RAFAEL COFFI TONON

REDE DE SENSORES DE TEMPERATURA UTILIZANDO O PROTOCOLO POPNET

Trabalho de Conclusão de Curso
apresentado à Escola de Engenharia de São
Carlos, da Universidade de São Paulo

Curso de Engenharia de Computação

ORIENTADOR: Evandro Luis Linhari Rodrigues

São Carlos
2011

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

Tonon, Rafael Coffi
T666r Rede de sensores de temperatura utilizando o protocolo POPNET / Rafael Coffi Tonon ; orientador Evandro Luis Linhari Rodrigues. -- São Carlos, 2011.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo.

1. Wireless. 2. Protocolos de comunicação. 3. Memória ROM. 4. Sistemas embutidos. I. Título.

AGRADECIMENTOS

Agradeço aos meus pais, Maximino e Rosangela, por minha educação e apoio moral e financeiro durante toda graduação.

Agradeço ao Prof. Dr. Evandro pelo suporte e ajuda para a realização deste projeto.

Agradeço a Mauro Massanori Miyashiro e ao Instituto Eldorado pelo fornecimento dos materiais necessários para o desenvolvimento do projeto.

Agradeço aos meus amigos Paulo, Rodolfo, Vinicius, Gabriel, Marcos e Bruno “Mega” pelo convívio durante os anos de graduação e o apoio para a realização deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

PAN – Rede pessoal (*Personal Area Network*)

WPAN – Rede pessoal sem fio (*Wireless Personal Area Network*)

LR-WPAN – Rede pessoal sem fio de baixas taxas (*Low Rate – Wireless Personal Area Network*)

IEEE – Instituto de Engenheiros Elétricos e Eletrônicos (*Institute of Electrical and Electronics Engineers*)

ISM – Industrial, Medicina e Científico (*Industrial Medicine Scientific*)

PHY – Camada física

MAC – Camada de controle de acesso ao meio (*Media Access Control*)

I²C – Circuito Inter Integrado (*Inter Integrated Circuit*)

EEPROM – Memória Somente de Leitura Programável e Apagável Eletricamente (*Electrically Erasable Programmable Read-Only Memory*)

EPROM – Memória Somente de Leitura Programável e Apagável (E*rasable Programmable Read-Only Memory)*

OSI – Interconexão de Sistemas Abertos (*Open Systems Interconnection*)

AES – Padrão de criptografia avançada (*Advanced Encryption Standard*)

LED – Diodo emissor de luz (*Light-emitting Diode*)

LCD – Display de cristal líquido (*Liquid Crystal Display*)

CPU – Unidade Central de Processamento (*Central Processing Unit*)

PC – Computador Pessoal (*Personal Computer*)

BSP – Pacote de Suporte à Placa (*Board Support Package*)

USB – Barramento Serial Universal (*Universal Serial Bus*)

BDM –*Background Debug Mode*

SDK – Kit de Desenvolvimento de Software (*Software Development Kit*)

IDE – Ambiente de Desenvolvimento Integrado (*Integrated Development Environment)*

RESUMO

Sensores são dispositivos utilizados para medir grandezas físicas. Tais dispositivos são muito utilizados no monitoramento de ambientes e sistemas. Com o monitoramento, é possível prevenir erros e realizar melhorias de acordo com os resultados observados. O objetivo deste trabalho é unir a utilização de sensores ao conceito de redes sem fio para uma aplicação de rede de sensores sem fio, com taxas de transmissão até 250kbps, consumo reduzido de energia e com sensores de temperatura. No intuito de fornecer diretrizes e padronizações para a construção deste tipo de rede, a organização IEEE criou, em 2003, o padrão 802.15.4 e sobre este padrão surgem protocolos adicionando novas funcionalidades. O intuito deste trabalho é apresentar e utilizar um protocolo que roda sobre o padrão 802.15.4: PopNet, desenvolvido pela empresa San Juan Software. A aplicação conta com um dispositivo sem fio como coordenador da rede, conectado a um computador e placas de aquisição remota realizando as medições de temperatura. O microcontrolador utilizado para a comunicação sem fio foi o MC13213, produzido pela Freescale. Foi utilizado também memórias EEPROM para armazenagem dos dados. O sistema desenvolvido com o protocolo PopNet mostrou-se funcional, realizando corretamente o armazenamento das medidas de temperatura na memória, além da comunicação sem fio entre os dispositivos utilizando o conceito de redes *mesh*. O protocolo PopNet é uma boa alternativa para o desenvolvimento de rede de sensores sem fio.

Palavras-chave: sensores sem-fio, padrão 802.15.4, PopNet, redes pessoais, MCU13213, EEPROM**.**

ABSTRACT

Sensors are devices utilized to measure physical quantities. Such devices are wisely utilized to monitoring systems and environments. By monitoring, it's possible to prevent mistakes and make improvements in accordance with the results. The project's objective is to link the use of sensors to the concept of wireless networks by developing an application of wireless sensor network, with transmission rates up to 250kbps, reduced energy consumption and temperature sensors. To provide guidelines and standards in building this type of network, the IEEE organization created in 2003, the 802.15.4 standard. About this standard, new protocols come with new features for development. The purpose of this paper is to present and use a protocol that runs on the 802.15.4 standard: PopNet, developed by San Juan Software. This paper presents the features of this protocol, with the development of a wireless network of temperature sensors. The application relies on a wireless device as the network coordinator, connected to a computer acquisition board and performing remote temperature measurements. The microcontroller used for wireless communication MC13213 was produced by Freescale. It was also used EEPROM memories for data storage. The system developed with the protocol PopNet proved to be functional, correctly performing the storage of temperature measurements in memory, and wireless communication between devices using the concept of mesh networks. The PopNet protocol is a good alternative for the development of wireless sensor network.

