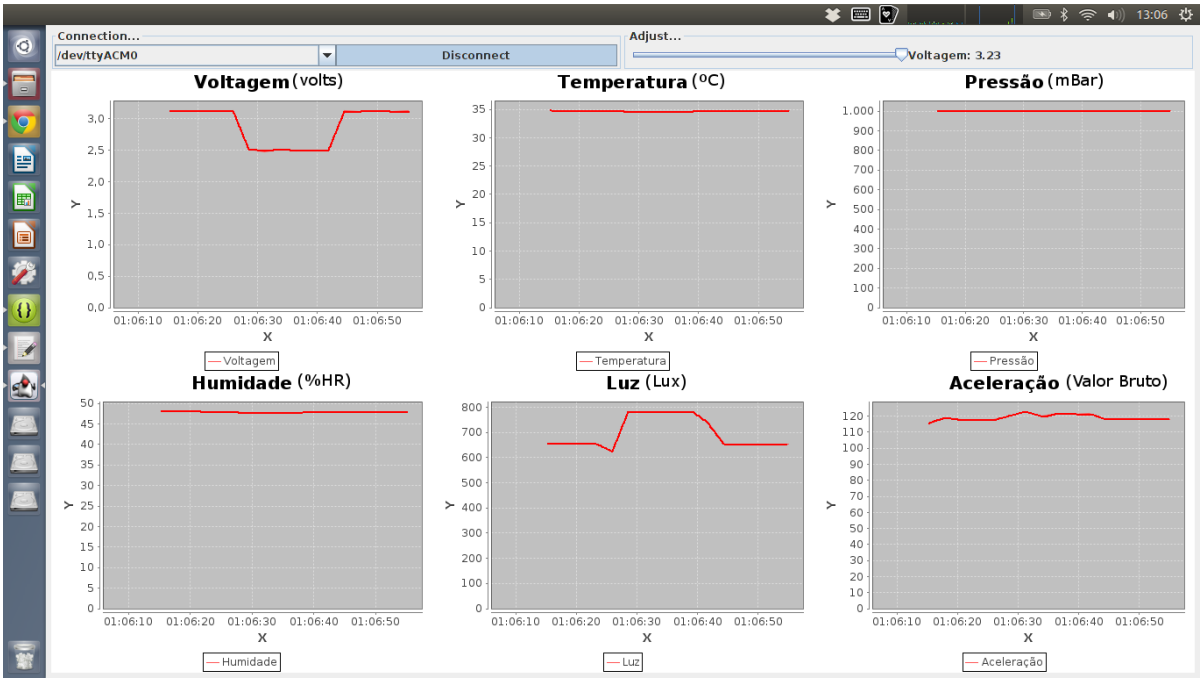


Figura 10: Aplicação desktop em execução.

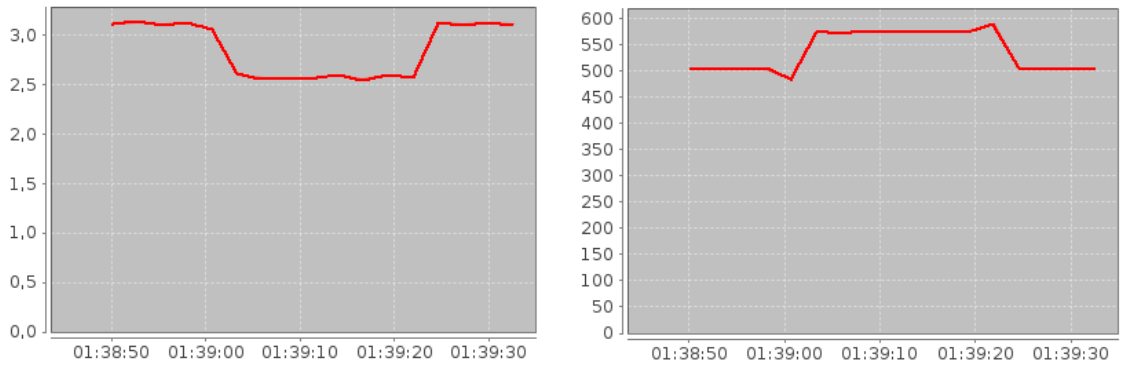
Funções auxiliares da aplicação desktop permitem selecionar a porta de comunicação com Arduino, estabelecer conexão com este e controlar a tensão no nodo sensor. A conexão com estação base é uma funcionalidade que não necessita acionamento manual.

O limite de tensão de operação recomendado para os nodos sensores utilizados é de 3,6 (três vírgula seis) volts à de 2,7 (dois vírgula sete) volts (MEMSIC, 2010b). No entanto, a faixa de tensão inicial aplicada foi de 3,3 (três vírgula três) volts com decréscimo até por volta de 2,5 (dois vírgula cinco) volts. Decidiu-se utilizar esse limite inferior, pois em tensões abaixo desse valor constatou-se que os hardwares sensores passam a apresentar mal funcionamento.

Fale a pena ressaltar que diante da possiblidade de inconsistência nos valores do gráfico de nível de energia, pois estes são enviados pelo próprio nodo, uma leitura auxiliar de tensão é feita pelo Arduino, nos terminais de saída do circuito de controle de tensão, para fins de verificação da corretude dessa informação.

