UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

LEANDRO PRYTULA

PROJETO DE DIPLOMAÇÃO

SENSOR INTELIGENTE

Porto Alegre
2011

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SENSOR INTELIGENTE

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para Graduação em Engenharia Elétrica.

ORIENTADOR: Prof. Dr. Valner João Brusamarello

Porto Alegre
2011

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

LEANDRO PRYTULA

SENSOR INTELIGENTE

Este projeto foi julgado adequado para fazer jus aos créditos da Disciplina de "Projeto de Diplomação", do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: Prof. Dr. Valner João Brusamarello, UFRGS

Doutor pela Universidade Federal de Santa Catarina - Florianópolis, Brasil

Banca Examinadora:

Prof. Dr. Renato Machado de Brito, UFRGS

Doutor pela Universidade Federal do Rio Grande do Sul - Porto Alegre, Brasil

Prof. MSc. Ivan Müller, UFRGS

Mestre pela Universidade Federal do Rio Grande do Sul - Porto Alegre, Brasil

Porto Alegre, Julho de 2011.

DEDICATÓRIA

Dedico este trabalho a meus pais, Nicolau e Rachel, que estiveram ao meu lado em todos os momentos, nos bons comemorando e nos difíceis prestando auxílio com total atenção.

AGRADECIMENTOS

A Deus por tudo que fez por mim durante toda minha vida, abrindo portas que jamais imaginei que poderiam existir.

A Universidade Federal do Rio Grande do Sul pela disponibilização de estrutura e por todo o conhecimento transferido.

Aos professores do Laboratório de Instrumentação Eletro-Eletrônica por todo o período em que trabalhamos juntos.

Ao professor Valner Brusamarello, pela orientação no desenvolvimento deste e de outros trabalhos.

Aos meus irmãos e amigos pelo companheirismo, auxílio e compreensão.

RESUMO

A combinação de transdutores e microprocessadores visa facilitar a inserção de um dispositivo, chamado de “sensor inteligente”, em uma rede de comunicação digital, diminuindo os esforços de configuração do mesmo. No entanto, a diversidade de protocolos existentes ou a criação de soluções independentes por parte dos fabricantes dificulta a interoperabilidade desses transdutores, tornando-os pouco flexíveis e mais caros. Dentro desse contexto foi criado o padrão IEEE 1451, que padroniza o hardware e o método de comunicação de um transdutor inteligente. Este trabalho apresenta o desenvolvimento de um sensor inteligente baseado no padrão IEEE 1451. O sensor inteligente desenvolvido é composto de duas partes, o nó sensor e o nó de rede. A comunicação entre essas duas partes é realizada por meio de uma interface sem fio utilizando o protocolo *Simple MAC*, que opera baseado no padrão IEEE 802.15.4. As principais características do sensor inteligente desenvolvido é a sua capacidade de auto-configuração e auto-identificação quando conectado a uma rede. Além disso, esse sensor possui capacidade de comunicação com outros nodos e auto-calibração. Os resultados foram validados utilizando-se uma plataforma *Freescale* composta por módulos sensores e um módulo base.

Palavras-chave: Redes de sensores sem fio, IEEE 1451, auto-configuração, auto-identificação, auto-calibração.

ABSTRACT

The combination of transducers and microprocessors facilitate the insertion of a device called a "smart sensor" in a digital communication network and decrease the efforts of proper configuration. However, the diversity of existing protocols or the creation of independent solutions by manufacturers hinders the interoperability of these transducers, making them less flexible and more expensive. Within this context was created IEEE 1451, which standardizes the hardware and communication method of a smart transducer. This paper presents the development of a smart sensor based on the IEEE 1451. The smart sensor developed is composed of two parts, the sensor node and network node. Communication between these two parts is accomplished through a wireless interface using the Simple MAC protocol based on the IEEE 802.15.4. The main features of the smart sensor developed is its ability to self-configuration and self-identification when connected to a network. In addition, this sensor is able to communicate with other nodes and do self-calibration. The results were validated using a Freescale platform consists of sensor modules and a module base.

Keywords: wireless sensor networks, IEEE 1451, self-configuring, self-identification, self-calibration.

SUMÁRIO

1 . INTRODUÇÃO 12
2 . OBJETIVOS 14
3 . MOTIVAÇÃO 15
4 . PADRÃO IEEE 1451 16
4.1 TERMOS UTILIZADOS NO PADRÃO 1451 16
4.1.1 TII - *Transducer Independent Interface* 16
4.1.2 API – *Application Program Interface* 17
4.1.3 TIM – *Transducer Interface Module* 17
4.1.4 STIM - *Smart Transducer Interface Module* 17
4.1.5 TBIM - *Transducer Bus Interface Modules* 17
4.1.6 NCAP - *Network Capable Application Processor* 17
4.1.7 TEDS - *Transducer Electronic Data Sheet* 18
4.2 PADRÃO IEEE1451.0 18
4.3 PADRÃO IEEE1451.1 19
4.4 PADRÃO IEEE1451.2 20
4.5 PADRÃO IEEE1451.3 20
4.6 PADRÃO IEEE1451.4 20
4.7 PADRÃO IEEE1451.5 21
4.8 PADRÃO IEEE P1451.6 22
4.9 PADRÃO IEEE P1451.7 22
5 . ESCOLHA DA PLATAFORMA DE DESENVOLVIMENTO 23
5.1 HARDWARE 23
5.2 AMBIENTE DE DESENVOLVIMENTO 26
6 . ESCRITA DOS TEDS 27
6.1 FORMATAÇÃO DOS TEDS 27
6.2 META TEDS 28
6.3 TEDS DE CANAL DO TRANSDUTOR (TRANSDUCER CHANNEL TEDS) 30
6.4 NOME DE USUÁRIO DO TRANSDUTOR (USER TRANSDUCER NAME TEDS) 34
6.5 TEDS DE CALIBRAÇÃO (CALIBRATION TEDS) 36
6.6 TEDS DE CAMADA FÍSICA (PHY TEDS) 38
6.7 TEDS NO NÓ SENSOR 42
7 . COMUNICAÇÃO ENTRE O NÓ SENSOR E O NÓ DE REDE 43
7.1 PROTOCOLO DE COMUNICAÇÃO SIMPLE MAC 43
7.2 ESTRUTURA DE MENSAGEM 44
7.2.1 MENSAGEM DE COMANDO 44
7.2.2 MENSAGEM DE RESPOSTA 45

7.3 COMANDOS 45
7.3.1 COMANDOS COMUNS AO TIM E AO CANAL DO TRANSDUTOR 45
7.3.2 COMANDOS COM O TRANSDUTOR EM ESPERA (*IDLE*) 47
7.3.3 COMANDOS COM O TRANSDUTOR EM ESTADO DE OPERAÇÃO 47
7.3.4 COMANDOS COM O TRANSDUTOR EM OPERAÇÃO OU EM ESPERA 47
7.3.5 COMANDOS COM O TIM ATIVO 48
8 . ROTINA DE AUTO-CALIBRAÇÃO 49
8.1 MÉTODO DE REGRESSÃO LINEAR 49
8.2 METODOLOGIA DO ALGORITMO DE CALIBRAÇÃO 50
9 . OPERAÇÃO DO SENSOR INTELIGENTE 51
10 . RESULTADOS OBTIDOS 53
11 . CONCLUSÕES 58
12 . TRABALHOS FUTUROS 59
13 . REFERÊNCIAS 60
14 . APÊNDICE A – CÓDIGO UTILIZADO NA SOLICITAÇÃO DAS INFORMAÇÕES DOS SENSORES PELO NÓ DE REDE E NA SINCRONIZAÇÃO DOS RÁDIOS 62
15 . APÊNDICE B – ROTINAS UTILIZADAS NA SOLICITAÇÃO DE TEDS 66
16 . APÊNDICE C – ROTINA DE AUTO-CALIBRAÇÃO DESENVOLVIDA 71

LISTA DE FIGURAS

Figura 1 - Exemplo de monitoração de ambiente por rede sem fio. (fonte:[2]) 12
Figura 2- Modelo proposto pelo IEEE 1451.0 (fonte [10]) 19
Figura 3 - Diagrama em blocos do IEEE1451.5 (fonte [12]) 21
Figura 4 – 1322X NCB (fonte [14]) 24
Figura 5 – 1322X SRB (fonte [15]) 25
Figura 6 – Diagrama em blocos do MC13225 (fonte [16]) 26
Figura 7 – Estrutura do sensor inteligente 27
Figura 8 – Diagrama em blocos do MC1322X SMAC (fonte [17]). 43
Figura 9 – Exemplo de uma curva ajustada por regressão linear 49
Figura 10 – Fluxograma nó sensor 51
Figura 11 – Fluxograma nó de rede 52
Figura 12 – Canais e frequências de operação do IEEE 802.15.4 (fonte [20]) 54
Figura 13 - Leitura e atualização dos TEDS 55
Figura 14 - Conexão dos sensores aos canais A/D 56
Figura 15 - Retas de resposta dos sensores 56
Figura 16 - Sensor inteligente em operação e calibrado 57

LISTA DE TABELAS

Tabela 1 – Formatação de um TEDS genérico, composto por 2 *TLV's*.....28
Tabela 2 – Mensagem de comando.....44
Tabela 3 – Mensagem de resposta.....45
Tabela 4 – Comandos Comuns ao TIM e ao canal do transdutor.....46
Tabela 5 – Comandos com o transdutor em operação.....47
Tabela 6 - Comandos com o transdutor em qualquer estado.....48
Tabela 7 – Comandos com o TIM ativo.....48

1 . INTRODUÇÃO

As redes de sensores sem fio vêm sendo objeto de estudo já faz algum tempo, mas o desenvolvimento dessas redes se tornou mais comum apenas há alguns anos, devido aos avanços nas áreas de microprocessadores, novos materiais de sensoriamento, micro sistemas eletro-mecânicos (MEMS –*Micro Electro-Mecanical Systems*) e de comunicações sem fio. É usual ter num único *chip* vários transdutores, que são controlados pela lógica do circuito integrado, com uma interface de comunicação sem fio [1]. A substituição das tradicionais redes de sensores cabeadas por redes sem fio traz muitas vantagens e permite que não só áreas industriais sejam beneficiadas. Dentre essas pode-se citar:

Ambiental: nessa área as redes de sensores sem fio são usadas para monitoração de condições ambientais, pois em condições de relevo irregulares e vegetação densa a instalação de estruturas de monitoramento é muito desfavorável. As redes sem fio permitem nesses casos, por exemplo, a rápida detecção de pontos de incêndio ou de enchente em uma floresta, monitoração de movimentos de animais ou ainda monitoração de fatores ambientais, como nível de poluição no ar, concentração de pesticidas na água, temperatura, umidade e pressão. A figura 1 mostra uma aplicação de redes sem fio.

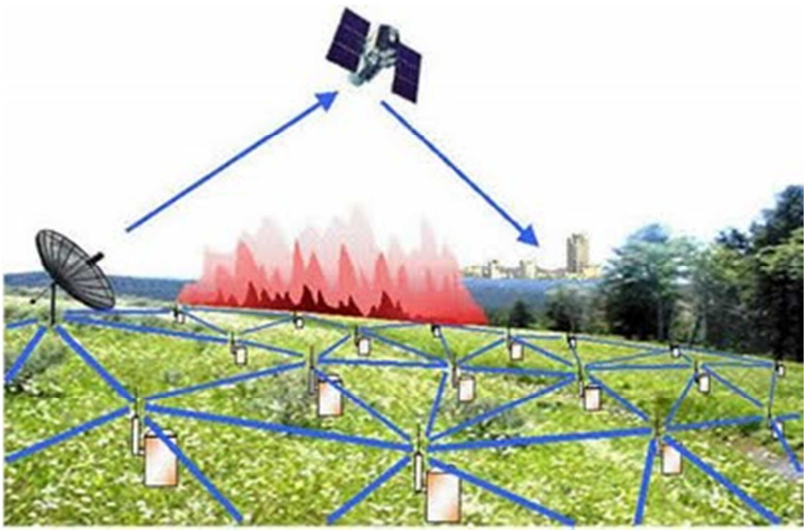

Figura 1 - Exemplo de monitoração de ambiente por rede sem fio. (fonte:[2])

Saúde: nas aplicações relacionadas à saúde as redes de sensores sem fio podem ser utilizadas para rastreamento de médicos, monitoração de movimentos dos pacientes, monitoração de determinadas funções do corpo (batimentos cardíacos ou pressão arterial), além da administração de medicamentos.

Outras áreas: aplicações de controle de tráfego, modelamento e monitoração de estruturas, controle de qualidade de produtos, etc [3].

A ideia principal de redes de sensores sem fio é tirar proveito de dispositivos com baixo consumo de energia, pequenos, baratos, de fácil reposição e manutenção que possam ser usados em tão larga escala que viabilizem aplicações até então imaginadas apenas na ficção, como sensoriamento espacial inteligente (os dispositivos seriam tão pequenos que permaneceriam suspensos no ar, comunicando-se por horas ou dias), auto-identificação, e rastreamento de praticamente qualquer espécie de objeto físico [4]. As informações disponibilizadas por esses sensores poderiam ser utilizadas de forma independente, servindo para atuação pontual, ou em conjunto, podendo atuar em um sistema complexo ou ainda realizando auto-calibração de sensores. No entanto, a auto-identificação e auto-calibração de sensores só é possível se o sensor tiver suporte ao mesmo protocolo de comunicação da rede a que pretende se conectar e se os comandos de acesso às informações do sensor forem conhecidos. Com a grande variedade de protocolos e de fabricantes essa tarefa tornou-se muito complicada, não só em redes sem fio, mas também em redes cabeadas. Devido a esse inconveniente o Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) em parceria com o Instituto Nacional de Padrões e Tecnologia (NIST) (agência governamental não-regulatória da Administração de Tecnologia do Departamento de Comércio dos Estados Unidos) resolveu criar um padrão para facilitar essas operações, o padrão IEEE 1451[5].

O desenvolvimento desse trabalho ocorre em torno dos três pontos apresentados no parágrafo anterior: a auto-identificação, a auto-calibração de sensores sem fio e a padronização dos sensores utilizando o padrão IEEE 1451.

2 . OBJETIVOS

O objetivo principal deste trabalho é o desenvolvimento de um sensor inteligente sem fio baseado no padrão IEEE 1451, mais especificamente nos pontos 1451.0 e 1451.5 relacionados, respectivamente, a definições comuns a todos os pontos e a interface física sem fio, baseada no protocolo *Simple MAC,* com capacidade de auto-calibração. Para que esse objetivo principal seja atingido, objetivos específicos necessitam ser alcançados, a saber:

1) Escrita dos dados eletrônicos do transdutor (*Transducer Electronic Datasheets* – TEDS) do padrão IEEE 1451.0, com ênfase no TEDS de calibração;
2) Escrita dos TEDS do padrão IEEE 1451.5, que descreve a interface física;
3) Inserção dos TEDS na plataforma de hardware escolhida;
4) Elaboração dos algoritmos de leitura do sensor, baseado nos comandos apresentados no padrão 1451.