Keywords: wireless sensors, 802.15.4 standard, PopNet, personal networks, microcontroller MCU13213, EEPROM

CAPÍTULO 1 - INTRODUÇÃO

Sensor é um dispositivo que mede uma quantidade física e gera uma saída relacionada a tal medida através de um sinal elétrico ou ótico [1]. Para que isso ocorra, é fundamental conhecer a relação entre a grandeza a ser medida e as propriedades do material do sensor. Por exemplo, um termômetro de mercúrio utiliza-se da expansão do mercúrio provocada pela temperatura a fim de se obter um valor mensurável.

A presença de sensores nos dias atuais é enorme. No mercado encontram-se sensores para diversas grandezas físicas: temperatura, umidade, pressão, calor, luz, radiação, som, etc. São incontáveis os dispositivos, aparelhos, ambientes e situações onde sensores são fundamentais para o correto funcionamento. Quando se usa sensores, a ideia principal envolve questões de monitoramento. Monitorar constitui-se uma característica essencial em um sistema, já que registrar e analisar as informações fornece meios para melhorar, observar e prevenir erros em um sistema.

Quando se pensa em aplicações de rede de sensores surgem inúmeros exemplos imediatos. Pode-se imaginar uma indústria com uma rede de sensores de temperatura, umidade, pressão, dentre outras grandezas, nas várias etapas e setores de fabricação; um hospital, visando oferecer melhores condições no ambiente; uma central monitorando diversos caminhões frigoríficos; sensores diversos espalhados por uma casa, com o dono podendo monitorar tudo através de seu computador ou celular; monitorar um automóvel e transmitir a uma central; monitorar condições em usinas geradoras de energia. O exemplo visto na figura 1 demonstra a utilização de uma rede de sensores sem fio para o monitoramento de um vulcão. Com diversos sensores espalhados em torno do vulcão, é possível observar alterações relevantes nos sensores e prever uma possível erupção.

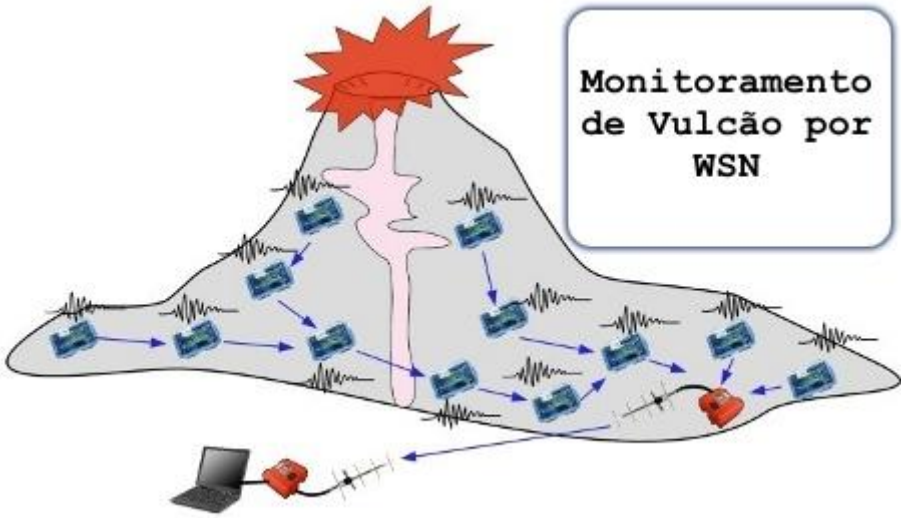

Figura 1- Exemplo de vulcão monitorado por uma rede de sensores sem fio.
Fonte: http://www.vivasemfio.com/blog/redes-de-sensores-sem-fios/

1.1 Motivação

Diante de tantos ambientes e situações onde o monitoramento de certas quantidades físicas se faz necessário, unir o monitoramento com sensores e a comunicação sem fio torna-se uma solução prática e facilmente adaptável a qualquer ambiente, já que dispensa o uso de fios.

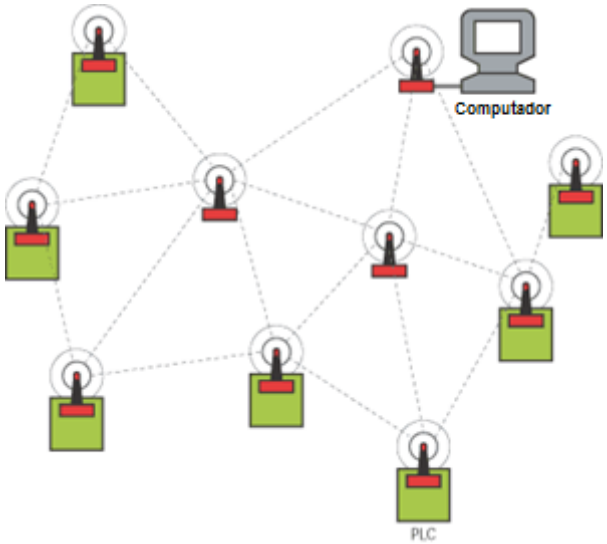

Figura 2 - Rede sem fio em que todos podem receber e transmitir dados.
Fonte: http://www.wirelessbrasil.org/wirelessbr/colaboradores/thienne_johnson/rssf-intro.htm

A utilização de vários sensores que se comunicam por rádio constitui uma rede pessoal sem fio (WPAN). Uma WPAN é uma rede onde o objetivo principal é compartilhar informações entre diversos dispositivos em um curto alcance através da comunicação sem fio. Dentro do conceito de WPAN, existe ainda a LR-WPAN, que

são redes pessoais sem fio de curto alcance que utilizam taxas de transmissão baixas para utilizar menos energia e assim tornar o sistema ainda mais portátil. A figura 2 ilustra uma rede pessoal sem fio em que os nós podem receber e transmitir dados.