Na análise executada nos nodos, era esperado que não houvesse alteração nos valores lidos enquanto o nível de energia se mantivesse dentro dos limites estabelecidos pelo fabricante. Porém, durante a simulação do enfraquecimento do suprimento de energia notou-se nos 6 (seis) nodos utilizados o mesmo comportamento de desvio nos dados de

luminosidade, como pode ser visto no gráfico da Figura 11. Conforme fosse decrescida a tensão elétrica, o valor de luminosidade era incrementado, ou à medida que a tensão elétrica fosse aumentada, o valor de luminosidade sofria diminuição.



a. b.

Figura 11: Variação em dados captados: a → tensão (volts) no nodo sensor, b → luminosidade (lux) captada pelo nodo sensor.

Em se tratando de uma aplicação real para monitoramento e controle de ambiente, seja industrial, hospitalar, de produção, variações bruscas desse tipo poderiam ocasionar prejuízos financeiros ou até de vidas humanas. Pode ser mencionado como exemplo um sistema de controle automático de luminosidade em estufas em que dados errôneos poderiam ocasionar a diminuição de produtividade ou até morte de uma cultura que necessitasse de quantidades específicas de luz solar.

Na tentativa de categorizar o desvio identificado, características como: leituras não isoladas, valores além do considerado verdadeiro, comportamento contínuo (conforme a variação de energia) e dentro da faixa de operação do sensor, levam crer que se assemelha à uma falha de calibração, conforme abordado na Sessão 2.5.

Uma vez que não se faz praticável a troca do hardware, devido ao acoplamento físico, e o problema ser comum a todos os sensores avaliados, soluções em software se tornam alternativas. No caso identificado, uma correção por meio de fórmula matemática a nível de software pôde ser aplicada (Apêndice D).

Em uma avaliação dos dados brutos do sensor, antes de sua conversão em informação útil de luminosidade, pôde-se perceber que, independentemente da quantidade de luz no ambiente, a variação nesses dados se mostrava na ordem de 4 (quatro) inteiros para luz visível

de 3 (três) inteiros para luz infravermelha[17], no intervalo de 3,3 (três vírgula três) volts à 2,5 (dois vírgula cinco) volts. A medida que o nível de energia diminuísse, tal variação ocorria linearmente.

Assim, pôde-se perceber que havia uma relação de variação dados por tensão:

R = VAR / (VBATi – VBATf), onde:

- VAR: variação do dado bruto percebida;
- VBATi: valor ideal de tensão → 3,3 (três vírgula três) volts;
- VBATf: valor mínimo de tensão → 2,5 (dois vírgula cinco) volts;
- R: relação de variação por tensão;

Com essa relação, chegou-se a seguinte fórmula de ajuste:

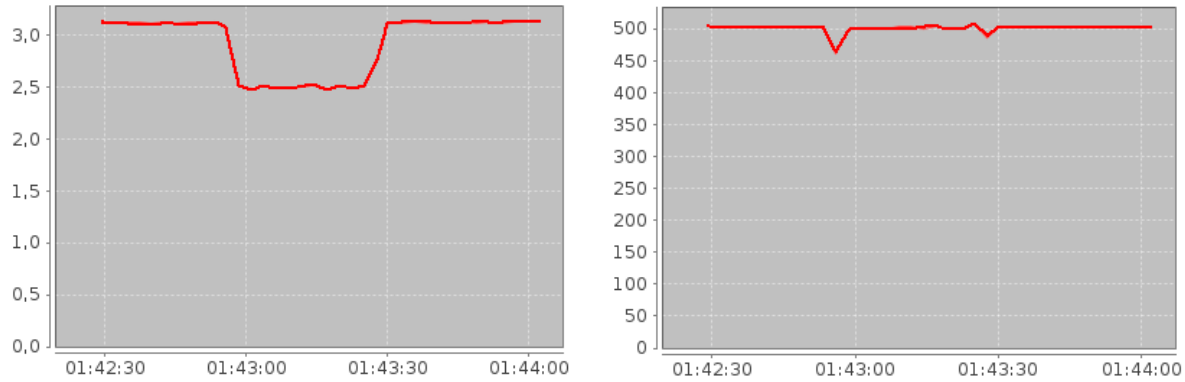
DATAr = DATA – (VBATi -VBAT) * R, onde:

- DATA: é o dado bruto lido do sensor;
- VBAT: é a tensão atual da bateria em volts;
- DATAr: dado bruto ajustado a ser aplicado no cálculo de luminosidade.

Portanto, uma vez que o dado bruto do sensor sofre aumento, conforme o nível de tensão decresça, por meio dessa formula, é subtraído do mesmo dado bruto o montante necessário para normalizar tal variação.

Como resultado da aplicação dessa fórmula, obteve-se maior constância nas informações de luminosidade captadas do ambiente pelo nodo sensor, mesmo com variação de seu nível de energia, evidenciado no gráfico da Figura 12.

[17] No cálculo de luminosidade, são utilizadas duas variáveis numéricas provenientes do sensor TSL2550D: luz visível e luz infravermelha.



a. b.