3 . MOTIVAÇÃO

O padrão IEEE 1451 teve os seus primeiros resultados publicados em 1997. A partir dessa data diversos outros resultados foram sendo publicados e outros foram revisados de forma a tornar o padrão mais flexível e ampliar suas aplicações, tanto que o próprio documento que trata das aplicações sem fio (IEEE 1451.5) foi publicado apenas em 2007. Nesse período, alguns trabalhos baseados no IEEE 1451 foram publicados, dentre estes pode-se citar o apresentado em [6], o qual trata da implementação de um módulo de sensoriamento baseado no IEEE1451.2 e a sua implementação ainda é baseada em redes cabeadas. O sistema proposto não realiza auto-calibração. O trabalho apresentado em [7] trata da implementação de um aplicação web, dinâmica e interativa, para o controle e monitoramento de transdutores inteligentes conectados em rede de acordo com o padrão IEEE 1451, mas os sensores utilizados são cabeados e o sistema não é capaz de realizar auto-calibração. Já o trabalho mencionado em [5] trabalha com o IEEE1451.5 e com o IEEE1451.0 implementando sensores sem fio conectados a um computador que faz o papel de nó de rede, mas o sensor implementado não é capaz de realizar a auto-calibração. Como todos os trabalhos citados foram desenvolvidos em conformidade com alguma das publicações do padrão IEEE 1451 a auto-identificação é algo inerente a esses sensores. A motivação para o desenvolvimento desse trabalho é baseada justamente no fato de nenhum dos trabalhos citados apresentar a característica de auto-calibração. Outra motivação importante encontrada é o fato da grande aceitação e tendência de migração de redes cabeadas para redes sem fio, sendo o padrão IEEE1451 uma das alternativas.

4 . PADRÃO IEEE 1451

A integração de microprocessadores e transdutores tem a função de adicionar inteligência ao transdutor, tornando possível a inserção deste em uma rede de comunicação digital. Essa integração visa tornar mais simples a inserção de transdutores em uma rede, evitando diversos processos de configuração. No entanto, com a grande variedade de redes de campo atualmente disponíveis, cada fabricante de sensores microprocessados tende a escolher um protocolo de rede e trabalhar em torno dele, ou ainda criar um novo protocolo para seus produtos. Esta diversidade de redes de campo e protocolos leva um fabricante a produzir equipamentos pouco flexíveis e, em geral soluções mais caras. A família 1451 de normas procura resolver esse problema propondo um conjunto de hardware padronizado e interfaces de software que funcionam como "*plugs*", possibilitando que diferentes transdutores possam ser conectados a uma mesma rede [8]. Cabe salientar que a família 1451 não é um protocolo de rede, mas uma padronização.

O padrão IEEE1451 é composto por diversos grupos de trabalho, cada um responsável por elaborar um ponto do padrão. Atualmente seis desses grupos já tornaram disponíveis os resultados de seu trabalho, e dois ainda estão trabalhando no desenvolvimento do seu ponto. A seguir serão apresentados alguns termos comuns utilizados e o conteúdo de cada um dos grupos de trabalho do 1451.

4.1 TERMOS UTILIZADOS NO PADRÃO 1451

4.1.1 TII - *Transducer Independent Interface*

A TII é o meio físico de comunicação e o protocolo de transferência de informações entre o *Transducer Interface Module* (TIM) e o *Network Capable Application Processor* (NCAP).

4.1.2 API – *Application Program Interface*

API's são conjuntos de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por programas aplicativos que não querem envolver-se em detalhes do desenvolvimento do software, mas apenas usar seus serviços [9].

4.1.3 TIM – *Transducer Interface Module*

O TIM é o módulo que contém a interface, condicionadores de sinal, conversores analógico-digital e/ou digital-analógico e em muitos casos o transdutor. A composição do TIM pode ser desde um simples sensor ou atuador a uma unidade contendo diversos transdutores (sensores e atuadores) [10].

4.1.4 STIM - *Smart Transducer Interface Module*

O STIM é um TIM quando implementado em conformidade com o padrão IEEE 1451.2. O STIM é um transdutor que tem a sua comunicação com o NCAP baseada em uma interface similar a Serial Peripheral Interface (SPI), com linhas de hardware adicionais para controle de fluxo e tempo, resultando em um total de 10 linhas para a interface [6].

4.1.5 TBIM - *Transducer Bus Interface Modules*

O TBIM é um TIM quando implementado em um arranjo de transdutores distribuídos, definido no padrão IEEE 1451.3.

4.1.6 NCAP - *Network Capable Application Processor*

O NCAP é um conjunto composto por hardware e software que faz a função de gateway

entre o TIM e a rede da aplicação ou host [10].

4.1.7 TEDS - *Transducer Electronic Data Sheet*

Os TEDS são arquivos armazenados no TIM que contêm informações relacionadas ao fabricante, como nome, número de série, tipo de sensor, dados de calibração, etc. Os TEDS permitem que um sensor ou atuador seja automaticamente identificado e descreva a si próprio a uma rede [6].

4.2 PADRÃO IEEE1451.0

O padrão IEEE1451.0 contém o embasamento para todas as outras especificações da família IEEE1451 desenvolvendo um conjunto de funcionalidades comuns para a família de padrões IEEE 1451. Essas funcionalidades são independentes do meio físico de comunicação. O IEEE 1451.0 inclui as funções básicas necessárias para controlar e gerenciar os transdutores inteligentes, protocolos comuns de comunicação, e a formatação dos TEDS. Também são definidos nesse padrão um conjunto de APIs independentes de implementação, contendo métodos de leitura e escrita dos canais do transdutor, leitura e escrita dos TEDS, envio de comandos, configurações, controles e operações ao TIM.

Este padrão não especifica condicionamento de sinal e conversão, o meio físico, ou como os dados TEDS são utilizados em aplicações [10]. Na figura 2 é mostrado um diagrama de um sensor baseado no padrão 1451.

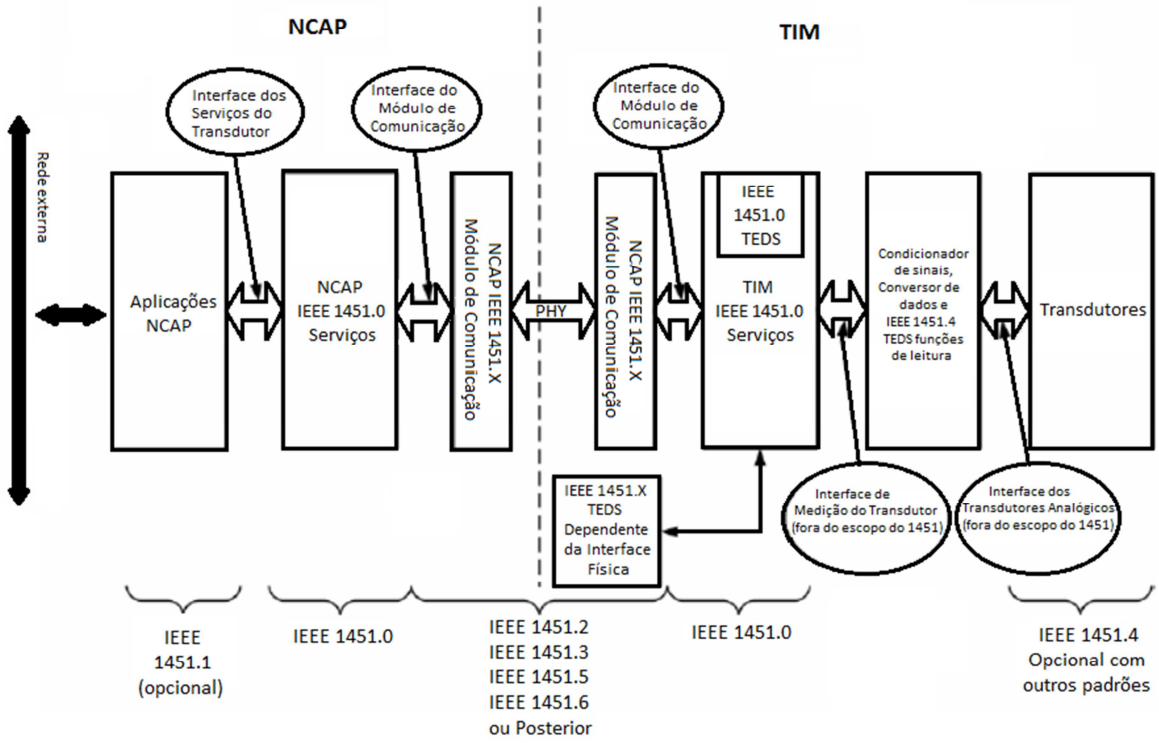

Figura 2- Modelo proposto pelo IEEE 1451.0 (fonte [10])

4.3 PADRÃO IEEE1451.1

O padrão IEEE 1451.1 trata especificamente do NCAP. O NCAP é a ligação entre o transdutor e a rede de comunicação, sendo composto basicamente por dois blocos: o bloco de campo e o bloco de rede [11].

O bloco de campo é responsável por detectar automaticamente, ler, escrever e disparar o transdutor.

O bloco de rede permite que um cliente remoto acesse o transdutor como se ele estivesse realmente no NCAP, independentemente do protocolo de rede ou dos processadores envolvidos. Por meio desse bloco é possível que o cliente remoto adquira informação (por exemplo, amostras digitalizadas), transmita informação (por exemplo, acione um relé), desencadeie ações (por exemplo, reinicie um canal do TIM), ou seja, notificado de eventos (por exemplo, ocorrência de

anormalidades).

Entre esses dois blocos o NCAP ainda pode rodar algoritmos que processam os dados recebidos de ambos os blocos.

4.4 PADRÃO IEEE1451.2

O padrão IEEE 1451.2 define a interface entre os transdutores e o NCAP, além dos TEDS para configurações ponto-a-ponto. Os transdutores estão contidos no “Smart Transducer Interface Module ”(STIM). O padrão original descreve uma camada de comunicação baseada em Serial Peripheral Interface (SPI), com linhas de hardware adicional para controle de fluxo e tempo, resultando em um total de dez linhas para essa interface. Esta norma está sendo revisada para interface com o IEEE 1451.0 e com as populares interfaces seriais: UART e Universal Serial Interface [12].

4.5 PADRÃO IEEE1451.3

Especificação da forma de comunicação para um arranjo de vários transdutores distribuídos, cujas informações precisam ser lidas de forma sincronizada por um barramento, composto de um par de fios, conectado ao NCAP [12]. O STIM nessa norma passa a se chamar TBIM (Transducer Bus Interface Module).

4.6 PADRÃO IEEE1451.4

Especifica como os sinais analógicos ou digitais de um transdutor podem ser disponibilizados ao NCAP por meio de uma mesma interface. Neste padrão também são desenvolvidos TEDS para sensores com pouca memória [12].

4.7 PADRÃO IEEE1451.5

Possui o objetivo de direcionar as aplicações do padrão IEEE1451 em ambiente de rede sem fio (*wireless*).

O padrão IEEE1451.5 define a interface sem fio entre o NCAP e o WTIM, sendo a descrição dessa interface feita por TEDS. Os protocolos WiFi, Bluetooth, ZigBee foram adotados como interfaces wireless padronizadas no IEEE1451.5 [12], mas não há impedimento de se utilizar uma interface proprietária . O NCAP contém um ou mais rádios baseados nos protocolos definidos e pode se comunicar com um ou mais WTIM. O NCAP pode se comunicar com cada WTIM utilizando diferentes protocolos “simultaneamente”. A figura 3 mostra a estrutura do padrão 1451.5.

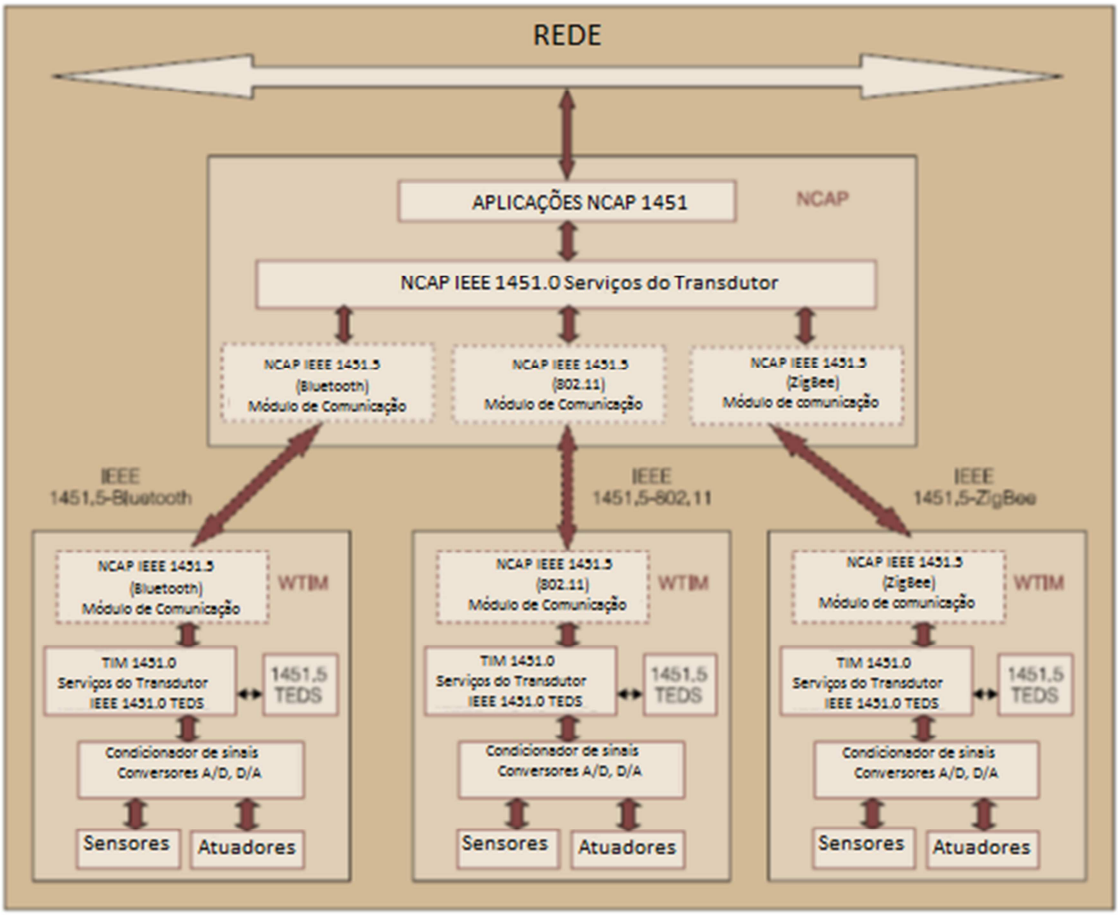

Figura 3 - Diagrama em blocos do IEEE1451.5 (fonte [12])

4.8 PADRÃO IEEE P1451.6

Esse padrão ainda está em desenvolvimento. Sua meta é o uso de uma rede de alta velocidade baseada no sistema CAN-OPEN com diversos módulos contendo transdutores e a definição de uma camada de segurança no modelo de comunicação [12].

4.9 PADRÃO IEEE P1451.7

Esse padrão ainda está em desenvolvimento. Seu objetivo é definir uma interface e um protocolo de comunicação entre os transdutores e sistemas baseados em RFID [12].

Ao analisar os pontos da norma 1451 verifica-se que os que se encaixam ao sensor que será implementado são o 1451.0, pois este serve de base para todos os demais padrões, e o 1451.5, que padroniza uma interface wireless entre o NCAP e o TIM.