As taxas de transmissão baixas não constituem um gargalo para o sistema já que tais redes são para envio de informações simples e pequenas. Uma rede de sensores se encaixa neste conceito, já que os sensores apenas fornecem dados de grandezas físicas, basicamente números.

No intuito de fornecer diretrizes, configurações e padrões para definir e construir uma LR-WPAN, em maio de 2003 surge o protocolo 802.15.4, desenvolvido pela organização IEEE [2]. A organização IEEE é uma associação formada por engenheiros eletricistas que tem como objetivo ajudar o desenvolvimento tecnológico em benefício da humanidade. Para isto, realizam inúmeras palestras, conferências e atividades, além de produzir padrões e publicações que auxiliem na padronização de diversas tecnologias relacionadas.

O protocolo 802.15.4 define a formação da rede WPAN, suas topologias possíveis, velocidades de transmissão, a divisão do espectro de frequência em vários canais, questões de segurança, confiabilidade, consumo de energia, forma de envio de dados, dentre outro aspectos. O protocolo 802.15.4 fornece especificações para a camada física (PHY) e a camada de controle de acesso ao meio (MAC).

Neste contexto surge o grande objetivo deste trabalho: apresentar um protocolo que roda sobre o padrão IEEE 802.15.4. O protocolo PopNet é um protocolo desenvolvido recentemente pela empresa San Juan Software [3] e promete ser um protocolo de fácil e rápida compreensão no desenvolvimento de aplicações que se utilizem de recursos sem fio. Voltado justamente para redes de sensores sem fio, promete oferecer uma interface amigável, com bibliotecas e exemplos prontos para diversos modelos de placas e microcontroladores utilizados em comunicação sem fio.

No entanto, este trabalho não contará apenas com a utilização do protocolo PopNet e a comunicação sem fio. Como a comunicação de uma rede LR-WPAN ocorre a curtas distâncias, uma ideia para melhorar o monitoramento de situações de constantes deslocamentos (como veículos ou caminhões frigoríficos) é utilizar memórias para guardar leituras dos sensores, para posteriormente serem transferidas para uma central de dados. Com isso, este trabalho envolveu questões de memórias EEPROM utilizando o protocolo de comunicação I^2C, além da comunicação necessária entre a leitura dos sensores e o microcontrolador MC13213.

1.2 Objetivo

Estudar o protocolo PopNet e desenvolver uma aplicação que gerenciará uma rede de sensores de temperatura, com um dos nós sendo responsável por receber as leituras dos demais nós, exibindo os dados na tela de um computador.

1.3 Organização

O trabalho está organizado da seguinte forma:

- No capítulo 2 encontra-se a base do conhecimento do trabalho. Nele há uma breve descrição sobre o protocolo IEEE 802.15.4, explicando algumas funcionalidades e o que ele estabelece; a apresentação do protocolo PopNet, descrevendo uma série de recursos e possibilidades e por fim, o protocolo de comunicação I²C utilizado para se comunicar com uma memória EEPROM.
- No capítulo 3 encontra-se todo o desenvolvimento do projeto, desde instruções de como usar o protocolo PopNet, os softwares e equipamentos utilizados, informações sobre os sensores de temperatura, etc.
- No capítulo 4 são apresentados os testes do sistema implementado e os resultados obtidos.
- No capítulo 5, conclusões a cerca da utilização do protocolo PopNet e sugestões para trabalhos futuros.

CAPÍTULO 2 – BASE DO CONHECIMENTO

Este capítulo tem por objetivo fornecer uma rápida introdução sobre os principais conceitos abordados neste trabalho, dentre os quais se podem citar a compreensão do principal protocolo para desenvolvimento de redes pessoais sem fio (WPAN), o IEEE 802.15.4; e uma rápida apresentação sobre o protocolo tema do trabalho, o PopNet. Por fim, compreender o protocolo de comunicação I²C, utilizado para a comunicação com as memórias EEPROM presentes.

2.1 IEEE 802.15.4

O padrão 802.15.4 é desenvolvido pelo grupo IEEE [4] e tem como objetivo fornecer especificações para a camada física (PHY) e de controle de acesso ao meio (MAC) para redes pessoais sem fio. A camada física tem como função converter os bits de dados para um sinal de rádio frequência e vice-versa. Já a camada de controle de acesso ao meio cuida da topologia da rede (se a rede utilizará um esquema ponto-a-ponto ou modo estrela, por exemplo). O IEEE 802.15.4 constitui a base para outros padrões como ZigBee, WirelessHART e também para o PopNet.

O IEEE 802.15.4 trabalha sobre as faixas de frequências denominadas ISM (*Industrial Medicine Scientific*) de 868MHz (Europa), 915MHz(EUA) e 2.4GHz (resto do mundo) que não precisam ser licenciadas. A frequência mais utilizada é a de 2.4GHz, onde suas taxas de transferência atingem até 250kbps. No espectro de 2.4GHz, existem 16 canais, numerados de 11 a 26.

Por utilizar um espectro de frequência bastante conhecido por todos, já que a banda de 2.4GHz também é utilizada em protocolos Wi-fi, pode-se pensar que uma rede 802.15.4 não pode estar presente em um ambiente com uma rede Wi-fi. No entanto, o protocolo estabelece uma série de parâmetros e configurações que permitem que ambas as redes possam coexistir. A figura 3 mostra a coexistência dos canais da rede Wi-Fi e do protocolo 802.15.4: os canais da rede Wi-Fi (em rosa) não colidem com os canais do protocolo 802.15.4 (em azul). Mas como todo sinal sem fio, pode ocorrer interferências e bloqueios de sinal causados por água, metal, cimento, pessoas, plantas e micro-ondas.