Figura 12: Menor variação em dados captados: a → tensão (volts) no nodo sensor, b → luminosidade (lux) captada pelo nodo sensor.

Vale a pena ressaltar que, além do nível de tensão, existem outros fatores que podem alterar o funcionamento do sensor. Como exemplo, temos a temperatura, humidade, vibração, que podem causar variações em leituras, de acordo com o tipo do sensor.

Contudo, é de extrema importância observar que tal imprecisão detectada no sensor de luminosidade tem uma grande probabilidade de passar despercebida em aplicações reais, pelo fato de poderem ser atribuídas à mudanças de fenômenos climáticos. De outro ponto de vista, mesmo sendo identificado o tipo de falha, seu(s) causador(es) pode(m) ser confundido(s) com mal funcionamento do hardware, por tempo de utilização ou por ação de fenômenos externos. Porém, neste dois casos, uma forma de contornar tal problema, pode ser provida por meio da utilização de sensores de tecnologias diferentes, pois dificilmente estariam sujeitos aos mesmos tipos de falha em dados, ou as mesmas condições que levariam a falhar.

4. CONCLUSÕES

Em RSSF, a conversão de sinais do mundo físico em dados é um processo essencial, no entanto sujeito a falhas devido fatores como, por exemplo, utilização de hardware de baixo custo e atuação em regiões hostis. Antes de sua implantação, é esperado que a rede esteja em perfeito funcionamento no que diz respeito a atender condições de corretude em seu sensoriamento, conexão, execução de tarefas e protocolos.

Para garantir tais requisitos, análises como de tráfego, dados ou consumo de energia podem ser feitas. Dentre estas, a verificação prévia de dados visa a identificação de alterações indesejadas no processo de sensoriamento, possibilitando a execução de medidas cabíveis como, por exemplo, técnicas de correção ou tolerância à falhas.

Uma forma de prover tal verificação é coletando e analisando, por meio de comparação, uma quantidade determinada de amostras (dados) de nodos sensores. Dessa forma, em vista disso, foi desenvolvido um sistema que possibilitasse a análise de dados de uma RSSF, capaz de simular o esgotamento de energia de nodos sensores. Este requisito foi definido pelo fato do consumo de energia ser inevitável em aplicações reais de tais redes, assim como, também, por se tratar de um dos causadores de falhas em dados, provenientes da tarefa de sensoriamento.

Em teste do sistema, no estudo de caso, nodos reais foram analisados em funcionamento antes da implantação no ambiente, onde se tinha conhecimento a respeito de valores numéricos desejados. Em comparação entre tais dados e os reportados pelos nodos, pôde-se identificar a presença de desvios em leituras de uma das variáveis físicas sensoriadas. Assim, foi percebido que conforme ocorresse o decaimento do nível de energia, mesmo dentro do limiar correto de operação, havia a ocorrência de alterações bruscas em valores luminosidade repassados pelo sensor específico presente nos nodos.

O padrão de alteração identificado teve relação diretamente proporcional ao nível de energia entregue ao nodo sensor: à medida que o nível de energia diminuísse, a variação do dado bruto ocorria, linearmente, de forma crescente. A partir desse padrão, foi possível propor uma correção por meio de fórmula matemática, a nível de software. Tal fórmula estabeleceu uma relação entre dado e nível de energia onde, conforme o valor de tensão fosse decrementado, por consumo, uma quantia numérica era subtraída do dado bruto obtendo-se a

normalização deste. Todavia o método de correção proposto se aplica em específico ao sensor avaliado.

Por meio do sistema, conclui-se que esse tipo de análise é pertinente na verificação de corretude do sensoriamento de variáveis físicas do ambiente desempenhado por nodos de uma RSSF, assim como, também, constatou-se que especificações de fabricantes são passíveis de serem contestadas. Todavia, contribuição do sistema é permitir um estudo prévio sobre a integridade dos dados reportados por nodos sensores, assim possibilitando a execução de medidas antecipadas à implantação da rede, como, por exemplo, técnicas de correção ou tolerância a falhas, ou até mesmo o conhecimento de que determinado nodo possui alterações indesejadas em suas leituras.

Ainda sobre o sistema, este é apenas um protótipo e como trabalhos futuros podem ser realizados, por exemplo, o desenvolvimento de funcionalidades adicionais para detecção automática de desvios em dados, melhoramento do circuito de controle de energia para suportar a análise de mais de um nodo simultaneamente.

Por fim, também podem ser feitas a verificação de outras formas de correção, a nível de software ou hardware, para o mesmo ou diferentes problemas que venham a ser identificados por meio da análise outros tipos de sensores, sejam de tecnologia, fabricante, ou arquitetura diferentes.

REFERÊNCIAS

ALI, N.A.; DRIEBERG, M. and SEBASTIAN, P. 2011. **Deployment of MICAz mote for Wireless Sensor Network applications**. Computer Applications and Industrial Electronics (ICCAIE), 2011 IEEE International Conference on, vol., no., pp.303, 308, 4-7. doi: 10.1109/ICCAIE.2011.6162150.