5 . ESCOLHA DA PLATAFORMA DE DESENVOLVIMENTO

5.1 HARDWARE

A seleção do hardware para o desenvolvimento do **Sensor Inteligente** foi baseada na necessidade de uma interface sem fio entre o nó de rede e o nó sensor. Por possuir uma interface sem fio e pela disponibilidade imediata foram escolhidos dois diferentes kits de desenvolvimento da Freescale, o 1322X-NCB, utilizado para desenvolver o nó de rede, e o 1322X-SRB, utilizado para desenvolver o nó sensor, ambos baseados no controlador MC13225. As figuras 4 e 5 mostram, respectivamente o kit 1322X – NCB e o kit 1322X –SRB e na figura 6 é mostrada a estrutura do controlador MC13225 . No nó sensor também foram conectados dois sensores de temperatura (modelo LM35 [13]) que serão utilizados como sensor a ser calibrado e como sensor de referência (ver seção 8.2). As principais características desses kits e do controlador são:

1) 1322X-NCB

- Protocolo 802.15.4 2.4 GHz PiP;
- Antena impressa;
- Conector SMA;
- LCD gráfico monocromático;
- Alto falante;
- Joystick, botões e LEDs;
- Interface J-TAG de programação e depuração;
- Interface Nexus de depuração.

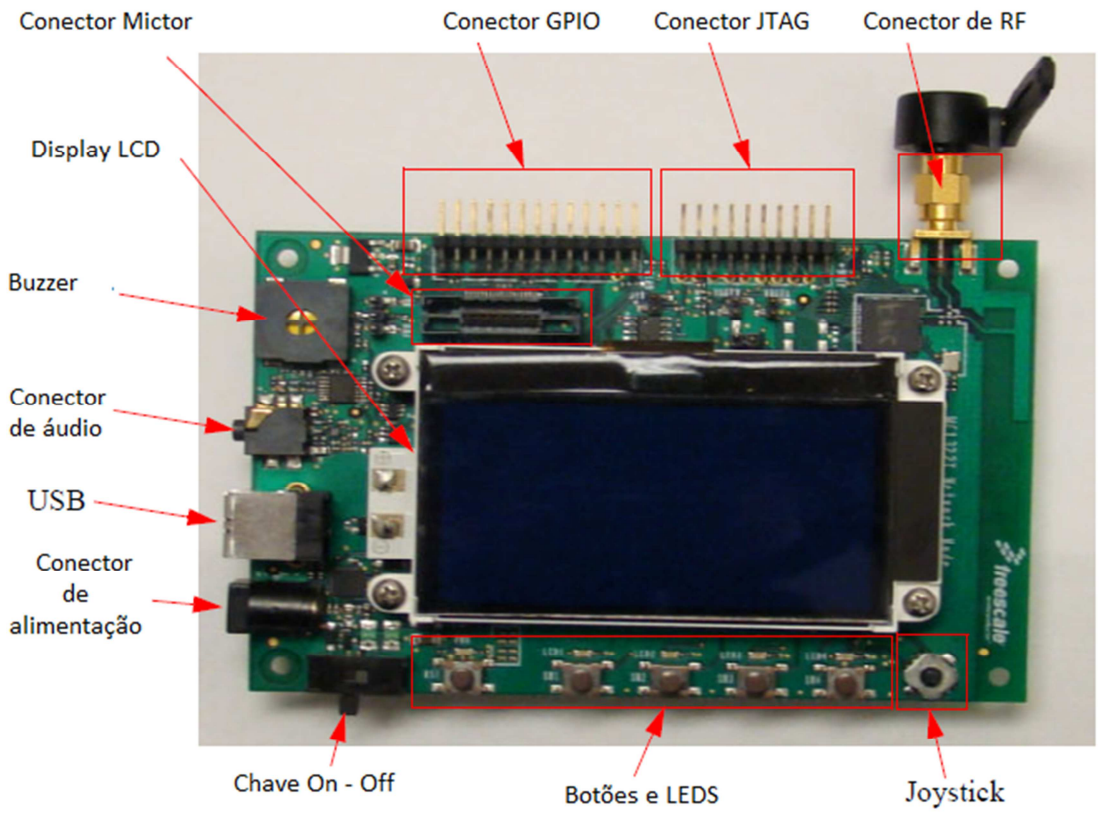

Figura 4 – 1322X NCB (fonte [14])

2) 1322X-SRB

- Protocolo 802.15.4 2.4 GHz PiP;
- Acelerômetro MMA7260Q (três eixos de medição);
- Sensor de pressão MPXV5010G;
- Sensor de temperatura;
- Antena impressa;
- Conector SMA;
- Alto falante;
- Joystick, botões e LEDs;
- Interface J-TAG de programação e depuração;
- Interface Nexus de depuração.

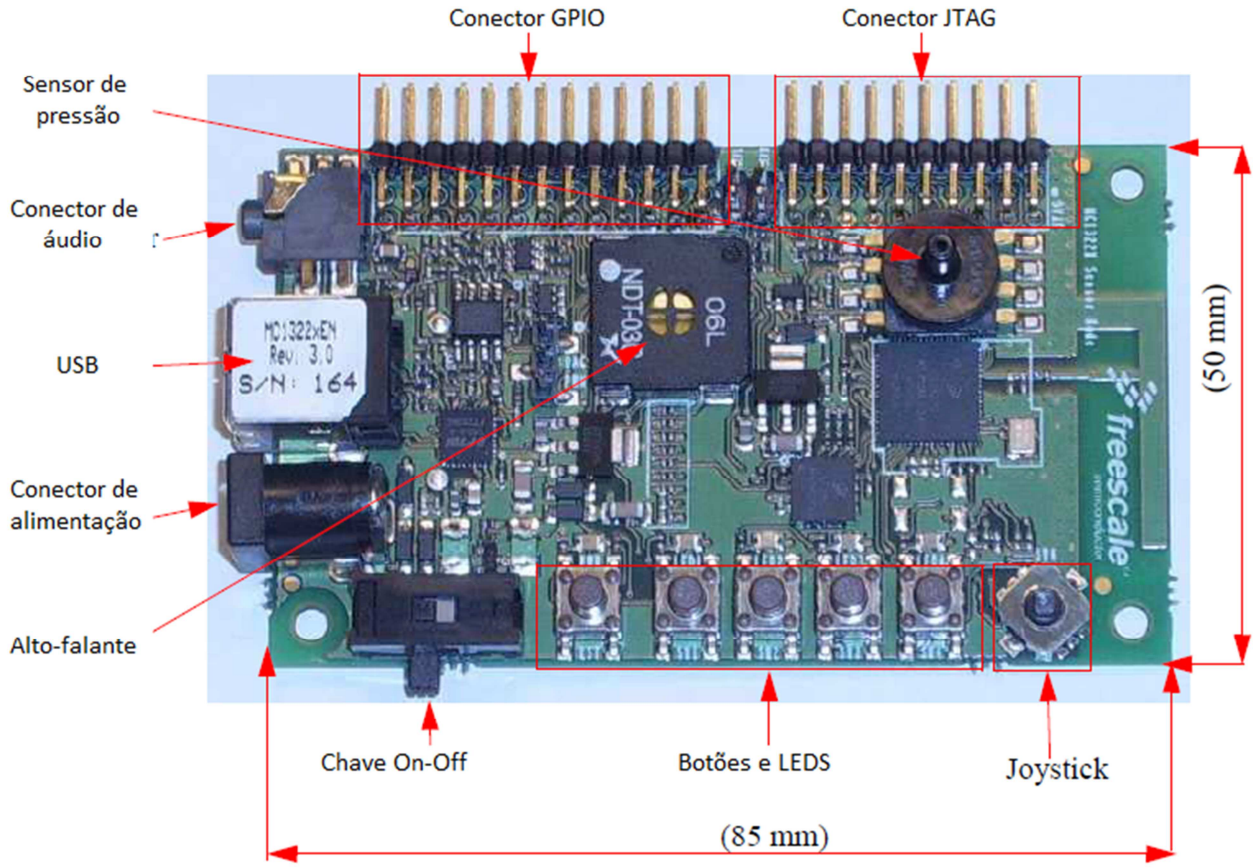

Figura 5 – 1322X SRB (fonte [15])

3) MC13225

- 128KB memória *flash* serial;
- 96KB memória RAM estática;
- 80KB memória ROM;
- Aceleração por hardware para IEEE 802.15.4.
- Módulo dedicado de interface modem/radio 802.15.4
- Interface dedicada NVM SPI para gerenciamento da memória *flash*;
- Dois módulos UART com capacidade de 2Mbps com suporte a CTS/RTS;
- Porta SPI com operação *master* e *slave* programáveis;
- Dois conversores analógico-digital de 12 bits compartilhados em 8 canais de entrada;
- Quatro *timers* independentes de 16 bits com função de PWM. Podem ser combinados em cascata para operação em 64 bits;
- *Interface Inter-integrated circuit* (I2C);

- *Synchronous Paralell Interface* (SPI);
- Capacidade acima de 64 I/O (compartilhada com periféricos);
- Porta JTAG de depuração;
- Interface Nexus de depuração.

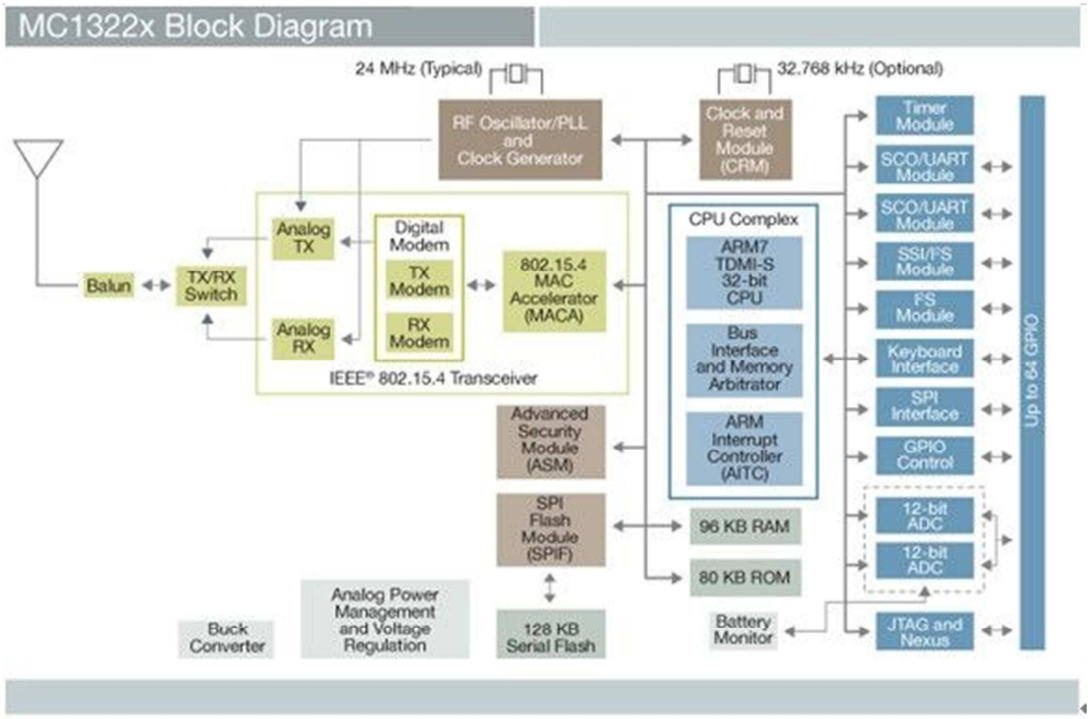

Figura 6 – Diagrama em blocos do MC13225 (fonte [16])

5.2 AMBIENTE DE DESENVOLVIMENTO

O software escolhido para desenvolvimento do *firmware* do nó sensor e do nó de rede foi o IAR Embedded Workbench versão 5.2, produzido pela IAR Systems. Esse software possui ferramentas de desenvolvimento para a construção e depuração de aplicativos utilizando *assembler,* C e C++. Seu ambiente de desenvolvimento permite editar, gerenciar, compilar e depurar o *firmware* desenvolvido. A configuração inicial da plataforma de hardware (inicialização de IO's e periféricos) foi feita pelo software Beekit, fornecido pela Freescale.

Para gravar o *firmware* na plataforma de hardware foi utilizado o IAR J-Link, também fornecido pela IAR Systems, uma ferramenta JTAG de depuração de hardware.

6 . ESCRITA DOS TEDS

A escrita dos TEDS visa, como citado na seção 4.1.7, descrever o sensor como um todo e identificá-lo a rede.

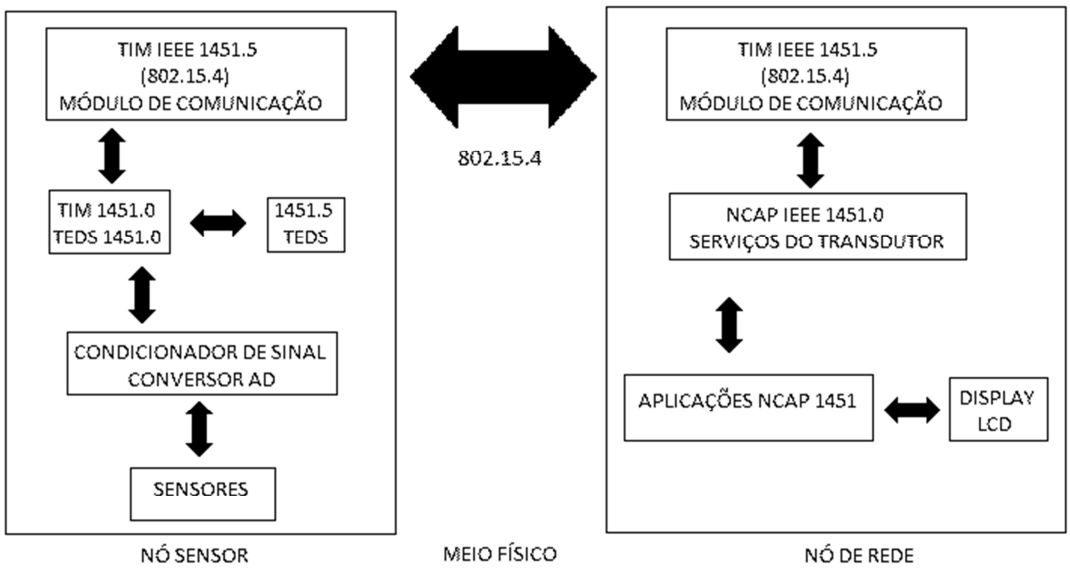

Figura 7 – Estrutura do sensor inteligente

Para que a descrição seja completa e adaptável a qualquer tipo de transdutor, várias estruturas de TEDS foram criadas, algumas são obrigatórias e outras opcionais. Das estruturas criadas quatro pertencem ao padrão IEEE1451.0 e uma ao IEEE1451.5. A figura 7 mostra a estrutura do sensor inteligente desenvolvido e pode-se observar nessa figura o papel dos TEDS no sensor. A seguir serão apresentados a formatação dos TEDS e a composição de cada um deles.