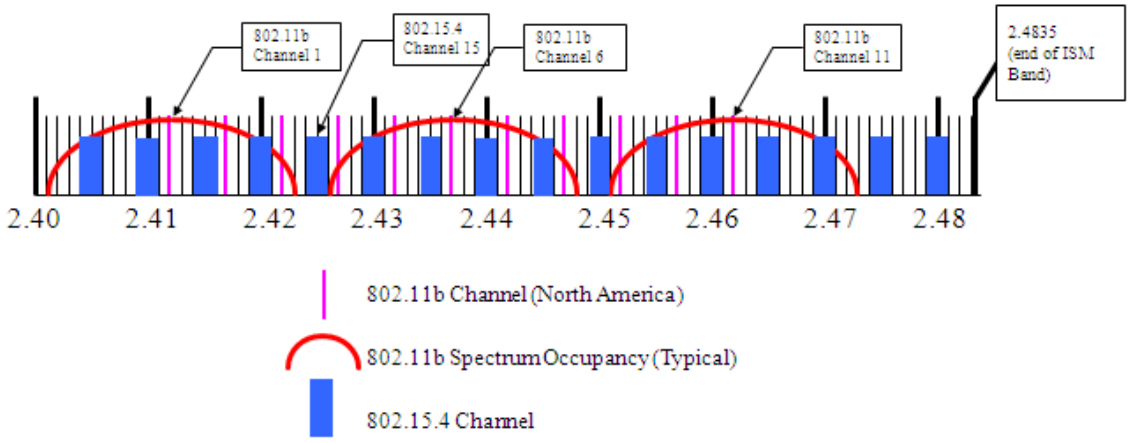

Figura 3– Coexistência do Wi-fi e 802.15.4 no espectro de 2.4GHz [3]

Além disso, a estrutura de uma rede seguindo o padrão 802.15.4 pode ter dois tipos de dispositivos:

- **Dispositivos de função reduzida –** *Reduced Function Device (RFD):* dispositivos que não precisam desempenhar grandes papéis na rede, apenas tendo como função repassar seus dados a dispositivos acima deles. Não são capazes de descobrir novos caminhos, nem de formar redes. Conseguem apenas se comunicar com FFDs.
- **Dispositivos de função completa** – *Full Function Device (FFD):* estes dispositivos podem controlar uma rede, aceitar novos nós e até formar uma nova rede. Podem se comunicar com outros FFDs e também com os RFDs.

A topologia de uma rede 802.15.4 pode ter duas configurações (figura 4):

- **Topologia em Estrela**: existe um dispositivo responsável por controlar toda a rede, chamado de coordenador PAN. É ele quem inicia e gerencia toda a rede, recebe os dados de outros dispositivos e realiza roteamento dos nós. Ele precisa ser um FFD. Exemplos de aplicações que utilizam essa topologia são automação doméstica, periféricos ligados a um computador pessoal e *personal health care.*
- **Topologia Ponto-a-ponto**: neste tipo de comunicação todos os nós podem se comunicar e ajudar a gerenciar a rede. Para isto basta que os nós sejam FFDs. Assim um FFD pode se comunicar e ajudar a gerenciar toda a rede. Este tipo de topologia permite que as redes sejam *ad-hoc* (dispositivo é incorporado à rede por qualquer elemento desta). As redes *mesh* utilizam a topologia ponto-a-ponto para que uma mensagem possa percorrer vários nós até chegar ao seu destino (esta função deve ser implementada na camada superior, no caso o protocolo PopNet). É a topologia mais usada em aplicações relacionadas a

monitoramento e controle industrial, redes de sensores sem fio, rastreamento, dentre outras.

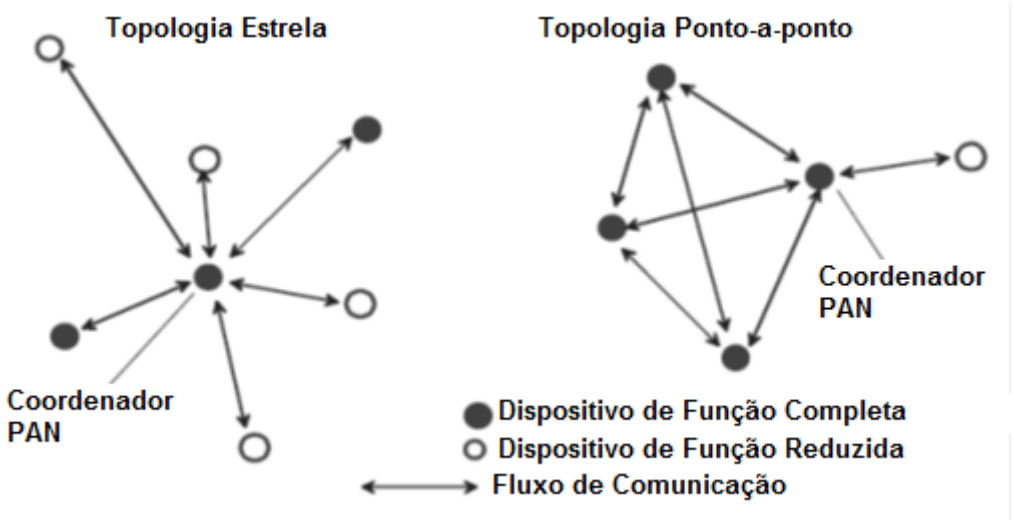

Figura 4 – Diagrama das topologias em estrela e ponto-a-ponto [4]

Quando se fala em endereçamento em uma rede 802.15.4, refere-se a oferecer recursos para definir a fonte e o destino de um pacote. O protocolo 802.15.4 trabalha com três tipos de identificadores [3]:

- **Endereço MAC**: endereço de 64 bits, sendo um identificador único globalmente. Os primeiros 24 bits do endereço são fornecidos ao fabricante do dispositivo pela IEEE.
- **PAN ID**: identificador de 16 bits utilizado para separar diferentes tipos de rede em um mesmo local. Com este identificador diferente, duas redes podem utilizar o mesmo canal para comunicação sem que haja interferências.
- **Short Address**: identificador de 16 bits também chamado de endereço do nó. É utilizado para identificar um nó dentro de uma rede. Tal endereço está ligado ao PAN ID. Com esse identificador, é possível caracterizar o nó, de modo a permitir uma comunicação direta.