ANASTASI, G.; CONTI, M.; FRANCESCO, M. D. and PASSARELLA, A. 2009. **Energy conservation in wireless sensor networks: A survey**. Ad Hoc Netw. 7, 3 (May 2009), 537-568. DOI=10.1016/j.adhoc.2008.06.003 <http://dx.doi.org/10.1016/j.adhoc.2008.06.003>.

ANASTASI, G.; FALCHI, A.; PASSARELLA, A.; CONTI, M. and GREGORI, E. 2004. **Performance measurements of motes sensor networks**. In Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM '04). ACM, New York, NY, USA, 174-181. DOI=10.1145/1023663.1023695 <http://doi.acm.org/10.1145/1023663.1023695>.

ATMEL, 2011. **ATmega128L: Rev. 2467XS–AVR–06/11**. Disponível em: <http://www.atmel.com/Images/2467S.pdf>. Acesso em: 2014.

BANZI, M. **Getting Started with Arduino**. 2nd Edition. 2011. USA: O'Reilly Media, 2011. ISB ISBN: 978-1-449-309879.

DARGIE, W. and POELLABAUER, C. 2010. **Fundamentals of Wireless Sensor Networks: Theory and Practice**. Wiley Publishing. ISBN 978-0-470-99765-9.

JANANI. K.; DHULIPALA V. R. S. and CHANDRASEKARAN, R.M. 2011. **A WSN Based Framework for Human Health Monitoring**. IEEE International Conference on Devices and Communication ICDECOM.2011.5738453, pages (l-5), IEEE2011.

KAMAL, A. R. M.; BLEAKLEY, C. and DOBSON, S. 2013**. Packet-level attestation (PLA): A framework for in-network sensor data reliability**. ACM Trans. Sen. Netw. 9, 2, Article 19 (April 2013), 28 pages. DOI=10.1145/2422966.2422976 http://doi.acm.org/10.1145/2422966.2422976.

KARL, H. and WILLIG, A. 2005. **Protocols and Architectures for Wireless Sensor Networks. John Wiley & Sons**. ISBN: 0-470-09510-5.

KILTS, S. 2006. **Advanced Simulation, in Advanced FPGA Design: Architecture, Implementation, and Optimization**. John Wiley & Sons, Inc., Hoboken, NJ, USA. doi: 10.1002/9780470127896.ch11.

LOUREIRO, A. A.; NOGUEIRA, J. M. S.; RUIZ, L. B.; NAKAMURA, E.; SERÓDIO, C. M. e MINI, R. (2003). **Redes sensores sem fio**. Pág. 179–226. Capítulo 4 Livro texto de mini-cursos do XXI Simpósio Brasileiro de Redes de Computadores (SBRC). ISBN 85-88442-48-5.

LUO, Q.; NI, L. M.; HE, B.; WU, H. and XUE, W. 2004. **MEADOWS: modeling, emulation, and analysis of data of wireless sensor networks**. In Proceedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004 (DMSN '04). ACM, New York, NY, USA, 58-67. DOI=10.1145/1052199.1052210 <http://doi.acm.org/10.1145/1052199.1052210>.

MAHAPATRO, A. and KHILAR, P.M., **Fault Diagnosis in Wireless Sensor Networks: A Survey**, Communications Surveys & Tutorials, IEEE , vol.15, no.4, pp.2000,2026, Fourth Quarter 2013 doi: 10.1109/SURV.2013.030713.00062.

MEMSIC, 2010a. **XServe USER MANUAL.** Document Part Number: 7430-0111-02 RevA. Disponível em: <http://www.memsic.com/userfiles/files/User-Manuals/xserve_users_manual-7430-0111-02_a-t.pdf>. Acesso em: 2014.

MEMSIC, 2010b. **MTS/MDA Sensor Board USERS MANUAL.** Document Part Number: 7430-0020-05 Rev B. Disponível em: <http://www.memsic.com/userfiles/files/User-Manuals/mtsmda-sensor-board-users-manual_7430-0020-05_b-t.pdf>. Acesso em: 2014.

MENDES, L. D. P. and RODRIGUES, J. J. P. C. 2011. Review: **A survey on cross-layer solutions for wireless sensor networks**. J. Netw. Comput. Appl. 34, 2 (March 2011), 523-534. DOI=10.1016/j.jnca.2010.11.009 <http://dx.doi.org/10.1016/j.jnca.2010.11.009>. Ding, W. and Marchionini, G. 1997. A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

NI, K.; RAMANATHAN, N.; CHEHADE, M. N. H.; BALZANO, L.; NAIR, S.; ZAHEDI, S.; KOHLER, E.; POTTIE, G.; HANSEN, M. and SRIVASTAVA, M. 2009. **Sensor network data fault types**. ACM Trans. Sen. Netw. 5, 3, Article 25 (June 2009), 29 pages. DOI=10.1145/1525856.1525863 <http://doi.acm.org/10.1145/1525856.1525863>.

NI, L.M.; LIU, Y. and ZHU, Y. 2007. **China's national research project on wireless sensor networks**. Wireless Communications, IEEE, vol.14, no.6, pp.78, 83, doi: 10.1109/MWC.2007.4407230.

PAK, W. and BAHK, S. 2012. **Centralized route recovery based on multi-hop wakeup time estimation for wireless sensor networks with ultra low duty cycles**. Comput. Commun. 35, 11 (June 2012), 1355-1367. DOI=10.1016/j.comcom.2012.03.014 <http://dx.doi.org/10.1016/j.comcom.2012.03.014>.