6.1 FORMATAÇÃO DOS TEDS

Os TEDS são compostos por três tipos de campo, a saber:

1) Campo Comprimento do TEDS (TEDS *Lenght*) – formado por quatro octetos (conjuntos de 8 bits). Representa o número total de octetos contido no bloco de dados, acrescidos ainda os dois octetos do *checksum*;
2) Campo Bloco de Dados (*Data Block*) – o número de octetos que o compõe é variável. É o campo que contém a informação do TEDS, sendo composto por diversas estruturas

"tipo/comprimento/valor" (*TLV – Type/Lenght/Value*). O "tipo" é composto por um octeto que identifica a estrutura TLV. O "comprimento" especifica o número de octetos do campo "valor", que por sua vez representa o dado atual;

3) Campo *Checksum* – formado por dois octetos. Esse campo é formado pelo complemento de um da soma de todos os octetos precedentes a esse campo. A soma dos octetos deve ser mantida em um número de dezesseis bits, sendo descartados os bits mais significativos caso o valor da soma ultrapasse o número de bits especificado. O controle de dados é feito pelo cálculo do *checksum*.

$$checksum = 0\text{xFFFF} - \sum_{i=1}^{Total\ de\ octetos\ -\ 2} \text{Octetos TEDS (i)}$$

A tabela 1 mostra a formatação de um TEDS genérico.

Tabela 1 – Formatação de um TEDS genérico, composto por 2 *TLV's*

CAMPO	ESTRUTURA
COMPRIMENTO	4 OCTETOS
BLOCO DE DADOS	TLV 1
	TLV 2
CHECKSUM	2 OCTETOS

6.2 META TEDS

É um TEDS obrigatório, pertence ao padrão IEEE 1451.0 [10]. Tem a função de tornar disponível toda a informação necessária para acessar qualquer canal de transdutor, fornecer o número de transdutores que compõem o sensor, além de fornecer informações comuns a todos canais do sensor.

1) TLV Identificador (TEDSID – TEDS *Identification Header*) – estrutura obrigatória, contém a identificação do TEDS. Composição (valores em hexadecimal):

   03 – Identificador do Meta TEDS

   04 – Comprimento do "valor" do TLV

00 – Número do ponto da família 1451 a que pertence

01 – Código de acesso desse TEDS

01 – Versão do TEDS

01 – Número de octetos do campo “Comprimento”

2) TLV Identificador Exclusivo (UUID – *Globally Unique Identifier*) – estrutura obrigatória, identifica o TIM com um número exclusivo, o qual é uma composição da data de fabricação (dia, mês e ano), da quantidade de unidades produzidas no dia, do número sequencial de produção do TIM no dia e das coordenadas geográficas do fabricante. Composição:

   04 – Identificador do UUID

   0A – Comprimento do “valor” do TLV

   0D 34 88 B3 EC C1 F6 C1 AD B1 – Identificador do TIM (Produzido em 20/04/2011, sendo apenas uma unidade produzida no referido dia, nas coordenadas 30°02’15” Sul, 51°13’13” Oeste (Coordenadas de Porto Alegre))

3) TLV Tempo fora de operação (OHoldOff – *Operational time-out*) – estrutura obrigatória, contém o intervalo de tempo máximo, em segundos, entre o envio de um comando e a resposta. Caso seja ultrapassado é interpretado que houve uma falha na operação. Ainda pode ser o tempo pelo qual o dispositivo deve estar pronto para aceitar um próximo comando quando nenhuma resposta é necessária para o comando atual. Composição:

   0A – identificador do OholdOff

   04 – Comprimento do “valor”

   3F 80 00 00 – Valor de um segundo, expresso em ponto flutuante com precisão simples

4) TLV Tempo para auto-teste (TesteTime – *Transducer module self-test time requirement*) – estrutura obrigatória, contém o tempo necessário para auto-teste, em segundos. Se não existe auto-teste deve conter o valor zero. Composição:

   0C – Identificador do TestTime

   04 – Comprimento do “valor”

00 00 00 00 – Valor do auto-teste, em ponto flutuante simples. Indica que não há auto-teste

5) Número de canais de transdutores implementados (MaxChan – *Number of Implemented Transducer Channels*) – estrutura obrigatória, contém o número de transdutores do sensor. Composição;

   0D – Identificador do MaxChan

   02 – Comprimento do “valor”

   00 02 – Número de canais implementados

6) Campo Comprimento (*Lenght*) – Número Total de octetos dos TLV’s acrescidos do *Checksum*. Composição:

   00 00 00 24 – Os TLV’s contém 34 octetos acrescido dos 2 octetos do *checksum*

7) Campo *Checksum* – Complemento de um da soma dos octetos anteriores. Composição:

   02 CC – Complemento de um

6.3 TEDS DE CANAL DO TRANSDUTOR (TRANSDUCER CHANNEL TEDS)

Esse TEDS faz parte do padrão IEEE 1451.0 [10]. Descreve o transdutor que está conectado ao canal do nó sensor. Como serão implementados dois canais serão necessários dois TEDS de canal, mas como serão dois sensores de temperatura as estruturas serão iguais.

1) TLV Identificador (TEDSID – TEDS *Identification Header*) – estrutura obrigatória, contém a identificação do TEDS. Composição (valores em hexadecimal):

   03 – Identificador do *Transducer Channel* TEDS

   04 – Comprimento do “valor” do TLV

   00 – Número do ponto da família 1451 a que pertence

   03 – Código de acesso desse TEDS

   01 – Versão do TEDS

   01 – Número de octetos do campo “Comprimento”

2) TLV chave de calibração (CalKey – *Calibration Key*) – estrutura obrigatória, contém a informação de onde será feita a calibração do sensor, se será fornecida pelo TIM ou algum outro tipo de calibração. Composição:

   0A – Identificado do CalKey

   01 – Comprimento do “valor”

   01 – Calibração fornecida pelo “Calibration TEDS” e executada no NCAP (CAL_SUPPLIED)

3) TLV Canal do transdutor (ChanType – *Transducer Channel type key*) – estrutura obrigatória, indica se o transdutor é um sensor ou um atuador. Composição:

   0B – Identificador do ChanType

   01 – Comprimento do “valor” do TLV

   00 – Indica que o transdutor é um sensor

4) TLV unidades físicas (PhyUnits – *Physical Units*) – estrutura obrigatória, indica as unidades físicas da informação do sensor. A saída do sensor é em Kelvin. Composição:

   0C – Identificação do PhyUnits

   0C – Comprimento do “valor”

   32 – subcampo Physical Units Interpretation Enumeration

   01 – Comprimento do “valor” do subcampo

   00 – Sistema utilizará unidades físicas do SI

   39 – Unidade de medida Kelvin

   01 – Comprimento do campo do expoente da unidade metros

   82 – expoente da unidade Kelvin igual a 1

5) TLV Limite inferior de operação (LowLimit – *Design Operationa lower range limit*) – estrutura obrigatória, informa o limite inferior de operação do sensor, no caso 275 Kelvin. Composição:

   0D – Identificador do TLV LowLimit

04 – Comprimento do “valor”

43 89 80 00 – 275 Kelvin em ponto flutuante simples

6) TLV Limite Superior de operação (HiLimit – *Design Operationa upper range limit*) – estrutura obrigatória, informa o limite superior de operação do sensor, no caso 383 Kelvin. Composição:

   0E – Identificador do TLV HiLimit

   04 – Comprimento do “valor”

   43 BF 80 00 – 383 Kelvin em ponto flutuante simples

7) TLV Incerteza (OError – *Uncertainty under worst-case conditions*) – estrutura obrigatória, informa a maior incerteza possível. Composição

   0F – Identificador do TLV OError

   04 – Comprimento do “valor”

   40 00 00 00 – equivale a 2 Kelvin em ponto flutuante simples

8) TLV Auto-teste (SelfTest – *Self-Test key*) – estrutura obrigatória, informa se o sensor possui auto-teste. Composição;

   10 – Identificador do TLV SelfTest

   01 – Comprimento do “valor”

   00 – Sem auto-teste

9) TLV Definições de amostragem (Sample – *Sample definition*) – estrutura obrigatória, informa dados referentes à amostragem do sinal do transdutor. Composição:

   12 – Identificador do TLV Sample

   09 – Comprimento do “valor”

   28 – subcampo Data Model

   01 – Comprimento do “valor” do Data model

   00 – Indica que a informação de leitura do canal será enviada ao nó de rede em forma de contagens do canal AD

29 – subcampo Data model lenght

01 – Comprimento do campo “valor” do subcampo *Data Model lenght*

02 – Indica 2 bytes

2A – Subcampo *Model significant bits*

01 – Comprimento do subcampo *model significant bits*

0C – Indica que o AD utilizado é de 12 bits

10) TLV Taxa de atualização do canal do transdutor (UpdateT – *Transducer channel update time*) – estrutura obrigatória, informa a taxa de atualização do sinal do transdutor. Operando livremente o valor deve ser zero. Composição:

14 – Identificador do TLV UpdateT

04 – Comprimento do “valor”

00 00 00 00 – Operando livremente, zero em ponto flutuante simples

11) TLV Taxa de leitura após comando (RSetupT – *Transducer channel read setup time*) – estrutura obrigatória, informa quanto tempo após um sinal de trigger o canal é lido. Operando livremente o valor deve ser zero. Composição:

16 – Identificador do TLV RSetupT

04 – Comprimento do “valor”

00 00 00 00 – Operando livremente, zero em ponto flutuante simples

12) TLV Tempo de amostragem (SPeriod – *Transducer channel sampling period*) – estrutura obrigatória, informa o tempo que o sensor demora para realizar a amostragem, limitado basicamente pelo tempo de conversão do AD. Quando o valor não é relevante o valor deve ser zero. Composição:

17 – Identificador do TLV Speriod

04 – Comprimento do “valor”

00 00 00 00 – Valor não relevante, zero em ponto flutuante simples

13) TLV Taxa de energização (WarmUpT – *Transducer channel warm-up time*) – estrutura

obrigatória, informa o tempo que o sensor demora para estabilizar sua operação ao ser energizado. Composição:

18 – Identificador do TLV WarmUpT

04 – Comprimento do “valor”

40 A0 00 00 – Demora 5 segundos para estabilizar

14) TLV Atraso de leitura do canal do transdutor (RDelayT – *Transducer channel read delay time*) – estrutura obrigatória, informa o tempo que o sensor demora entre o comando de leitura e o envio do frame de dados. Composição:

19 – Identificador do TLV RDelayT

04 – Comprimento do “valor”

3D CC CC CD – Demora 100 mili segundos para enviar (definido pelo *firmware*)

15) TLV Atributos de amostragem (Sampling – *Sampling atribute*) – estrutura obrigatória, define alguns atributos da amostragem. Composição

1F – Identificador do TLV Sampling

03 – Comprimento do “valor”

30 – subcampo *sampling mode capability*

01 – Comprimento do *subcampo sampling mode capability*

02 – Transdutor opera livremente, sem pré trigger

16) Campo Comprimento (*Lenght*) – Número Total de octetos dos TLV’s acrescidos do *Checksum*. Composição:

00 00 00 59 – Os TLV’s contém 87 octetos acrescido dos 2 octetos do *checksum*

17) Campo *Checksum* – Complemento de um da soma dos octetos anteriores. Composição:

F6 14 – Complemento de um

6.4 NOME DE USUÁRIO DO TRANSDUTOR (USER TRANSDUCER NAME TEDS)

É o último TEDS obrigatório do padrão IEEE 1451.0 [10]. Contém informações referentes

ao nome do transdutor, como ele será conhecido no restante da rede. As informações contidas nesse TEDS podem ter o formato que o fabricante do transdutor decidir, ou baseado em estruturas de texto definidos pelo IEEE 1451.0. No desenvolvimento desse sensor ficou definido que ele seria composto pelo código ASCII dos caracteres do nome dado ao sensor.

1) TLV Identificador (TEDSID – TEDS *Identification Header*) – estrutura obrigatória, contém a identificação do TEDS. Composição (valores em hexadecimal):

   03 – Identificador do *User Transducer Name* TEDS

   04 – Comprimento do "valor" do TLV

   00 – Número do ponto da família 1451 a que pertence

   0C – Código de acesso desse TEDS

   01 – Versão do TEDS

   01 – Número de octetos do campo "Comprimento"

2) TLV formato (Format – *Format*) – estrutura obrigatória, define o formato do campo de dados.Composição:

   04 – Identificador do TLV *Format*

   01 – Comprimento do "valor"

   00 – Formato definido pelo usuário

3) Nome do sensor. Composto pelo código ASCII dos caracteres do nome do sensor:

   54 43 43 20 50 52 59 54 55 4C 41 – Código ASCII dos caracteres "TCC PRYTULA"

4) Campo Comprimento (*Lenght*) – Número Total de octetos dos TLV's acrescidos do *Checksum*. Composição:

   00 00 00 16 – Os TLV's contém 19 octetos acrescido dos 2 octetos do *checksum*

5) Campo *Checksum* – Complemento de um da soma dos octetos anteriores. Composição:

   FC A6 – Complemento de um

6.5 TEDS DE CALIBRAÇÃO (CALIBRATION TEDS)

Esse TEDS é opcional para a construção de um sensor no padrão IEEE 1451.0 [10], mas sua escrita para o objetivo de auto-calibração é de extrema importância. Nesse TEDS estão campos que informam o coeficiente linear e o coeficiente angular da curva de calibração do sensor. Esses campos serão alterados durante a rotina de auto-calibração para o sensor que será calibrado, chamado também por conveniência de sensor operacional. Portanto esse TEDS é a configuração inicial do sensor que será calibrado e o TEDS definitivo do sensor de referência.

1) TLV Identificador (TEDSID – TEDS *Identification Header*) – estrutura obrigatória, contém a identificação do TEDS. Composição (valores em hexadecimal):

   03 – Identificador do *Calibration* TEDS

   04 – Comprimento do “valor” do TLV

   00 – Número do ponto da família 1451 a que pertence

   05 – Código de acesso desse TEDS

   01 – Versão do TEDS

   01 – Número de octetos do campo “Comprimento”

2) TLV Última data de calibração (LstCalDt – *Last Calibration Date*) – estrutura opcional, contém a data da última calibração do sensor. Composição:

   0A – Identificador do TLV *LstCalDT*

   08 – Comprimento do “valor”

   4D A2 44 80 00 00 00 00 – Calibração realizada dia 20 abril 2011, às 0h.

3) TLV Intervalo de calibração (CalInrvl – *Calibration Interval*) – estrutura opcional, contém a data da última calibração do sensor. Composição:

   0B – Identificador do TLV CalInrvl

   08 – Comprimento do “valor”

   27 8D 00 00 00 00 00 00 – Calibração foi realizada ao longo de um mês.

4) TLV Conversão de unidades do SI (SIConvrt – *SI units Conversion*) – estrutura opcional, contém dados de conversão de unidades. Composição:

   0C – Identificador do TLV SIConvrt

   0C – Comprimento do "valor"

   1E – Identificador subcampo *Slope*

   04 – Comprimento do "valor" do subcampo *Slope*

   3F 8D 00 00 – 1 em ponto flutuante simples.

   1F – Identificador subcampo *Intercept*

   04 – Comprimento do "valor" do subcampo *Intercept*

   00 00 00 00 – 0 em ponto flutuante simples.

   43 88 80 00 – Coeficiente linear (*Intercept*) 273 Kelvin em ponto flutuante simples.

5) TLV Método de Conversão Linear (LinOnly – *Linear Conversion Method*) – estrutura opcional, contém os dados da curva de calibração, o coeficiente angular e o coeficiente linear. **Esse TLV será alterado no sensor que será calibrado durante a rotina de auto-calibração.** Composição:

   14 – Identificador do TLV LinOnly

   0A – Comprimento do "valor"

   33 – Identificador subcampo *Set of Coeficientes*

   08 – Comprimento do "valor" do subcampo *Set of Coeficientes*

   3D A3 FF EF - Coeficiente angular (*Slope*) 0,080078 em ponto flutuante simples.