A formação de uma rede é função de camadas superiores ao padrão 802.15.4 [4]. No entanto, o princípio básico de uma topologia em estrela é um dispositivo FFD ser ativado e a partir disto estabelecer sua própria rede e se tornar o coordenador PAN. Para que não haja interferências com outras redes em estrela já existentes, o coordenador PAN gera um PAN ID ainda não usado e permite então que novos dispositivos FFD e RFD façam parte da rede.

Na topologia ponto-a-ponto todos os dispositivos são capazes de se comunicar entre si, desde que estejam ao alcance do sinal. Neste tipo de rede, um RFD é o

dispositivo final e deve se conectar a um FFD, que será um coordenador de rede. Qualquer FFD poderá atuar como coordenador e sincronizar serviços da rede entre mais dispositivos finais e coordenadores. Ainda existe a necessidade de um FFD atuando como o coordenador PAN (normalmente o primeiro dispositivo a ocupar o canal), tendo também uma maior quantia de recursos computacionais para gerenciar toda a rede. Ele cria um novo PAN ID, que é fornecido a qualquer dispositivo que tente se conectar a rede. O dispositivo que tentar entrar na rede deverá realizar um pedido ao coordenador PAN. Assim que o coordenador aceitar a entrada deste novo dispositivo, o novo nó guardará informações referentes ao coordenador PAN e iniciará transmissões de pacotes.

É possível a criação de uma árvore de múltiplos clusters (figura 5). Nela um coordenador PAN já existente pode enviar informações a um nó para que este se torne um novo coordenador PAN de um cluster adjacente, permitindo a comunicação entre vários clusters em uma rede e melhorando o alcance da rede. No entanto, o aumento no tamanho da rede sempre aumenta o tempo de latência para a entrega de uma mensagem.

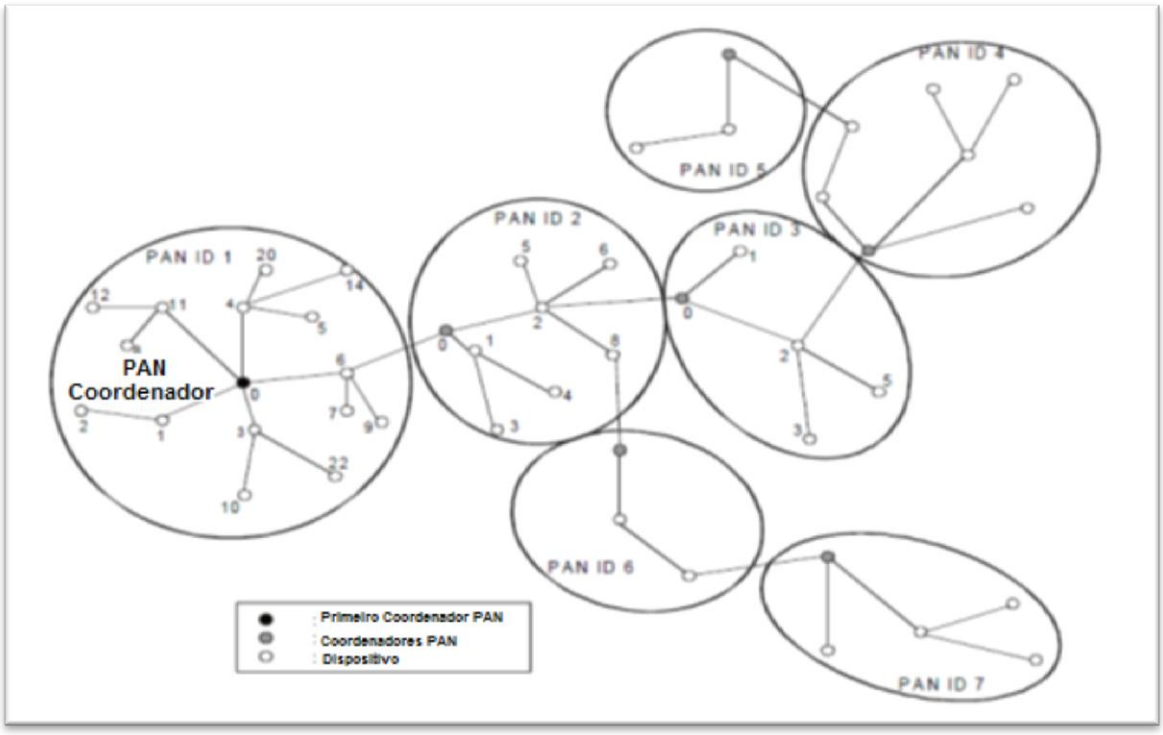

Figura 5 – Rede com múltiplos clusters. [4]

2.1.1 Arquitetura

O modelo de arquitetura do protocolo 802.15.4 baseado no modelo OSI pode ser visto na figura abaixo (figura 6):

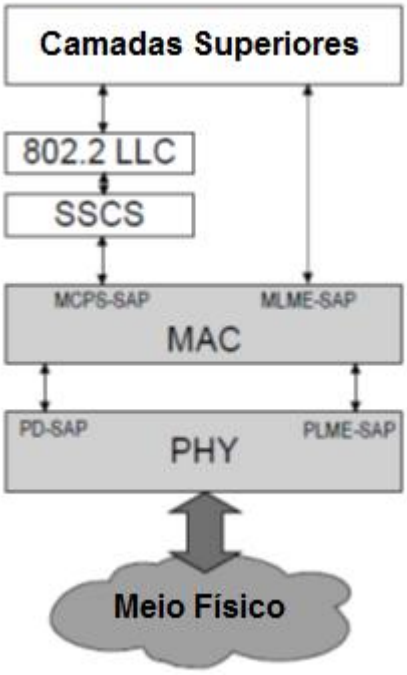

Figura 6 – Arquitetura do protocolo 802.15.4 [4]

Cada bloco da figura representa uma camada e é responsável por uma parte do padrão, sempre fornecendo serviços para as camadas superiores. A camada PHY contém informações e normas para o envio do sinal em radiofrequência e a camada MAC estabelece como acessar a camada física para a transmissão de informações.