Proceedings of the Distributed Sensor Nets Workshop, 1978. Pittsburgh, USA. Department of Computer Science, Carnegie Mellon University. Disponível em: <http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA143691> Acesso em: 2014.

RAJASEGARAR, S.; BEZDEK, J. C.; LECKIE, C. and PALANISWAMI, M. 2007. **Analysis of Anomalies in IBRL Data from a Wireless Sensor Network Deployment**. In Proceedings of the 2007 International Conference on Sensor Technologies and Applications (SENSORCOMM '07). IEEE Computer Society, Washington, DC, USA, 158-163. DOI=10.1109/SENSORCOMM.2007.27 <http://dx.doi.org/10.1109/SENSORCOMM.2007.27>.

RENJITH, P.N. and BABURAJ, E. **An analysis on data aggregation in Wireless Sensor Networks**, Radar, Communication and Computing (ICRCC), 2012 International Conference on , vol., no., pp.62,71, 21-22 Dec. 2012 doi: 10.1109/ICRCC.2012.6450549.

REZAEI, Z. and MOBININEJAD, S. 2012. **Energy Saving in Wireless Sensor Networks**. International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.1, February 2012. DOI: 10.5121/ijcses.2012.3103.

ROEDIG, U.; RUTLIDGE, S.; BROWN, J. and SCOTT, A. 2010. **Towards multiprocessor sensor nodes**. In Proceedings of the 6th Workshop on Hot Topics in Embedded Networked Sensors (HotEmNets '10). ACM, New York, NY, USA, Article 16, 5 pages. DOI=10.1145/1978642.1978663 <http://doi.acm.org/10.1145/1978642.1978663>.

SHARMA, A. B.; GOLUBCHIK, L. and GOVINDAN, R. 2010. **Sensor faults: Detection methods and prevalence in real-world datasets**. ACM Trans. Sen. Netw. 6, 3, Article 23 (June 2010), 39 pages. DOI=10.1145/1754414.1754419 <http://doi.acm.org/10.1145/1754414.1754419>.

STELTE, B. 2010. **Toward development of high secure sensor network nodes using an FPGA-based architecture**. In Proceedings of the 6th International Wireless Communications and Mobile Computing Conference (IWCMC '10). ACM, New York, NY, USA, 539-543. DOI=10.1145/1815396.1815521 <http://doi.acm.org/10.1145/1815396.1815521>.

WANG, Q. and BALASINGHAM, I. 2010. **Wireless Sensor Networks - An Introduction**. Wireless Sensor Networks: Application-Centric Design, Yen Kheng Tan (Ed.), ISBN: 978-953-307-321-7, InTech.

Warriach, E. U.; Tei, K.; Nguyen, T. A. and Aiello, M. 2012. **Fault detection in wireless sensor networks: a hybrid approach**. In Proceedings of the 11th international conference on Information Processing in Sensor Networks (IPSN '12). ACM, New York, NY, USA, 87-88. DOI=10.1145/2185677.2185690 http://doi.acm.org/10.1145/2185677.2185690

YICK, J.; MUKHERJEE, B. and GHOSAL, D. 2008. **Wireless sensor network survey. Computer Network.** 52, 12 (August 2008), 2292-2330. DOI=10.1016/j.comnet.2008.04.002 <http://dx.doi.org/10.1016/j.comnet.2008.04.002>.

APÊNDICE A – Especificação Técnica de Sensores MICAz

A.1. Características de Hardware do Professional Kit Memsic

Composto basicamente por estação base MIB520 (MEMSIC, 2010a) para conexão computador/sorvedouro e 6 (seis) nodos sensores formados por: módulo de processamento e comunicação sem fio MICAz, e placa de sensoriamento MTS400 (MEMSIC, 2010b), onde residem os hardwares sensores.

A.1.1. Estação Base MIB520

Fornece interface USB para comunicação computador-rede/rede-computador como mostrado na Figura A.1. Além disso, através de um conector padrão com 51 (cinquenta e um) pinos, qualquer sensor desse kit pode funcionar como estação base (sorvedouro), quando acoplado.

Figura A.1: Estação base MIB520 com módulo MICAz acoplado (MEMSIC, 2010a).

A MIB520 oferece duas portas seriais separadas sobre a conexão USB: uma dedicada a programação e implantação de aplicações em nodos sensores e outra para transmissão de dados. Não necessita de uma fonte externa de energia.

A.1.2. Módulo MICAz: Processador e Rádio (MPR2400)

O MICAz é um módulo utilizado para prover baixo consumo de energia e comunicação sem fio. Este possui, também, um conector padrão com 51 (cinquenta e um) pinos para conexão com placa sensora com a MTS400, e pode ser observado na Figura A.2.

Figura A.2: Módulo de processamento e rádio MICAz[18].

O conjunto formado por processador e rádio é referenciado como MPR2400, baseado em um microcontrolador Atmel de baixa potência ATMEGA128L (ATMEL, 2011), com 16 MHz de frequência, 128 KB de memória programável e 4 KB de memória ativa SRAM, e transceptor *Chipcon* (CC2420), que implementa o padrão IEEE 802.15.4.