   43 89 80 00 – Coeficiente linear (*Intercept*) 275 em ponto flutuante simples.

6) Campo Comprimento (*Lenght*) – Número Total de octetos dos TLV's acrescidos do *Checksum*. Composição:

   00 00 00 36 – Os TLV's contém 52 octetos acrescido dos 2 octetos do *checksum*

7) Campo *Checksum* – Complemento de um da soma dos octetos anteriores. Composição:

   F7 93 – Complemento de um

6.6 TEDS DE CAMADA FÍSICA (PHY TEDS)

Esse TEDS é obrigatório e está em conforme com o padrão IEEE 1451.5 [17]. O PHY TEDS descreve a estrutura física de comunicação entre o nó sensor e o nó de rede.

1) TLV Identificador (TEDSID – TEDS Identification Header) – estrutura obrigatória, contém a identificação do TEDS. Composição (valores em hexadecimal):

   03 – Identificador do User Transducer Name TEDS

   04 – Comprimento do “valor” do TLV

   05 – Número do ponto da família 1451 a que pertence

   13 – Código de acesso desse TEDS

   01 – Versão do TEDS

   01 – Número de octetos do campo “Comprimento”

2) TLV rádio (Radio – *Radio type*) – estrutura obrigatória, contém a identificação do rádio. Composição:

   0A – Identificador do Radio

   01 – Comprimento do “valor” do TLV

   FF – protocolo específico do fabricante (ver 7.1)

3) TLV máxima transferência de dados (MaxBPS – *Maximum data throughput*) – estrutura obrigatória, contém a taxa máxima de transferência de dados do rádio. Composição:

   0B – Identificador do MaxBPS

   04 – Comprimento do “valor” do TLV

   00 03 D0 90 – 250 Kbps em decimal

4) TLV número máximo de dispositivos conectados (MaxCDev – *Maximum number of connected devices*) – estrutura obrigatória, contém o número máximo de dispositivos que podem operar simultaneamente com o nó sensor. Composição:

   0C – Identificador do MaxCDev

02 – Comprimento do “valor” do TLV

00 01 – no máximo um dispositivo pode ser conectado

5) TLV número máximo de dispositivos registrados (MaxRDEV – *Maximum number of registered devices*) – estrutura obrigatória, contém o número máximo de dispositivos que podem simultaneamente serem registrados no rádio. Composição:

   0D – Identificador do MaxRDev

   02 – Comprimento do “valor” do TLV

   00 01 – Máximo um dispositivo registrado

6) TLV Suporte à criptografia (Encrypt - *Encryption support*) – estrutura obrigatória, contém o tipo de criptografia suportada pelo rádio. Composição:

   0E – Identificador do Encrypt

   02 – Comprimento do “valor” do TLV

   00 00 – Não foi utilizada criptografia

7) TLV Autenticação (Authent - *Authentication*) – estrutura obrigatória, informa se é suportada autenticação MAC pelo rádio. Composição:

   0F – Identificador do Authent

   01 – Comprimento do “valor” do TLV

   00 – Autenticação MAC não utilizada.

8) TLV Comprimento mínimo da chave (MinKeyL - *Minimum Key Lenght*) – estrutura obrigatória, contém comprimento mínimo da chave de criptografia suportada pelo rádio. Composição:

   10 – Identificador do MinKeyL

   02 – Comprimento do “valor” do TLV

   00 00 – Criptografia não utilizada

9) TLV Comprimento máximo da chave (MaxKeyL - *Maximum Key Lenght*) – estrutura obrigatória, contém comprimento máximo da chave de criptografia suportada pelo rádio.

Composição:

11 – Identificador do MaxKeyL

02 – Comprimento do “valor” do TLV

00 00 – Criptografia não utilizada

10) TLV Tamanho máximo SDU (MaxSDU - *Maximum SDU Size*) – estrutura obrigatória, contém o tamanho máximo da unidade de serviço de dados (SDU) em bytes que pode ser utilizado na transferência de dados entre dois dispositivos. Composição:

    12 – Identificador do MaxSDU

    02 – Comprimento do “valor” do TLV

    00 64 – 100 bytes de transmissão

11) TLV Latência mínima de acesso (MinALat - *Minimum Acess Latency*) – estrutura obrigatória, contém o tempo necessário para iniciar a primeira transmissão a um dispositivo desconectado. Composição:

    13 – Identificador do MinALat

    04 – Comprimento do “valor” do TLV

    05 F5 E1 00 – 100 mili segundos para iniciar a transmissão

12) TLV Latência mínima de transmissão (MinTLat - *Minimum Trasnmit Latency*) – estrutura obrigatória, contém o tempo necessário para iniciar a transmissão a um dispositivo desconectado. Composição:

    14 – Identificador do MinTLat

    04 – Comprimento do “valor” do TLV

    00 01 D4 C0 – 120 micro segundos para transmitir um pacote de dados.

13) TLV Número máximo de transações simultâneas (MaxXact - *Maximum Number of Simultaneous Transactions*) – estrutura obrigatória, contém o número máximo de transações simultâneas que podem ficar pendentes no API do TIM. Composição:

    15 – Identificador do MaxXact

04 – Comprimento do “valor” do TLV

15 02 00 01 – 1 transação simultânea.

14) TLV Alimentado por bateria (Battery - *Battery Powered*) – estrutura obrigatória, contém a informação sobre a alimentação do TIM. Composição:

16 – Identificador do Battery

01 – Comprimento do “valor” do TLV

00 – Não alimentado por bateria.

15) TLV Versão do radio (RadioVer - *Radio Version*) – estrutura obrigatória, contém a informação sobre a versão do radio. Composição:

17 – Identificador do RadioVer

02 – Comprimento do “valor” do TLV

00 00 – Versão desconhecida.

16) TLV Tentativas máximas antes de desconectar (MaxRetry - *Maximum Retries Befor Disconect*) – estrutura obrigatória, contém o número máximo de tentativas que o rádio executa antes de realizar uma operação de desconexão. Composição:

18 – Identificador do MaxRetry

04 – Comprimento do “valor” do TLV

00 00 00 00 – Nunca desconecta.

8) Campo Comprimento (*Lenght*) – Número Total de octetos dos TLV’s acrescidos do *Checksum*. Composição:

00 00 00 42 – Os TLV’s contém 66 octetos acrescido dos 2 octetos do *checksum*

9) Campo *Checksum* – Complemento de um da soma dos octetos anteriores. Composição:

F6 EB – Complemento de um

6.7 TEDS NO NÓ SENSOR

A partir da apresentação dos TEDS nesse capítulo, nota-se que eles são estruturas que contém diversas informações, muitas das quais não devem ser alteradas. Tendo essa concepção em mente decidiu-se que todos os TEDS seriam inicializados no nó sensor cada vez que este for energizado. Os campos que devem ser fixos serão inicializados com valores fixos e os campos que podem ser alterados serão inicializados com valores armazenados na memória *flash* do microcontrolador.

Para exemplificar a aplicação dessa definição pode-se verificar o TEDS de calibração do sensor a ser calibrado. No TLV Método de Conversão Linear tem-se octetos que representam o coeficiente angular e o coeficiente linear da curva de calibração. Todos os TLV's do TEDS serão inicializados com valores fixos a exceção do TLV citado, o qual buscará da memória *flash* o valor obtido durante o processo de auto-calibração. Se for realizado um novo procedimento de calibração os novos coeficientes linear e angular serão armazenados na memória *flash* e em uma nova inicialização o TLV buscará esses valores atualizados.

7 . COMUNICAÇÃO ENTRE O NÓ SENSOR E O NÓ DE REDE

A comunicação entre o nó sensor e o nó de rede é baseada em três fundamentos: um protocolo de comunicação, em uma estrutura de mensagem e em comando padronizados. O protocolo de comunicação utilizado foi o *Simple MAC* (SMAC), desenvolvido pela Freescale. Já os comandos e a estrutura de mensagem estão definidos pelo IEEE1451.0 e são encapsulados no *Simple MAC*.

7.1 PROTOCOLO DE COMUNICAÇÃO *SIMPLE MAC*

O SMAC é uma pilha de comunicação baseado em linguagem C desenvolvido para aplicações que utilizem *transceivers* 802.15.4 da Freescale [18]. Seu código fonte é aberto e permite o desenvolvimento de rotinas para adaptá-lo as funções desejadas em uma rede sem fio. A camada física (PHY) do SMAC é baseada no padrão IEEE 802.15.4 sendo ainda utilizada a camada MAC de forma simplificada. A figura 8 mostra o diagrama em blocos do MC1322X SMAC. Cabe ressaltar que não foram utilizadas as estruturas de segurança nem a OTAP (*Over The Air Programmer*).

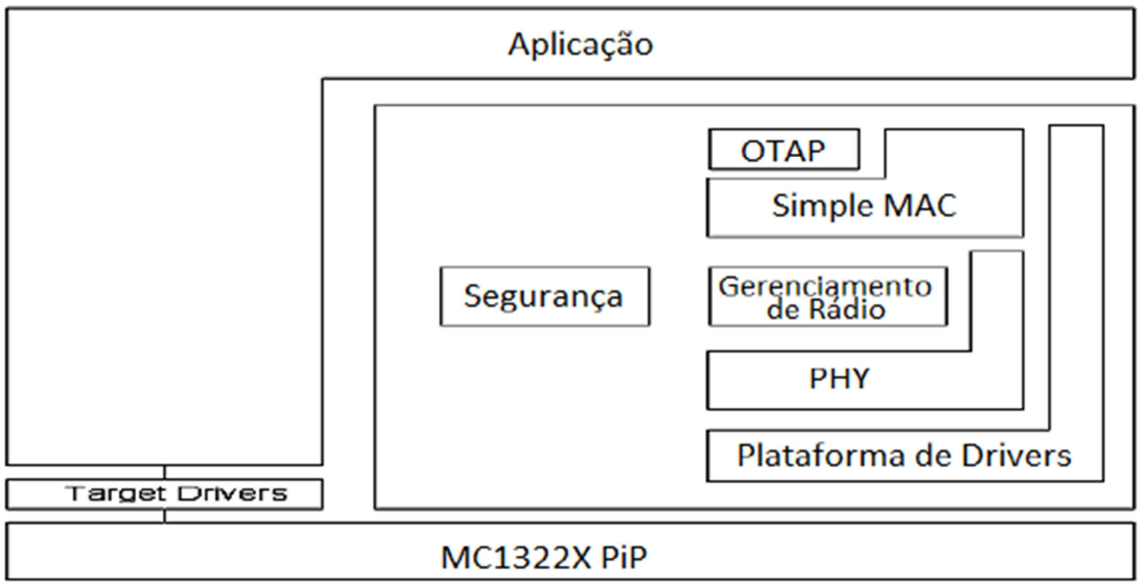

Figura 8 – Diagrama em blocos do MC1322X SMAC (fonte [17]).

7.2 ESTRUTURA DE MENSAGEM

As mensagens trocadas entre o nó sensor e o nó de rede estão regidas por três tipos de estrutura [10]: estrutura de mensagem de comando, estrutura de mensagem de resposta e estrutura de mensagem iniciada pelo TIM (nó sensor). Cada campo da estrutura de mensagem é formada por um octeto e o tamanho da estrutura é variável conforme a informação da mensagem, sendo o tamanho total da mensagem limitado pelo protocolo de comunicação. A seguir são mostradas as estruturas de mensagem de comando e de resposta. A estrutura de mensagem iniciada pelo TIM não foi utilizada no desenvolvimento do sensor inteligente, por isso não será mostrada.

7.2.1 MENSAGEM DE COMANDO

A estrutura da mensagem de comando é utilizada quando o nó de rede solicita alguma informação ou envia alguma ordem ao nó de rede. A composição dessa mensagem é mostrada na tabela 2. Quanto ao conteúdo:

1) O número do canal do transdutor é um endereço de 16 bits que contém o destino da mensagem. O bit 16 desse número indica se a mensagem é destinada ao TIM (nó sensor) como um todo (bit 16 é zero) ou a um canal específico (bit 16 é um).

2) O comprimento é composto pelo número de octetos dependentes do comando.

3) Os octetos de comando serão vistos posteriormente, em 7.3.

Tabela 2 – Mensagem de comando

Octeto	Conteúdo
1	Número do canal do transdutor (octeto mais significativo)
2	Número do canal do transdutor (octeto menos significativo)
3	Classe de comando
4	Função do comando
5	Comprimento (octeto mais significativo)
6	Comprimento (octeto menos significativo)
7	Octeto dependente do comando

8	Octeto dependente do comando

7.2.2 MENSAGEM DE RESPOSTA

A estrutura da mensagem de resposta é utilizada, como o próprio nome sugere, em resposta a uma mensagem de comando. A composição dessa mensagem é mostrada na tabela 3. Quanto ao conteúdo:

1) Se o *flag* falha/sucesso possui o valor zero o comando enviado falhou, necessitando ser investigado pelo sistema o porquê. Se o *flag* for diferente de zero o comando foi aceito.

2) O comprimento é composto pelo número de octetos dependentes da resposta.

3) Os octetos de resposta serão vistos posteriormente, em 7.3.

Tabela 3 – Mensagem de resposta

Octeto	Conteúdo
1	Flag falha/sucesso
2	Comprimento (octeto mais significativo)
3	Comprimento (octeto menos significativo)
4	Octeto dependente da resposta
5	Octeto dependente da resposta
6	...

7.3 COMANDOS

Os comandos são compostos por dois octetos. O primeiro octeto define a classe de comando. O segundo, chamado de função, identifica o comando específico dentro da classe [10]. Cabe ressaltar que nem todos os comandos que foram inseridos no sensor inteligente desencadeiam alguma operação no TIM, mas por serem obrigatórios no padrão IEEE 1451 devem gerar alguma resposta.

7.3.1 COMANDOS COMUNS AO TIM E AO CANAL DO TRANSDUTOR

Essa classe de comando pode ser endereçada ao TIM ou a um canal específico. A tabela 4

mostra as funções do comando, os octetos dependentes do comando e os octetos dependentes da resposta que foram inseridos no sensor inteligente. Nessa classe a função a ser destacada é a *Read TEDS segment*, pois é por meio desse comando que o nó de rede acessa os TEDS.

Tabela 4 – Comandos Comuns ao TIM e ao canal do transdutor

Função do Comando	Octetos dependentes do comando		Octetos dependentes da resposta	
	Quant.	Descrição	Quant.	Descrição
Query TEDS	1	Código de acesso ao TEDS	1	Atributos do TEDS
			1	Status do TEDS
			4	Tamanho do TEDS
			2	CheckSum do TEDS
			4	Tamanho máximo do TEDS
Read TEDS segment	1	Código de acesso ao TEDS	4	Offset de leitura do TEDS
	4	Offset para leitura do TEDS	N	Octetos de composição do TEDS
Write TEDS segment	1	Código de acesso ao TEDS	-	
	4	Offset para escrita do TEDS	-	
	N	Octetos para compor o TEDS	-	
Update TEDS	1	Código de acesso ao TEDS	-	
Run Self-Test	1	Tipo de teste		
Write Service request mask	4	Máscara de serviço		
Read Service request mask	-		4	Máscara de serviço
Read status-event register	-		4	Status do registrador de eventos
Read status-condition register	-		4	Status do registrador de condição
Clear status-event register	-		-	
Write status-event protocol state	1	Habilita / desabilita protocolo	-	
Read status-event protocol state	-		1	Protocolo habilitado / desabilitado

7.3.2 COMANDOS COM O TRANSDUTOR EM ESPERA (*IDLE*)

Os comandos com o transdutor em espera só serão executados se o transdutor estiver nesse estado. A única função de comando obrigatória é a de definição de grupo (*Adress group definition*). Na definição de grupo são necessários dois octetos dependentes da função com o endereço do grupo e não são esperados octetos de resposta. Nessa classe de comando também está a função de calibração (*Calibrate transducer Channel*), que não necessita enviar octetos dependentes da função e nem espera octetos de resposta.