As camadas superiores são responsáveis pela configuração da rede, manipulação, roteamento de mensagens e por adicionar novos serviços. O PopNet pode ser compreendido como uma dessas camadas superiores. As camadas superiores podem ter acesso à camada MAC diretamente, ou utilizando ainda as camadas LLC (*logical link control*) e SSCS (*service specific convergence sublayer*).

2.1.2 Camada PHY

A camada física no protocolo IEEE 802.15.4 define como deve ser a comunicação através de um canal sem fio. Ela define as bandas ISM, descritas anteriormente, que são caracterizadas pelas frequências 868MHz, 915MHz e 2.4GHz.

Existe apenas um canal na frequência de 868MHz, ao passo que há 10 canais em 915MHz. A frequência de 2.4GHz é a que possui maior quantidade de canais, dezesseis ao todo. Quando se fala de taxas de transmissão dos dados, a frequência de 868MHz atinge apenas 20kbps. A frequência de 868Mhz parece ser muito desfavorável frente às outras, mas sua menor frequência produz menores perdas de propagação e com isso o sinal pode atingir áreas maiores; e sua baixa taxa de transmissão fornece maior sensibilidade e também maior área de cobertura. A frequência de 2.4GHz por oferecer maior taxa de transmissão garante menor tempo de latência para entrega dos dados e consequentemente maior vazão de dados [5]. A divisão do espectro de frequência nas bandas ISM pode ser visto na figura 7.

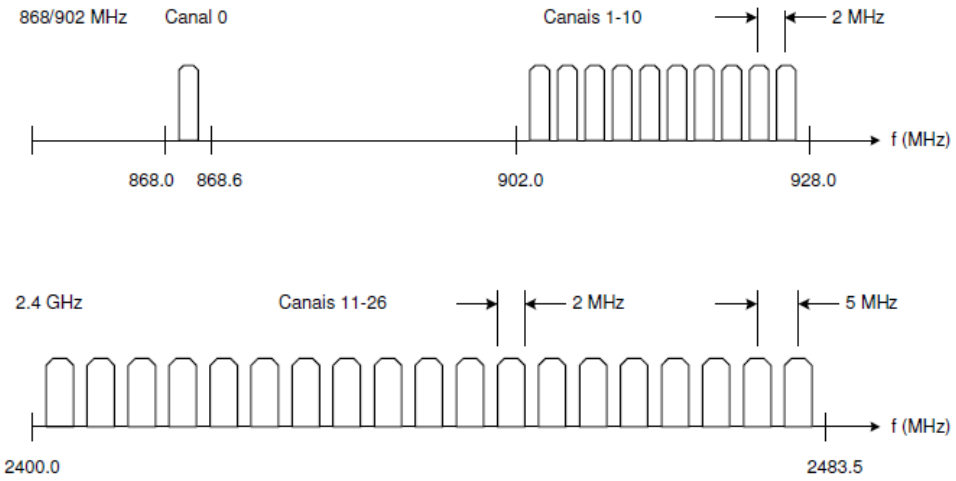

Figura 7 – Divisão dos espectros de frequência no padrão 802.15.4
Fonte: De TORRES DOS SANTOS, SERGIO - Redes de Sensores sem Fio em Monitoramento e Controle [Rio de Janeiro] 2007.

A camada PHY é a camada responsável por ligar e desligar o rádio transmissor quando necessário. É a camada PHY que realiza a transmissão e recepção de pacotes no meio físico, seleção do canal de frequência a ser utilizado, detecção de energia no canal (ED), monitora a qualidade do sinal (LQI) e avalia os canais livres através do CCA (*Clear Channel Assessment*).

A detecção de energia do receptor (ED) é um atributo utilizado pela camada de rede para selecionar o canal apropriado para a transmissão e recepção. Ele se baseia em uma estimativa da potência do sinal recebido no canal utilizado.

O monitoramento da qualidade do sinal é realizado pela medida LQI (*Link Quality Indication*) que caracteriza a intensidade e qualidade do pacote recebido. O LQI é obtido através de uma estimativa da relação sinal/ruído e da detecção de

energia do canal (ED). O LQI é armazenado em um número inteiro de 8 bits. O valor mais alto deste número representa a maior qualidade do sinal.

A avaliação dos canais livres (CCA) utiliza-se de um dos três seguintes métodos:

- Energia acima do limiar: se o nível de energia estiver acima da medida de detecção de energia do receptor (ED), o canal está ocupado.
- Detecção da portadora: se o CCA detectar uma portadora com as características do IEE 802.15.4, o canal está ocupado.
- Detecção da portada e da energia do limiar: o canal está ocupado se detectar tanto a portadora como a energia acima do limiar.

2.1.3 Camada MAC

A camada MAC é responsável por todo acesso à camada física para a transmissão e recepção de dados. Sua principal função é gerar e sincronizar pacotes utilizados para formação da rede, denominados *beacons*. O *beacon* é um pacote enviado por um nó na rede que contém seus identificadores (PAN ID, endereço do nó, MAC *Address*). Assim, permite a sincronização correta do nó com a rede desejada. Além disso, a camada é responsável por oferecer segurança aos dispositivos, empregar o mecanismo de CSMA-CA para evitar colisões de canais e realizar as associações e dissociações da rede PAN.