A.1.3. Placa de Sensoriamento MTS400

A MTS400, mostrada na Figura A.3, é uma placa de aquisição de dados do ambiente que é acoplada ao módulo de processamento e rádio MICAz.

Figura A.3: Placa de Sensoriamento MTS400 (MEMSIC, 2010b).

Os componentes da placa MTS400 envolvidos na aquisição de dados são:

- Sensor de Aceleração XY (ADXL202JE): possui resolução de ± 2G (onde 1G = 9.81 m/s^2);
- Sensor de Pressão (MS5534AM): com resolução de 300-1100 mbar e ± 1.25% à 25°C;
- Sensor de Luz (TSL2550D): com resposta espectral 400-1000 nm (a luz visível vai de 400 nm à 700 nm);

[18] http://www.memsic.com/wireless-sensor-networks/MPR2400CB

- Sensor de Humidade e Temperatura (SHT11): a) humidade, resolução de 0 à 100% HR e precisão de 0.03% HR, b) temperatura, resolução vai de -40°C à 80°C com precisão de ± 0.5°C.

A.1.4. Especificações de Tensão de Operação

Um nodo sensor do *Professional Kit* Memsic pode ser composto por módulo MICAz e placa sensora MTS400. As especificações de tensão de operação para cada componente verificado no manual são (MEMSIC, 2010b):

- Sensor de Aceleração XY (ADXL202JE): 3,6 (três vírgula seis) volts à 3,0 (três) volts;
- Sensor de Pressão (MS5534AM): 3,6 (três vírgula seis) volts à 2,2 (dois vírgula dois) volts;
- Sensor de Luz (TSL2550D): 3,6 (três vírgula seis) volts à 2,7 (dois vírgula sete) volts;
- Sensor de Humidade e Temperatura (SHT11): 3,6 (três vírgula seis) à 2,4 (dois vírgula quatro) volts;
- Microcontrolador ATMEGA128L: pode operar de 6,0 (seis) volts até 2,4 (dois vírgula quatro) volts, porém o funcionamento normal é de 3,6 (três vírgula seis) volts à 2,4 (dois vírgula quatro) volts.

Não se pode deixar de notar que tais especificações de tensão possuem, de certa forma, disparidade que aumenta ainda mais se levar em consideração que a alimentação do nodo sensor é feita por duas bateria padrões cuja tensão é de 1,5 (um vírgula cinco) volts cada, totalizando um máximo de 3,0 (três) volts. Todavia, a utilização de conversores DC-DC elimina problemas relacionados a tais diferenças de tensões.

APÊNDICE B – Código Fonte do Arduino

```
int pinPwm=10;  // porta de saída de sinal PWM para controle do circuito eletrônico auxiliar
int pinAlg=A5;  // porta de entrada para leitura de tensão (comparação com valor retornado pelo sensor)
int pinDbg=13;  //  porta ligado à led para informar que o sistema está ligado

void setup() {      // Função principal

  Serial.begin(9600);                 // conexão serial via cabo USB entre Arduino e aplicação em PC
  pinMode(pinPwm, OUTPUT);    // sentando modos de operação de porta utilizadas
  pinMode(pinDbg, OUTPUT);
  pinMode(pinAlg, INPUT);
}

int val;
long ini = 0;

void loop() {       // Função de laço infinito

  if(Serial.available()>0){           // Se houver dados advindos da aplicação no PC, leia-os
    val = Serial.read();
    analogWrite(pinPwm, val);       // Escreva o valor na porta PWM
  }


  if(millis() - ini > 500){           // A cada 0.5 segundos leia a tensão do de saída do circuito eletrônico auxiliar
    val = analogRead(pinAlg);
    val = map(val, 0, 1023, 0, 255);
    Serial.write(val);                // E envie o valor à aplicação no PC

    ini = millis();
    digitalWrite(pinDbg,!digitalRead(pinDbg));          // Pisque o Led de debug
  }

}
```

APÊNDICE C – Código Fonte NesC do Sensor MICAz

C.1. Estrutura de Dados

```
// Arquivo de cabeçalho
#ifndef RSSIDEMOMESSAGES_H__
#define RSSIDEMOMESSAGES_H__

enum {
	AM_TYPE = 10 // tipo associado a esta mensagem
};

typedef nx_uint16_t typ;

typedef nx_struct sensor_msg{	// Estrutura de dados que armazenará as informações de leituras
	typ temperature;		// realizadas pelo nodo sensor
	typ humidity;
	typ light;
	typ inflight;
	typ pressure;
	typ x_axis;
	typ y_axis;
	typ voltage;
} sensor_msg;

#endif //RSSIDEMOMESSAGES_H__
```