7.3.3 COMANDOS COM O TRANSDUTOR EM ESTADO DE OPERAÇÃO

Os comandos com o transdutor em espera só serão executados se o transdutor estiver nesse estado. A tabela 5 mostra os comandos implementados no sensor inteligente. Nessa classe destaca-se a função *Read TransducerChannel data-set segment*, pois por meio dessa será realizada a leitura dos transdutores de temperatura pelo nó de rede.

Tabela 5 – Comandos com o transdutor em operação

Função do Comando	Octetos dependentes do comando		Octetos dependentes da resposta	
	Quant	Descrição	Quant	Descrição
Read TransducerChannel data-set segment	4	Offset de leitura	4	Offset de leitura
			N	Dados do transdutor
Trigger	-		-	

7.3.4 COMANDOS COM O TRANSDUTOR EM OPERAÇÃO OU EM ESPERA

Os comandos aqui listados podem ser executados com o transdutor em qualquer estado. A tabela 6 mostra os comandos implementados.

Tabela 6 - Comandos com o transdutor em qualquer estado

Função do Comando	Octetos dependentes do comando		Octetos dependentes da resposta	
	Quant	Descrição	Quant	Descrição
Transducer Channel Operate	-		-	
Transducer Channel Idle	-		-	
Read Transducer Channel trigger state	-		1	Trigger habilitado / desabilitado
Read Adress Group assignment	-		2	Definição de grupo

7.3.5 COMANDOS COM O TIM ATIVO

Os comandos com o TIM ativo podem ser executados sempre que o TIM estiver energizado.

A tabela 7 mostra esses comandos.

Tabela 7 – Comandos com o TIM ativo

Função do Comando	Octetos dependentes do comando		Octetos dependentes da resposta	
	Quant	Descrição	Quant	Descrição
Read TIM version	-		2	Versão do TIM
Store Operational setup	1	Estado a ser armazenado	-	
Recall Operational setup	1	Estado a ser restaurado	-	
Read IEEE 1451.0 Version	-		1	Versão do padrão IEEE 1451.0 utilizado no TIM

8 . ROTINA DE AUTO-CALIBRAÇÃO

A rotina de auto-calibração empregada no projeto do sensor inteligente é baseada no método de regressão linear.

8.1 MÉTODO DE REGRESSÃO LINEAR

Supondo uma grandeza física *y*, relacionada com outra grandeza *x*, mediante a função *y=ax+b*, uma reta de inclinação *a* cuja ordenada na origem é *b* [19]. Os desvios "*e*" dos valores de *y*, mostrados na figura 9, serão:

$$e_1 = y_1 - (ax_1 + b)$$
$$e_2 = y_2 - (ax_2 + b)$$
$$\dots$$
$$e_i = y_i - (ax_i + b)$$
$$e_n = y_n - (ax_n + b)$$

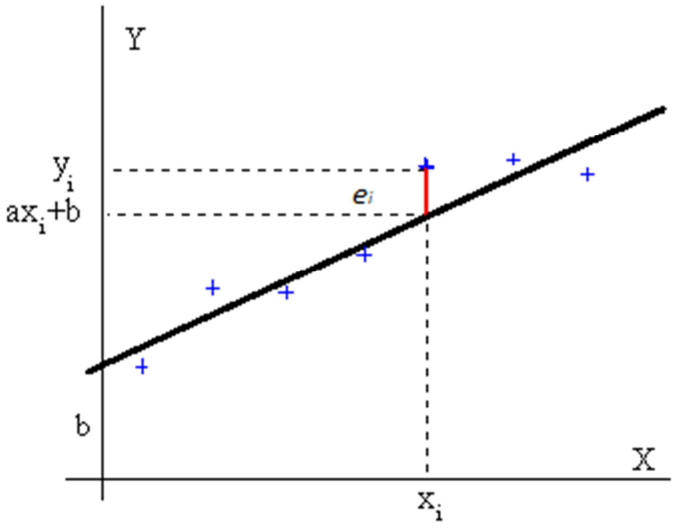

Figura 9 – Exemplo de uma curva ajustada por regressão linear

Seja *E(a,b)* a soma dos quadrados de todos estes desvios:

$$E(a,b) = (y_1 - ax_1 - b)^2 + (y_2 - ax_2 - b)^2 + \cdots (y_i - ax_i - b)^2 + (y_n - ax_n - b)^2$$

$$E(a,b) = \sum_{i=1}^{n} (y_i - ax_i - b)^2$$

Os valores que minimizam o $E(a,b)$ são aqueles para os quais:

$$\frac{\partial E}{\partial a} = 0 \qquad\qquad \frac{\partial E}{\partial b} = 0$$

Dessa forma temos, um sistema de duas equações com duas incógnitas *a* e *b*. A solução para o sistema obtido é:

$$a = \frac{n \sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n \sum_{i=1}^{n} {x_i}^2 - (\sum_{i=1}^{n} x_i)^2} \qquad (1)$$

$$b = \frac{\sum_{i=1}^{n} y_i - a \sum_{i=1}^{n} x_i}{n} \qquad (2)$$

8.2 METODOLOGIA DO ALGORITMO DE CALIBRAÇÃO

Os sensores de referência e o sensor a ser calibrado serão inseridos em um sistema com temperatura ambiente que será elevada gradativamente, sendo coletadas 20 amostras de cada sensor. O sensor de referência possui uma resposta definida pelo fabricante, conforme [13], sendo utilizado para informar ao sistema a temperatura do ambiente, o *"y"* do método de regressão linear. Já o sensor a ser calibrado fornecerá o *"x"* do método (para maiores detalhes de como serão conectados ao conversor A/D ver seção 10). Com as equações (1) e (2) serão obtidos, respectivamente, o coeficiente angular e o coeficiente linear, os quais serão armazenados no TEDS de calibração do sensor a ser calibrado.

9 . OPERAÇÃO DO SENSOR INTELIGENTE

Como o sensor inteligente é composto por dois controladores, a operação dos mesmos será feita da seguinte maneira:

1) O nó sensor operará no modo *Free running*, isto é, ficará adquirindo constantemente os dados provenientes dos transdutores de temperatura e somente executará ações diferentes desta quando solicitado pelo nó de rede. Nessas ações estão incluídas a rotina de auto-calibração, o envio das informações dos TEDS e das informações dos transdutores. A figura 7 mostra o fluxograma de operação do nó sensor.

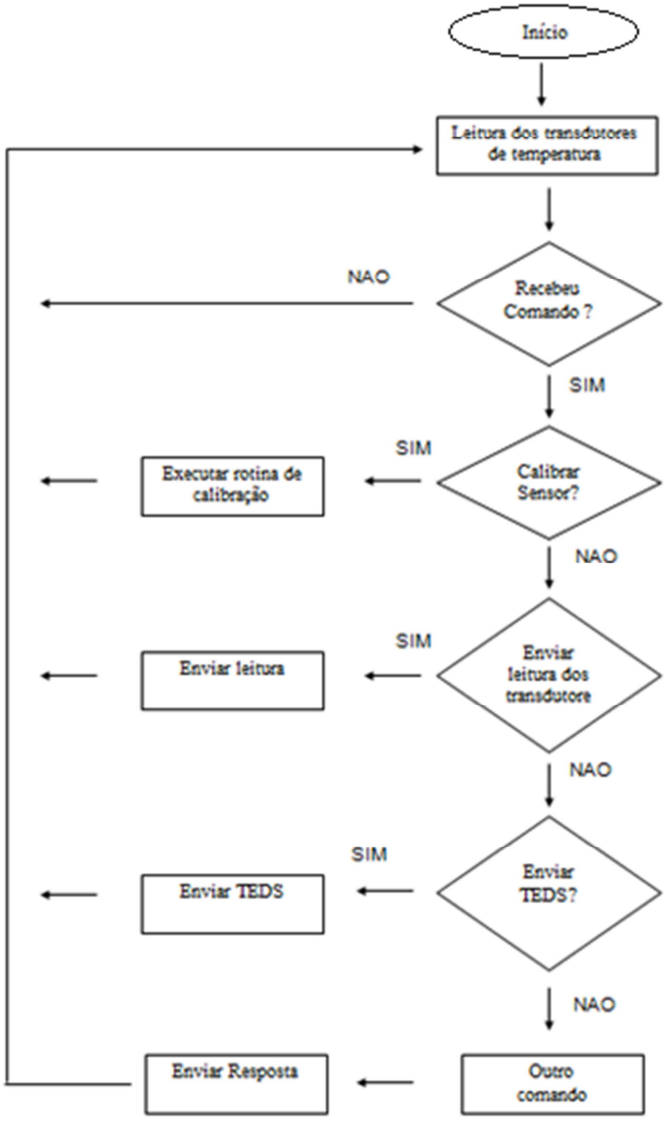

Figura 10 – Fluxograma nó sensor

2) O nó de rede solicitará as informações dos TEDS uma vez ao ser energizado e a cada segundo solicitará leitura das informações dos transdutores e exibirá essas informações no display LCD. A figura 8 mostra o fluxograma de operação do nó de rede. A informação dos transdutores será disponibilizada ao nó de rede em forma de contagens do conversor A/D (como descrito no TEDS de canal do transdutor - TLV Definições de amostragem) e então será aplicada a equação contida no TEDS de calibração - TLV "Método de Conversão Linear", sendo após executada a conversão de escalas do valor final da temperatura de Kelvin para Celsius pelo TLV "Conversão de unidades do SI", também do TEDS de calibração. A equação (3) mostra como é obtida a temperatura no nó de rede.

$$T = Slope_{Conv\ Linear} \,.\, Cont\ AD_{12\ bits} + Interc_{Conv\ Linear} - Interc_{Conv\ de\ un} \quad (3)$$

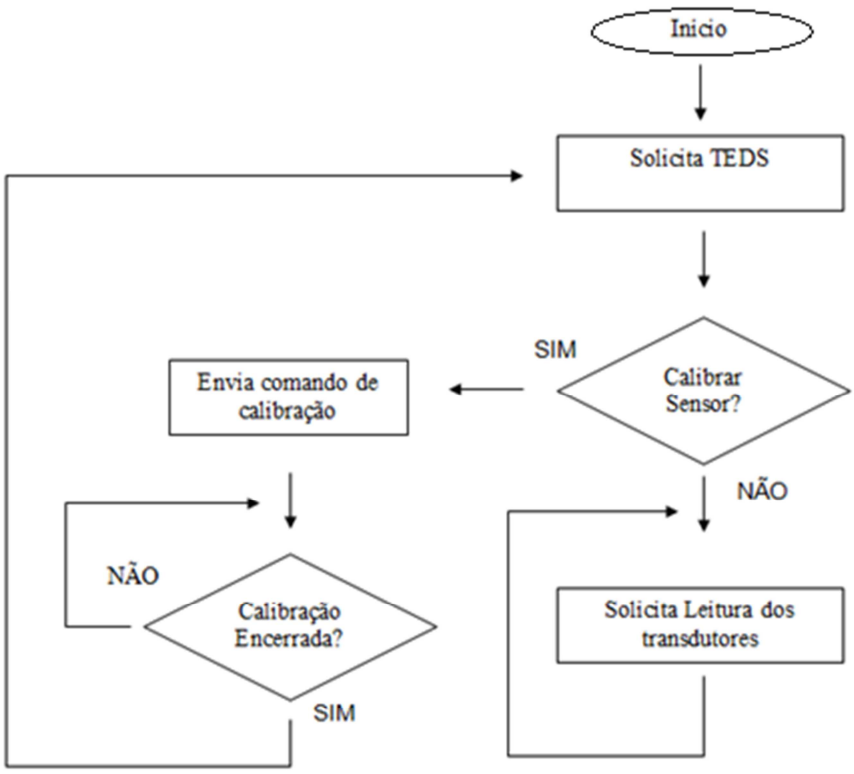

Figura 11 – Fluxograma nó de rede

10 . RESULTADOS OBTIDOS

O sensor inteligente foi testado em etapas:

1) Teste do LCD do nó de rede – esse teste foi realizado, por meio do LCD onde são exibidas as leituras do sensor e as informações dos TEDS, além dele ser utilizado como uma maneira de depurar o funcionamento das etapas de testes seguintes. Nessa etapa não foram encontradas dificuldades, pois foram fornecidos *drivers* pelo fabricante do kit que facilitaram a utilização dessa estrutura.

2) Teste dos canais A/D do nó sensor – executado primeiramente com a utilização de um potenciômetro conectado a tensão de referência do AD fornecendo o sinal de entrada, testando os limites de operação do A/D de 12 bits do kit de desenvolvimento. Cabe salientar que não foram utilizados condicionadores de sinal na entrada do A/D durante o desenvolvimento desse projeto e a referência para o canal A/D foi interna ao controlador, ou seja, 3,3V. Essa configuração do A/D de 12 bits, operando com referência de 3,3V, aliado à resposta do LM35 foram utilizados para fornecer os limites de operação do sensor, inferior em 275 e superior em 603 Kelvin, descritos, respectivamente, nos TLV's "Limite inferior de operação" e "Limite Superior de operação", do TEDS de canal do transdutor. Nessa etapa também não foram encontradas dificuldades.

3) Teste do *transceiver* dos nodos – essa etapa foi a mais crítica ao desenvolvimento do projeto. Apesar de ser fornecido um *driver* para realização da comunicação entre os nodos foi encontrada uma dificuldade de sincronizar a operação deles. Foi necessário criar uma rotina de monitoramento da sincronia e reenvio de informação no nó de rede. O código criado para solicitação das informações dos sensores e de sincronia de operação dos rádios é disponibilizado no apêndice A. Nessa etapa também foi configurado o canal de operação do sensor inteligente, configurado como canal 25. A figura 12 mostra as frequências e os canais de operação do padrão IEEE 802.15.4, no qual está baseado o rádio dos nodos. No teste dos *transceivers* também foram configurados, com o tamanho de cem bytes, os buffers de

transmissão e recepção dos nodos. Esse tamanho foi assim definido para que em um único pacote de dados fosse possível enviar o maior TEDS contido no nó sensor, no caso o TEDS de canal do transdutor, que contém 93 bytes (octetos).

Figura 12 – Canais e frequências de operação do IEEE 802.15.4 (fonte [20])

4) Teste da rotina de envio de comandos e recebimento de respostas – quanto aos comandos e a recepção de resposta, não foi exigida uma grande complexidade, mas as rotinas desenvolvidas foram bastante extensas. Os testes nessas rotinas foram bem-sucedidos, mas novamente o funcionamento dos *transceivers* atrapalhou o perfeito funcionamento das rotinas. Ao serem alterados os argumentos da rotina de preenchimento do buffer de transmissão (função envia_comando, mostrada no Apêndice A) o nó de rede realizava o envio do comando, mas o nó sensor não recebia o pacote de dados. Com argumentos fixos a função funcionava corretamente. Esse fato levou a necessidade de serem criadas funções distintas para solicitação de envio de TEDS. No apêndice B são mostradas duas das funções que contornaram o problema do envio de comandos.