2.2 PopNet

PopNet é um protocolo de rede criado pela empresa San Juan Software baseado no padrão IEEE 802.15.4. Seu lema "*The easy, economical wireless sensor network*" [2] revela que o protocolo tem por objetivo fornecer um ambiente de desenvolvimento de fácil utilização para redes de sensores sem fio. O protocolo se situa acima do padrão IEEE 802.15.4 como pode ser visto na figura 8. O PopNet utiliza-se muito pouco da camada MAC do padrão 802.15.4, possuindo um "mini-MAC" bem definido.

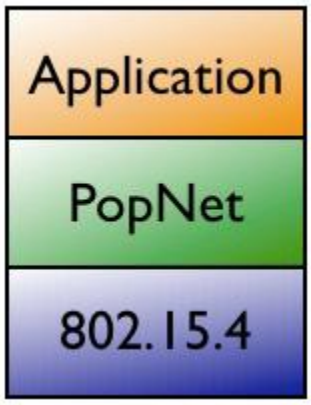

Figura 8 – PopNet na camada superior ao protocolo 802.15.4 [2]

Assim, o protocolo PopNet estende o padrão 802.15.4 adicionando novas funcionalidades:

- *Multi-hop* através da rede *mesh*, aumentando o alcance da rede.
- Suporte a descoberta de rede, dispositivos e serviços.
- Melhor segurança, com autenticação dupla e criptografia AES-128bit.
- Suporte a diversos dispositivos populares no mercado, oferecendo suporte completo a alguns tipos de rádios e microcontroladores.
- *Upgrades Over-the-air* (a aplicação pode ser atualizada e modificada com a rede sem fio já formada)
- Serviços de entrada e saída, *timers*, alocação de memória.

Por ser um protocolo construído sob as especificações IEEE 802.15.4, o PopNet é compatível com os pacotes IEEE 802.15.4 MAC e pode coexistir e operar perto de redes ZigBee e Wi-fi sem danos.

O protocolo PopNet possui diversas características interessantes:

- Interface simples, escrita em Linguagem C.
- Baseado totalmente em redes *Mesh*.
- Demonstrações de aplicações inclusas, facilitando o aprendizado.
- Protocolo flexível, que suporta qualquer aplicação sem fio.
- Segurança AES 128-bit para proteger os dados e a rede.
- Suporte a baixo consumo de energia.
- Utilização de poucos recursos. Uma aplicação simples em PopNet é capaz de ser executada em microcontroladores de 8 bits, com 1K de memória RAM e 16K de memória Flash. Mas também é capaz de utilizar recursos de microcontroladores mais poderosos.

Uma importante característica do protocolo PopNet é a sua grande integração com os sistemas da Freescale. Isto fica bem evidenciado uma vez que para o desenvolvimento de projetos baseados no PopNet, é necessária a instalação do programa Freescale BeeKit. O PopNet instala *codebases* no BeeKit, prontas para plataformas 802.15.4 da Freescale. Além disso, a edição, compilação e depuração das aplicações em PopNet são realizadas na IDE CodeWarrior, também da Freescale.

Uma rede PopNet se caracteriza por ser uma rede *mesh*. Uma rede *mesh* provê uma melhor utilização da largura de banda através do roteamento dos nós e permite maior alcance que outros tipos de rede, como a topologia estrela. Além disso, existe a descoberta automática de rotas, facilitando o desenvolvimento da aplicação e garantindo mais opções para um pacote atingir o nó destino. Para entender este conceito, considere a figura 9: o nó 1 deseja se comunicar com o nó 3. No entanto, o nó 3 está fora do alcance de sinal do nó 1. Para que seja possível a comunicação e troca de dados entre os nós, utiliza-se um nó intermediário que possa ser utilizado para uma "rota" de transmissão de dados, no caso o nó 2. Este é o conceito de redes *mesh*: cada nó da rede pode ser um receptor e também um transmissor de dados.

Agora observe o segundo esquema da mesma figura: existe uma barreira que impede a propagação do sinal do nó 1 e com isso a comunicação com o nó 2. A rede *mesh* deve então buscar uma rota alternativa para que o nó 1 continue se comunicando com o nó 3. Uma das qualidades do protocolo PopNet é a capacidade de automaticamente detectar o problema e traçar uma nova rota para o sinal. O que se observa, portanto, é que todos os nós em uma rede PopNet podem formar e entrar em uma rede, além de rotear pacotes.

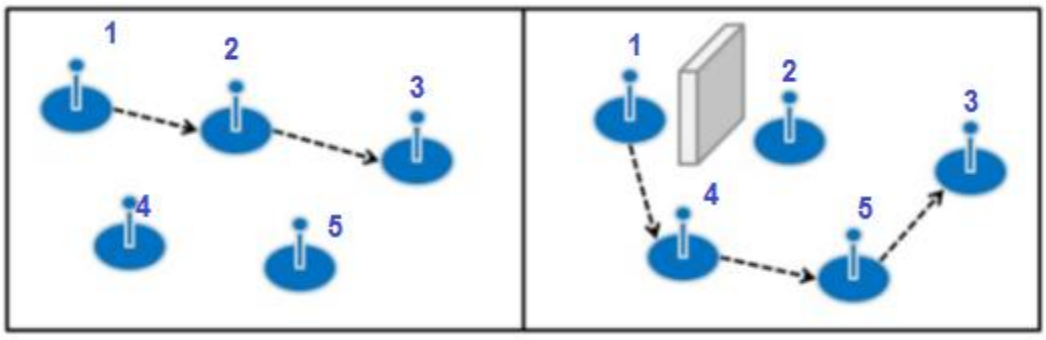

Figura 9 – Rede PopNet Mesh traça nova rota automaticamente [2]

Uma rede *mesh* utilizando o protocolo PopNet funciona corretamente com algumas dezenas de nós, mas teoricamente pode suportar até 64.000 nós. Além disso, a rede é capaz de realizar a comunicação entre dois nós utilizando até 50 nós intermediários.