C.2. Código Principal do Sensor

```
// Arquivo para configuração de componentes
#include "SensorMessage.h"

#define SLEEP 0

module SensorApp {			// componentes a serem utilizados

	uses interface Boot;
	uses interface Leds;

	uses interface State as NodeState;
	uses interface AMSend as Sender;

	uses interface SplitControl as Control;

	uses interface Read<uint16_t>  as Voltage;
	uses interface Read<uint8_t> as VisibleLight;
	uses interface Read<uint8_t> as InfraredLight;
	uses interface Read<uint16_t> as Temperature;

	uses interface Read<uint16_t> as Humidity;
	uses interface Read<uint16_t> as X_Axis;
	uses interface Read<uint16_t> as Y_Axis;
```

```
	uses interface Intersema;


	uses interface Timer<TMilli> as SendTimer;
}

implementation {

	message_t message;		// tipo genérico de pacote para mensagem

	sensor_msg * pacote;		// referência para estrutura que contém as informações
					// a serem enviadas

	event void Boot.booted(){	//evento que indica que o SO iniciou
		call Control.start();	// ative a camada de enlace
		call Leds.led0Toggle();
	}

	event void Control.startDone(error_t result){
		call SendTimer.startOneShot(SLEEP);	// inicie um temporizador para envio
	}						// contínuo de mensagem

	event void SendTimer.fired(){
		call Leds.led1Toggle();
		call Temperature.read();		// inicie a leitura de temperatura
	}

	event void Temperature.readDone(error_t err, uint16_t data){

		// instancie o pacote que guardará as informações a serem enviadas
		pacote = (sensor_msg *)call Sender.getPayload(&message,sizeof(sensor_msg));

		pacote->temperature = data;	// adicione o valor de temperatura sem conversão
		call Humidity.read();		// inicie a leitura de humidade
	}

	event void Humidity.readDone(error_t err, uint16_t data){

		pacote->humidity = data;	// adicione o valor sem conversão
		call Intersema.read();		// inicie a leitura de pressão
	}

	event void Intersema.readDone(error_t err, int16_t* data){

		pacote->pressure = data[1];	// adicione o valor sem conversão
		call X_Axis.read();		// inicie a leitura de aceleração no eixo x
	}


	event void X_Axis.readDone(error_t err, uint16_t data){

		pacote->x_axis = data;		// adicione o valor sem conversão
		call Y_Axis.read();		// inicie a leitura de aceleração no eixo y
	}

	event void Y_Axis.readDone(error_t err, uint16_t data){

		pacote->y_axis = data;		// adicione o valor sem conversão
		call Voltage.read();		// inicie a leitura de tensão atual
	}
```

```
	event void Voltage.readDone(error_t result, uint16_t data){

		pacote->voltage = data;	// adicione o valor sem conversão
		call VisibleLight.read();	// inicie a leitura de intensidade de luz visível
	}


	event void VisibleLight.readDone(error_t err, uint8_t data){

		pacote->light = data;		// adicione o valor sem conversão
		call InfraredLight.read();	// inicie a leitura de intensidade de luz infravermelha
	}

	event void InfraredLight.readDone(error_t err, uint8_t data){

		pacote->inflight = data;	// adicione o valor sem conversão
						// envia o pacote à estação base conectada ao PC
		call Sender.send(AM_BROADCAST_ADDR, &message, sizeof(sensor_msg));

	}

	event void Sender.sendDone(message_t *m, error_t error){
						// espere um tempo para enviar novamente
		call SendTimer.startOneShot(SLEEP);
	}

	event void Control.stopDone(error_t result){}
}
```

C.3. Código de Configuração do Sensor

```
// Arquivo para declaração de uso componentes
#include "SensorMessage.h"

configuration SensorConfig {}


implementation {

	components MainC;		// função principal
	components LedsC;		// controle de leds

	components ActiveMessageC ;	//controle da camada de enlace do sensor

	components new AMSenderC(AM_TYPE) as Sender;

	components new VoltageC();			// Sensor de voltagem
	components new Taos2550C();			// Sensor de luz
	components new SensirionSht11C();		// Sensor de temperatura e humidade
	components new Accel202C();			// Sensor de Aceleração
	components new Intersema5534C();		// Sensor de pressão


	components new TimerMilliC() as SendTimer;	// temporizador
```

```
    components SensorApp as App;                    // aplicação

                                                    // ligando interfaces a componentes do sensor

    // Sistema
    App.Boot -> MainC;                              // componente de Boot
    App.Leds -> LedsC;

    // Mensagem
    App.Control -> ActiveMessageC;
    App.Sender -> Sender;

    // Sensores
    App.VisibleLight -> Taos2550C.VisibleLight;
    App.InfraredLight -> Taos2550C.InfraredLight;
    App.Temperature -> SensirionSht11C.Temperature;
    App.Voltage -> VoltageC;

    App.X_Axis -> Accel202C.X_Axis;
    App.Y_Axis -> Accel202C.Y_Axis;
    App.Intersema -> Intersema5534C.Intersema;
    App.Humidity -> SensirionSht11C.Humidity;


    // Temporizadores
    App.SendTimer -> SendTimer;
}
```