5) Teste de rotinas de atualização de TEDS – Esse teste não apresentou problemas. Como já descrito anteriormente, apenas alguns campos apresentam a possibilidade de serem atualizados, sendo esses campos os contidos no TLV Método de Conversão Linear e os do *Checksum*, ambos do TEDS de calibração. A atualização dos TEDS é feita após a realização da rotina de auto-calibração, os campos são armazenados na memória *flash* do controlador. A gravação das informações dos campos na memória *flash* não apresentou problemas, visto que é fornecido pelo fabricante um *driver* que auxilia nessa tarefa. A figura 13 mostra como

é feita a leitura e atualização dos TEDS. No Apêndice C é possível visualizar a rotina de atualização do TEDS de calibração, denominada “grava_teds”.

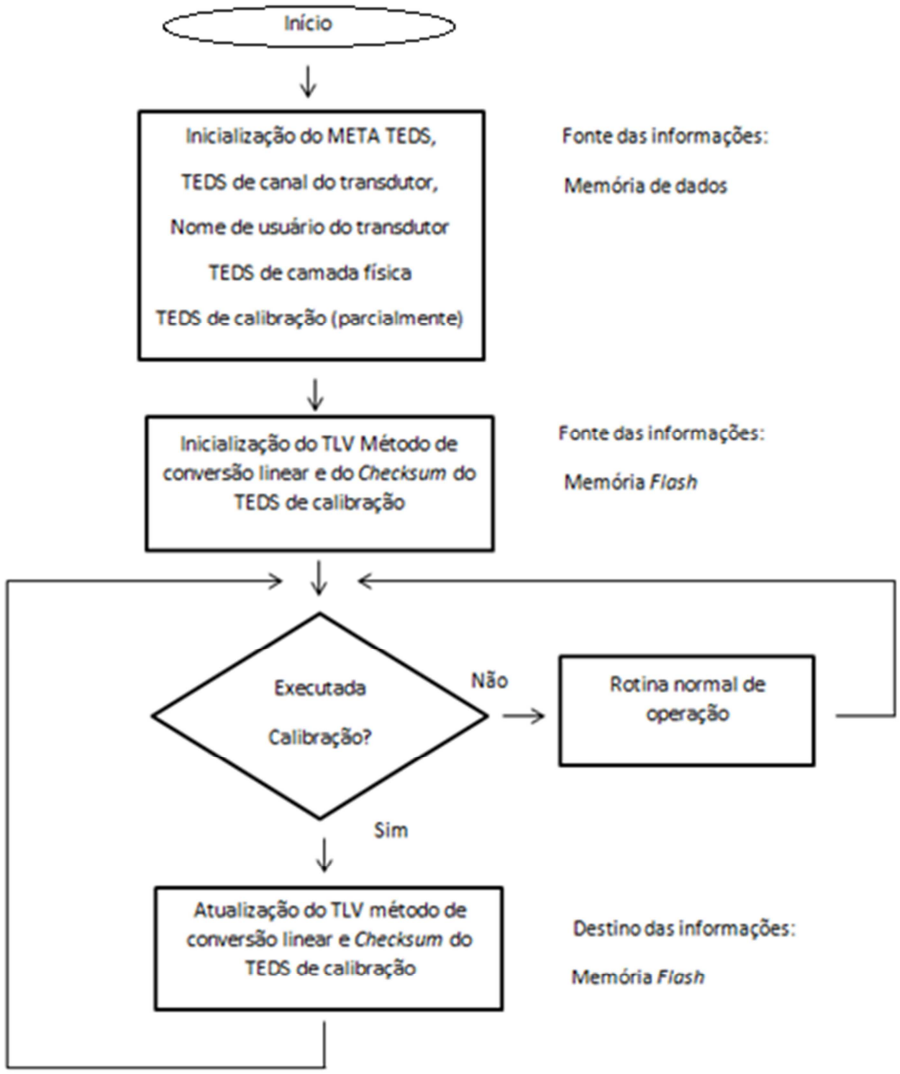

Figura 13 - Leitura e atualização dos TEDS

6) Exibição das informações dos sensores de temperatura no LCD do nó de rede – nesse foi feito todo o teste funcional do sensor operando sob o padrão IEEE 1451. Inicialmente o nó de rede solicitou o envio dos TEDS e exibiu as informações neles contidas no LCD e posteriormente solicitou continuamente a leitura dos sensores de referência e do sensor operacional.

7) Teste da rotina de auto-calibração – Com o sensor já em operação sob o padrão IEE 1451 foi elaborada a rotina de auto-calibração. Como foram utilizados dois sensores LM35 e estes

fornecem saídas em tensão muito parecidas quando estão sob a mesma temperatura de operação, não existiria sentido em realizar uma calibração nessa situação. Para gerar uma diferença entre os sinais de saída, no sensor operacional foi conectado um potenciômetro, formando um divisor resistivo na saída do mesmo, conforme mostra a figura 14. Na figura 15 são mostradas as retas temperatura *versus* tensão obtidas a partir das conexões mostradas na figura 14. A rotina foi embasada no conteúdo abordado na seção 8.2. No Apêndice C é disponibilizada a rotina de auto-calibração desenvolvida.

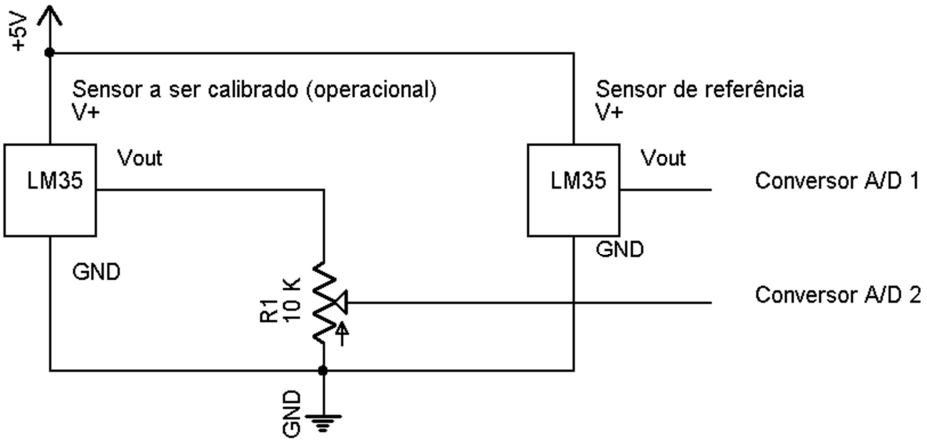

Figura 14 - Conexão dos sensores aos canais A/D

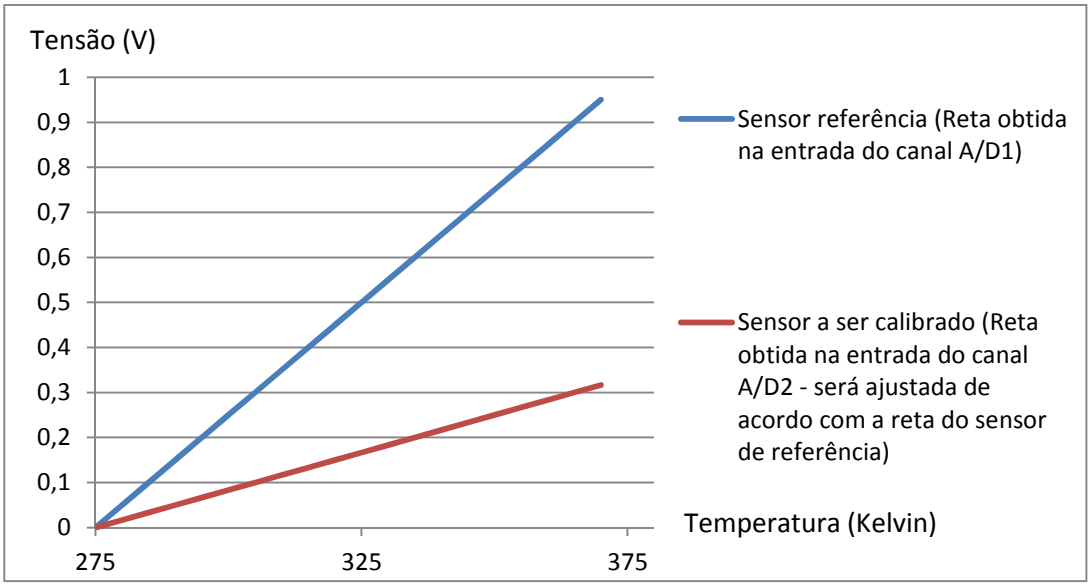

Figura 15 - Retas de resposta dos sensores

Ao final dos testes verificou-se o funcionamento adequado do sensor, o qual pode ser verificado pela correção da resposta do sensor operacional, de forma que os objetivos propostos no início deste trabalho foram atendidos. A figura 16 mostra as temperaturas e as leituras dos canais A/D obtidas com o sensor de referência e o sensor operacional após a execução da rotina de auto-calibração.

Figura 16 - Sensor inteligente em operação e calibrado

11 . CONCLUSÕES

Neste trabalho foi apresentado o desenvolvimento de um sensor inteligente sem fio em conformidade com o padrão IEEE 1451, sendo ainda inserido um procedimento para realização de auto-calibração.

O padrão IEEE 1451 cobre as mais variadas configurações de rede de sensores e ainda está sendo ampliado para se tornar mais abrangente e adequar-se a redes amplamente difundidas atualmente. A grande vantagem de um sensor padronizado pelo IEEE 1451 está na informação que ele carrega em sua memória, os TEDS, as quais permitem auto-identificação e auto-configuração, facilitando a inserção de nodos em uma rede, uma grande funcionalidade para redes sem fio. A possibilidade de alteração de algumas das informações dos TEDS durante a operação do sensor também é muito útil, e foi esta função que permitiu a criação de uma rotina de auto-calibração

Quanto à inserção do padrão IEEE 1451 em um sensor inteligente não é uma tarefa muito complexa, mas é bastante demorada, devido a grande quantidade de informação que precisa ser disponibilizada nos TEDS e também ao número de comandos que precisam ser inseridos no sensor. O hardware também não necessita ser muito complexo, devendo este conter uma interface de comunicação (com ou sem fio) e uma memória capaz de alocar os TEDS e as rotinas de funcionamento do sensor.

Em suma, pode-se dizer que o IEEE 1451 é um padrão ainda não muito utilizado devido a seu tempo de publicação, mas com um grande potencial de aplicação.

12 . TRABALHOS FUTUROS

Como o sensor inteligente foi desenvolvido com um protocolo não utilizado em indústrias, o *Simple MAC*, propõe-se que seja utilizado o protocolo ZigBee ou Bluetooth na interface entre o nó de rede e o nó sensor pois esses protocolos são sugeridos no padrão IEEE 1451.5 como possibilidade de interface sem fio. Sugere-se a substituição dos módulos utilizados, devido principalmente a problemas encontrados durante o desenvolvimento do sensor, os quais foram relatados na seção 10. Além disso, sugere-se que o sensor de referência seja transformado em um nó de referência, de forma que esse nó possa ser utilizado como referência para diferentes nodos operacionais na rotina de auto-calibração. Sugere-se ainda que uma nova aplicação buscando uma rede com um maior número de nodos seja utilizada e avaliadas as suas limitações.

13 . REFERÊNCIAS

[1] Antonio A.F. Loureiro, José Marcos S. Nogueira, Linnyer Beatrys Ruiz, Raquel Aparecida de Freitas Mini, Eduardo Freire Nakamura, Carlos Maurício Seródio Figueiredo. Redes de Sensores Sem Fio. Universidade Federal de Minas Gerais.

[2] http://cuimarinfo.blogspot.com/2009/10/rede-de-sensores-sem-fios-conhece-esta.html, acessado em 15/05/2011

[3] http://www.teleco.com.br/tutoriais/tutorialrssf/pagina_3.asp, acessado em 15/05/2011

[4] http://www.gta.ufrj.br/seminarios/semin2002_1/flavio/, acessado em 14/05/2011

[5] Lucas Bortolaso Torri. A norma IEEE 1451 aplicada a redes heterogêneas de sensores sem fio. Universidade Federal de Santa Catarina – Outubro 2008.

[6] Vítor Manuel Rodrigues Viegas. Projecto e Implementação de um Sistema de Sensores Inteligentes Baseado na Norma IEEE 1451. Universidade Técnica de Lisboa – Julho 2003.

[7] Rafael Marcelino de Jesus. MATIW 1451. Monitoramento e Acionamento de Transdutores Inteligentes através da Web (Padrão IEEE 1451). Universidade Estadual Paulista Júlio de Mesquita Filho - Agosto 2007.

[8] Vítor Viegas, J. M. Dias Pereira, P.M.B. Silva Girão. A Brief Tutorial on the IEEE1451.1 Standard. IEEE Instrumentation & Measurement Magazine - Abril 2008.

[9] site http://pt.wikipedia.org/wiki/API#cite_note-0, acessado em 14/05/2011

[10] IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats. IEEE Std 1451.0 - 2007.

[11] Erico Meneses Leão. Uma Arquitetura de Sensores Inteligentes Disparada por Eventos. Universidade Federal do Rio Grande do Norte – Julho 2007

[12] Eugene Y. Song, Kang Lee. Understanding IEEE 1451 - Networked Smart Transducer Interface Standard. IEEE Instrumentation & Measurement Magazine - Abril 2008.

[13] LM35 Precision Centigrade Temperature Sensors. National Semiconductor - Novembro 2008.

[14] 1322x Network Node - Reference Manual. Rev 1.4. Freescale Semiconductor – Setembro 2010.

[15] 1322x Sensor Node - Reference Manual. Rev 1.5. Freescale Semiconductor – Novembro 2010.

[16] MC1322x Advanced ZigBee™- Compliant Platform-in-Package (PiP) for the 2.4 GHz IEEE® 802.15.4 Standard. Freescale Semiconductor – Outubro 2010.

[17] IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats. IEEE Std 1451.5-2007

[18] MC132X Simple Media Access Controller (SMAC) - Reference Guide. Rev 1.4. Freescale Semiconductor – Maio 2009.

[19] site http://www.fisica.ufs.br/egsantana/cinematica/regresion/regresion.htm, acessado em 14/05/2011

[20] site http://www.eetimes.com/design/embedded-internet-design/4204872/ZigBee-applications---Part-3--ZigBee-PANs/, acessado em 4/07/2011.