APÊNDICE D – Código Fonte Java Principal

```
//Pacotes a serem utilizados
import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.GridLayout;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import quali.chart.JPanelArduinoControl;
import quali.chart.JPanelChartRecipe;
import quali.sensor.BaseConnection;
import quali.sensor.SensorMessage;

import net.tinyos.message.Message;
import net.tinyos.message.MessageListener;

public class Main extends JFrame {

private JPanel contentPane;

/**
 * Iniciando interface gráfica de aplicação
 */
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
	    public void run() {
			try {
				Main frame = new Main();
				frame.setVisible(true);
			} catch (Exception e) {
				e.printStackTrace();
			}
	    }
	});
}

private static long start, wait;

/**
 * Criando interface gráfica
 */
public Main() {

	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setBounds(100, 100, 650, 450);
	contentPane = new JPanel();
	contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

	setContentPane(contentPane);

	contentPane.setLayout(new BorderLayout(0, 0));

	// Classe de controle do Arduino
	JPanelArduinoControl panelControl = new JPanelArduinoControl();
	contentPane.add(panelControl, BorderLayout.NORTH);
```

```
JPanel gridPanel = new JPanel();
gridPanel.setLayout(new GridLayout(2, 3));
contentPane.add(gridPanel, BorderLayout.CENTER);

// Classes de gráficos
final JPanelChartRecipe jtemp = new JPanelChartRecipe("Temperatura");
final JPanelChartRecipe jhum = new JPanelChartRecipe("Humidade");
final JPanelChartRecipe jpress = new JPanelChartRecipe("Pressão");
final JPanelChartRecipe jlight = new JPanelChartRecipe("Luz");
final JPanelChartRecipe jaccel = new JPanelChartRecipe("Aceleração");
final JPanelChartRecipe jvolt = new JPanelChartRecipe("Voltagem");

gridPanel.add(jvolt);
gridPanel.add(jtemp);
gridPanel.add(jpress);
gridPanel.add(jhum);
gridPanel.add(jlight);
gridPanel.add(jaccel);

// Conexão com estação base
BaseConnection baseStation = new BaseConnection();
baseStation.connect("/dev/ttyUSB1");

start = System.currentTimeMillis();

// observador de mensagens de entrada, advindas de sensores
baseStation.addListener(new MessageListener() {

    @Override
    public void messageReceived(int i, Message msg) {

            if (msg instanceof SensorMessage) {

                    SensorMessage results = (SensorMessage) msg;

                    // Valores brutos recebidos
                    double temp = results.getTemperature();
                    double press = results.getPressure()/10;
                    double hum = results.getHumidity();
                    double light = results.getLight();
                    double inflight = results.getInfraLight();
                    double x = results.getXaxis();
                    double y = results.getYaxis();
                    double volt = results.getVoltage();

                    //CONVERÇÕES

                    // conversão de temperatura
                    temp = -39.4+(0.01*(double)temp);

                    // conversão de humidade relativa do ar
                    hum = ((-2.0468+0.0367*(double)hum -
                                    0.0000015955*((double)hum*(double )hum))+
                                    (temp-25)*(0.01+0.00008*(double)hum));

                    // conversão de tensão de alimentação do sensor
                    volt = ((double)1.223)*((double)1023.0)/((double)volt);
```

```
//*****************     //CALIBRAÇÃO DOS VALORES LUZ

                        //fórmula de calibração aplicada ao dado bruto de luz visível

                        light = light - (3.3-volt)*(4.0/(3.3-2.5));

                        //fórmula de calibração aplicada ao dado bruto de luz infravermelha

                        inflight = inflight - (3.3-volt)*(3.0/(3.3-2.5));

                        // conversão de tensão de valor de luz ambiente

                        double lux = convert_light2((int)light,(int)inflight);


                        // apresentação dos dados
                        jvolt.addDataValue(volt);
                        jtemp.addDataValue(temp);
                        jpress.addDataValue(press);
                        jhum.addDataValue(hum);
                        jlight.addDataValue(lux);
                        jaccel.addDataValue(x);

                        // debug
                        System.out.println("Temperature: " + temp + " ºC");
                        System.out.println("Pressure: " + press + " mBar");
                        System.out.println("Humidity: " + hum + " %RH");
                        System.out.println("Light:    " + light + " raw");
                        System.out.println("InfLight: " + inflight + " raw");
                        System.out.println("Light: " + lux + " Lux");
                        System.out.println("X_axis: " + x);
                        System.out.println("Y_axis: " + y);
                        System.out.println("Voltage: " + volt + " Volts\n");

                        wait = System.currentTimeMillis() - start;
                        start = System.currentTimeMillis();
                    }
                }
            });
}


// Função de conversão de valores brutos de luz infravermelha e luz visível em unidade LUX

double convert_light2(int channel0, int channel1) {
        double count0, count1;
        // decode the raw channel values
        // (this effectively expands the value back to its 12 bit range)
        count0 = Decode(channel0);
        count1 = Decode(channel1);
        if (count0 == 0)
            return 0;

        return ((count0 * 0.46) * Math.exp((-1) * 3.13 * (count1 / count0)));
}

// Função auxiliar utilizada na conversão
double Decode(int data) {

        int step, chordIndex;
```

```
        double stepValue, chordValue, decoded;
        step = data & 0x0f;
        // bits 3:0 are the step
        chordIndex = (data >> 4) & 0x07; // bits 6:4 are the chord index

        stepValue = Math.pow(2, chordIndex);
        chordValue = Math.floor(16.5 * (stepValue - 1));
        decoded = chordValue + (step * stepValue);
        return decoded;
}


}
```