14 . APÊNDICE A – CÓDIGO UTILIZADO NA SOLICITAÇÃO DAS INFORMAÇÕES DOS SENSORES PELO NÓ DE REDE E NA SINCRONIZAÇÃO DOS RÁDIOS

```
void NCAP_processo(void)
{
 (void)process_radio_msg();         /* Verifica se chegou alguma mensagem via
                                radio e seta o gbDataIndicationFlag caso  tenha chegado*/
 if(TRUE == gbDataIndicationFlag)          //Testa se chegou alguma informação
 {
  gbDataIndicationFlag = FALSE;     // limpa gbIndicationFlag
  Led1On();                        // sinaliza chegada nde mensagem
  DelayMs(49);
  Led1Off();
  Led3Off();                       // limpa o sinalizador de reenvio
  processa =0;                     //limpa flag de monitoramento de sincronia
  DelayMs(49);

  if (sensor == TEMP_OPERACIONAL)   // processa o sinal do sensor operacional
     {
     sensor = TEMP_REF;           /* manda o nó de rede solicitar no próximo
                                comando a leitura do sensor de referência*/
     operacional = (RX_msg.pu8Buffer->u8Data[4] & 0x0F) << 8;
     operacional = operacional + RX_msg.pu8Buffer->u8Data[5];
     temperatura = angular.fl1*operacional + linear.fl;
     LCD_WriteString_NormalFont(5,"                    ");
     LCD_WriteString_NormalFont(6,"                    ");
     LCD_WriteStringValue("Leitura AD ",(operacional),5,gLCD_DecFormat_c);//exibe leitura
     LCD_WriteStringValue("Temperatura OP ",(temperatura),6,gLCD_DecFormat_c); /*exibe
                                                                                Leitura*/

     }
```

```
  else                        //processa o sinal do sensor de referência
      {
      sensor = TEMP_OPERACIONAL;   /* manda o nó de rede solicitar no próximo
                                          comando a leitura do sensor operacional*/
      referencia = (RX_msg.pu8Buffer->u8Data[4] & 0x0F) << 8;
      referencia = referencia + RX_msg.pu8Buffer->u8Data[5];
      temperatura = 0.080078 * referencia + 2;
      LCD_WriteString_NormalFont(3,"                ");
      LCD_WriteString_NormalFont(4,"                ");
      LCD_WriteStringValue("Leitura AD ",(referencia),3,gLCD_DecFormat_c); //exibe leitura
      LCD_WriteStringValue("Temperatura REF ",(temperatura),4,gLCD_DecFormat_c); /*exibe
                                                                                  Leitura*/

      }

  DelayMs(400);
  transmitir = TRUE;            //Flag que indica que o radio deve transmitir
  buffer_trans = FALSE;          /*Flag que indica que o buffer de transmissão
                                 não é válido*/
  ouvir = FALSE;               /*Flag indica ao radio que ele não deve ficar
                                  aguardando dados em seu buffer de recepção*/
  TX_msg.u8Status.msg_state = MSG_TX_ACTION_COMPLETE_SUCCESS;
  RX_msg.u8Status.msg_state = MSG_RX_ACTION_COMPLETE_SUCCESS;
}
  if (transmitir == TRUE && buffer_trans == FALSE)  /*manda atualizar o buffer de
                                                        transmissão*/
{
     if (sensor == TEMP_OPERACIONAL) /* testa se deve ser solicitada a leitura
                              do sensor operacional*/
     {
     dados[0]= 0;
     dados[1]= 0;
     dados[2]= 0;
     dados[3]= 0;
       envia_comando (TEMP_OPERACIONAL, XDCR_OPERATE_STATE,
```

```
        READ_TRANSDUCER_CHANNEL_DATA_SET_SEGMENT, 0);
      //a função envia_comando preenche o buffer de transmissão
     }

     if(sensor == TEMP_REF)        /* testa se deve ser solicitada a leitura
                                do sensor de referência*/
      {
     dados[0]= 0;
     dados[1]= 0;
     dados[2]= 0;
     dados[3]= 0;
    envia_comando (TEMP_REF, XDCR_OPERATE_STATE,
    READ_TRANSDUCER_CHANNEL_DATA_SET_SEGMENT, 0);
     }

     TX_msg.u8Status.msg_state = MSG_TX_ACTION_COMPLETE_SUCCESS;
     RX_msg.u8Status.msg_state = MSG_RX_ACTION_COMPLETE_SUCCESS;
     buffer_trans = TRUE;
}

if(is_tx_msg_final_state(TX_msg) && transmitir==TRUE)// testa se o radio deve transmitir
     {
       MCPSDataRequest(&TX_msg); //transmite a informação
       Led2On();
       DelayMs(50);
       Led2Off();
       DelayMs(50);
       transmitir = FALSE;       //Flag para o rádio sair do modo de transmissão
       ouvir = TRUE;             /* Flag para o radio entrar no modo de recepção*/
     }

if(is_rx_msg_final_state(RX_msg) && ouvir==TRUE)  /*testa se o radio deve entrar em
                                                        modo de recepção*/
     {
       RX_msg.u8BufSize = RX_SIZE;
```

```
        MLMERXEnableRequest(&RX_msg, 0xffffffff); // coloca o radio no modo de recepção
       }
  if (processa > 100000) // monitoramento da sincronia entre os nodos
  {
   processa =0;         // limpa variavel de monitoramento de sincronia
   transmitir = TRUE;   // Flag para o rádio entrar no modo de transmissão
   buffer_trans = FALSE; /*Flag que indica que o buffer de transmissão
                                  não é válido*/
   ouvir = FALSE;                /*Flag indica ao radio que ele não deve ficar
                                   aguardando dados em seu buffer de recepção*/
   TX_msg.u8Status.msg_state = MSG_TX_ACTION_COMPLETE_SUCCESS;
   RX_msg.u8Status.msg_state = MSG_RX_ACTION_COMPLETE_SUCCESS;
   Led3On();            // seta indicador de reenvio de dados
  }

  processa++; // variavel de monitoramento do funcionamento do radio
}
```

15 . APÊNDICE B – ROTINAS UTILIZADAS NA SOLICITAÇÃO DE TEDS

```
/*****************************************************************************
*****
* le_META_TEDS
*
* Busca os TEDS no TIM e armazena na memoria.
*****************************************************************************
****/
void le_META_TEDS(void)
{
 uintn8_t cont;
 uintn8_t TEDS_ack = FALSE;

 while (TEDS_ack == FALSE)
 {
 (void)process_radio_msg();

 if(TRUE == gbDataIndicationFlag)
 {
    for (cont = 0; cont<40; cont++)
    {
    META_TEDS_ST[cont] = RX_msg.pu8Buffer->u8Data [cont+4];  /* Realiza o armazenamento
                                                  dos TEDS recebidos na estrutura META_TEDS_ST[]*/
    }
  gbDataIndicationFlag = FALSE;
  Led1On();
  DelayMs(49);
  Led1Off();
  Led3Off();
  DelayMs(49);
  TEDS_ack = TRUE;
  transmitir = TRUE;
  buffer_trans = FALSE;
  ouvir = FALSE;
```

```
  TX_msg.u8Status.msg_state = MSG_TX_ACTION_COMPLETE_SUCCESS;
  RX_msg.u8Status.msg_state = MSG_RX_ACTION_COMPLETE_SUCCESS;
  DelayMs(400);

 }


 if (transmitir == TRUE && buffer_trans == FALSE)
 {
     dados[0]= META_TEDS; /* coloca no campo de informação que o TEDS desejado é o
                                 META TEDS*/
     dados[1]= 0;
     dados[2]= 0;
     dados[3]= 0;
     dados[4]= 0;
     envia_comando (TIM_OP, COMMON_CMD_STATE, READ_TEDS_SEGMENT, 0);
//função envia_comando preenche o buffer de transmissão
     TX_msg.u8Status.msg_state = MSG_TX_ACTION_COMPLETE_SUCCESS;
     RX_msg.u8Status.msg_state = MSG_RX_ACTION_COMPLETE_SUCCESS;
     buffer_trans = TRUE;
 }


 if(is_tx_msg_final_state(TX_msg) && transmitir==TRUE)
     {
      MCPSDataRequest(&TX_msg);
      Led2On();
      DelayMs(50);
      Led2Off();
      DelayMs(50);
      transmitir = FALSE;
      ouvir = TRUE;
     }

 if(is_rx_msg_final_state(RX_msg) && ouvir==TRUE)
     {
      RX_msg.u8BufSize = RX_SIZE;
```

```
        MLMERXEnableRequest(&RX_msg, 0xffffffff);
       }
  if (processa > 100000)
  {
   processa =0;
   transmitir = TRUE;
   buffer_trans = FALSE;
   ouvir = FALSE;
   TX_msg.u8Status.msg_state = MSG_TX_ACTION_COMPLETE_SUCCESS;
   RX_msg.u8Status.msg_state = MSG_RX_ACTION_COMPLETE_SUCCESS;
   Led3On();
  }

  processa++; // variavel de monitoramento do funcionamento do radio
  }
}

/*****************************************************************************
*****
* le_CHANNEL_TEDS
*
* Busca os TEDS no TIM e armazena na memoria.
*****************************************************************************
****/
void le_CHANNEL_TEDS(void)
{
  uintn8_t cont;
  uintn8_t TEDS_ack = FALSE;

  while (TEDS_ack == FALSE)
  {
  (void)process_radio_msg();

  if(TRUE == gbDataIndicationFlag)
  {
     for (cont = 0; cont<93; cont++)
      {
```

```
    TRANSDUCER_CHANNEL_TEDS_ST[cont] = RX_msg.pu8Buffer->u8Data [cont+4];
/* coloca o TEDS de canal do transdutor recebido na estrutura TRANSDUCER_CHANNEL_TEDS_ST []*/

    }
  gbDataIndicationFlag = FALSE;
  Led1On();
  DelayMs(49);
  Led1Off();
  Led3Off();
  DelayMs(49);
  TEDS_ack = TRUE;
  transmitir = TRUE;
  buffer_trans = FALSE;
  ouvir = FALSE;
  TX_msg.u8Status.msg_state = MSG_TX_ACTION_COMPLETE_SUCCESS;
  RX_msg.u8Status.msg_state = MSG_RX_ACTION_COMPLETE_SUCCESS;
  DelayMs(400);
 }

 if (transmitir == TRUE && buffer_trans == FALSE)
 {
     dados[0]= CHANNEL_TEDS; /* coloca no campo de informação que o TEDS desejado é o
                                TEDS de canal do transdutor*/

     dados[1]= 0;
     dados[2]= 0;
     dados[3]= 0;
     dados[4]= 0;
     envia_comando (TEMP_REF, COMMON_CMD_STATE, READ_TEDS_SEGMENT, 0);
//função envia_comando preenche o buffer de transmissão
     TX_msg.u8Status.msg_state = MSG_TX_ACTION_COMPLETE_SUCCESS;
     RX_msg.u8Status.msg_state = MSG_RX_ACTION_COMPLETE_SUCCESS;
     buffer_trans = TRUE;
 }
```

```
  if(is_tx_msg_final_state(TX_msg) && transmitir==TRUE)
        {
         MCPSDataRequest(&TX_msg);
         Led2On();
         DelayMs(50);
         Led2Off();
         DelayMs(50);
         transmitir = FALSE;
         ouvir = TRUE;
        }

  if(is_rx_msg_final_state(RX_msg) && ouvir==TRUE)
        {
         RX_msg.u8BufSize = RX_SIZE;
         MLMERXEnableRequest(&RX_msg, 0xffffffff);
        }
  if (processa > 100000)
  {
   processa =0;
   transmitir = TRUE;
   buffer_trans = FALSE;
   ouvir = FALSE;
   TX_msg.u8Status.msg_state = MSG_TX_ACTION_COMPLETE_SUCCESS;
   RX_msg.u8Status.msg_state = MSG_RX_ACTION_COMPLETE_SUCCESS;
   Led3On();
  }
  processa++; // variavel de monitoramento do funcionamento do radio
  }
}
```

16 . APÊNDICE C – ROTINA DE AUTO-CALIBRAÇÃO DESENVOLVIDA

```
union
   {
      float fl;
      int in;
   } linear;

union
   {
      float fl1;
      int in1;
   } angular;
void calibracao(void)
{
  float auxilia  = 0;
  float auxilia1 = 0;
  float auxilia2 = 0;
  float auxilia3 = 0;
  float auxilia4 = 0;
  uintn8_t indice;

 for (conta=0; conta <= 19; conta++)          // controle do número de amostras
  {
    generic_channel();                        //lê o canal do AD de referência
    generic_channel_1();                      //lê o canal do AD do sensor operacional
    conta_ad[conta]= operacional;
    temper[conta] = 0.080078 * referencia +2;          //temperatura obtida pelo sensor de referência
    Led4On();
    DelayMs(100);
    Led4Off();
```

```
  DelayMs(100);

  for (indice = 0; indice<9; indice++)        //laço determina intervalo de tempo entre as amostras
  {
  Led3On();
  DelayMs(1000);
  Led3Off();
  DelayMs(1000);
  }
}


 for (conta=0; conta <=19; conta++)                    //inicio da rotina de regressão linear
 {
 auxilia = auxilia + (conta_ad[conta]*temper[conta]);
 auxilia1 = auxilia1 + conta_ad[conta];
 auxilia2 = auxilia2 + temper[conta];
 auxilia3 = auxilia3 + (conta_ad[conta]*conta_ad[conta]);
 }

 auxilia = 20 * auxilia;
 auxilia3 = 20 * auxilia3;
 auxilia4 = (auxilia1 * auxilia1);


 angular.fl1 = (auxilia - (auxilia1*auxilia2)) / (auxilia3 - auxilia4);
 linear.fl =  (auxilia2 - angular.fl1*auxilia1) / 20;    //fim da rotina de regressão linear


CALIBRATION_TEDS_ST[48] = (angular.in1 & 0xFF000000) >> 24; /*quebra dos coeficientes
                                                            em octetos*/
CALIBRATION_TEDS_ST[49] = (angular.in1 & 0x00FF0000) >> 16;
CALIBRATION_TEDS_ST[50] = (angular.in1 & 0x0000FF00) >> 8;
CALIBRATION_TEDS_ST[51] = (angular.in1 & 0x000000FF) ;

CALIBRATION_TEDS_ST[52] = (linear.in & 0xFF000000) >> 24;
CALIBRATION_TEDS_ST[53] = (linear.in & 0x00FF0000) >> 16;
```

```
CALIBRATION_TEDS_ST[54] = (linear.in & 0x0000FF00) >> 8;
CALIBRATION_TEDS_ST[55] = (linear.in & 0x000000FF) ;

grava_teds();              // armazenamento dos TEDS na memória Flash
}


void grava_teds(void)          // Rotina de atualização dos TEDS na memória Flash
{
 nvmErr_t  nvmError;
 uintn8_t indice;
 uintn16_t c_sum=0;
 uintn16_t soma = 0;

 for (indice=0; indice<58; indice++)
 {
 soma = soma + CALIBRATION_TEDS_ST[indice];
 }
 c_sum = 0xFFFF - soma;
 CALIBRATION_TEDS_ST[56] = (c_sum & 0xFF00) >> 8;
 CALIBRATION_TEDS_ST[57] = (c_sum & 0x00FF);

 nvmError = NVM_Erase(gNvmInternalInterface_c, NvmType,0x00010000 );
 for (indice = 48; indice<58; indice++)
 {
 nvmError = NVM_BlankCheck(gNvmInternalInterface_c, NvmType, 0x00010000+(indice-48),1);
 if(gNvmErrNoError_c == nvmError)
 {
  NVM_Write(gNvmInternalInterface_c,NvmType,(void)(&CALIBRATION_TEDS_ST[indice]),
  0x00010000+(indice-48), 1);

 }
 }
}

void le_teds(void)             //Rotina de leitura dos TEDS da memória Flash do controlador
{
```

```
  uintn8_t indice;
  for (indice = 48; indice<58; indice++)
  {
  NVM_Read(gNvmInternalInterface_c, NvmType, (void *)(&CALIBRATION_TEDS_ST[indice]),
0x00010000+(indice-48), 1);
  }
}
```