2.2.1 Arquitetura

A arquitetura do protocolo PopNet utiliza-se do conceito de multitarefa. Assim, o protocolo possui um *mini-kernel* multitarefa com serviços relacionados a gerenciar a rede, mas também serviços para interagir com recursos presentes nos dispositivos, tais como LEDs, botões, LCD, *timers*, interface serial. A figura 10mostra a arquitetura do protocolo:

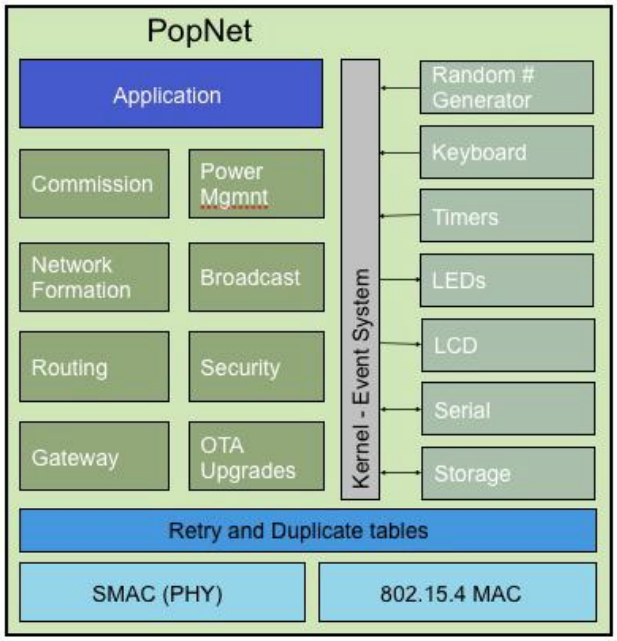

Figura 10 – Arquitetura do Protocolo PopNet [2]

O lado direito da figura 10 revela a comunicação do protocolo com os periféricos do dispositivo. Isto é feito através dos serviços BSP (Board Support Package). Tais serviços podem ser encontrados nos arquivos *PopBsp.c* e *PopBsp.h*, presentes em qualquer projeto criado no PopNet.

Já o lado esquerdo da figura trata dos serviços oferecidos para gerenciar a rede, tais como segurança, roteamento, gateway (comunicação com uma CPU ou PC), gerenciamento de energia, criação da rede.

Por fim, temos a aplicação no topo da estrutura, que deve ser desenvolvida pelo usuário, com um código abrangendo as funções necessárias para o correto funcionamento.

O desenvolvimento de uma aplicação em cima do protocolo PopNet é baseado em duas funções:

- *void PopAppTaskInit (void);*

- *void PopAppTaskEventLoop (popEvtId_t iEvtId, sPopEvtData_t sEvtData);*

A primeira função pode ser comparada à função *main ()* de qualquer programa escrito em C. Ela é responsável por inicializações de hardware ainda não feitas nos serviços BSP, tais como inicializar sensores, conversores A/D, interface serial, dentre outros. É nessa função que se inicia e se forma a rede.

A segunda função se assemelha ao loop principal em aplicações embarcadas existentes. Dentro desta função existem várias tarefas que são executadas quando eventos relacionados ocorrem na aplicação. Eventos são acontecimentos no sistema: um timer que chegou ao fim de sua contagem, um botão pressionado, dados chegando através da porta serial ou do rádio, dentre outros. É aquilo que o protocolo ou uma atividade externa inicia. O *kernel* do PopNet caracteriza-se por ser um kernel multitarefa não-preempitivo (figura 11). O termo não-preempitivo neste caso significa que, a menos que ocorra uma interrupção na aplicação, a aplicação continuará sendo executada.

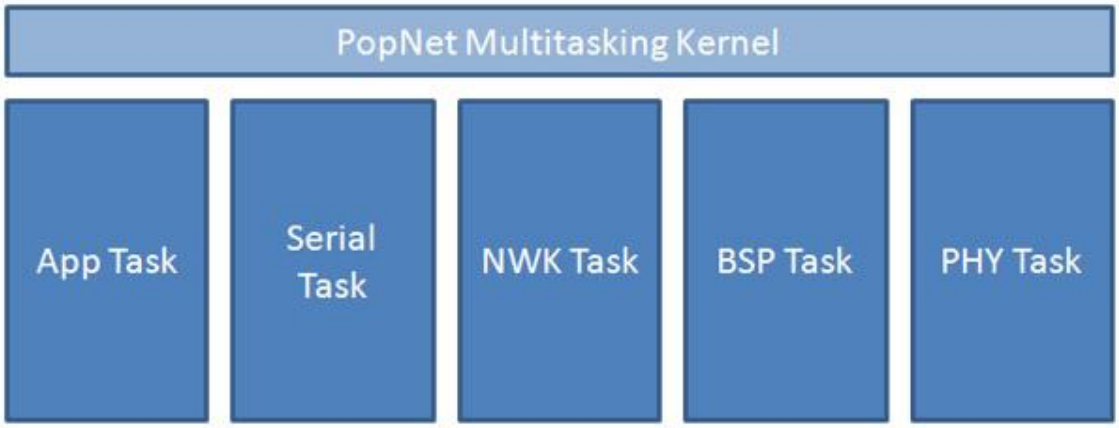

Figura 11 – Tarefas gerenciadas pelo kernel do Popnet [2]

Assim, quando um evento ocorre, a aplicação roda a tarefa relacionada a tal evento. É importante ressaltar que no protocolo PopNet não é possível definir prioridades no evento. Executa a tarefa aquele evento que ocorrer primeiro. Assim que terminar a tarefa, a aplicação sai da função *PopAppTaskEventLoop* e volta ao *kernel* do PopNet esperando novas tarefas a serem realizadas. A figura 12 ilustra os eventos tendo acesso à função *PopAppTaskEventLoop*:

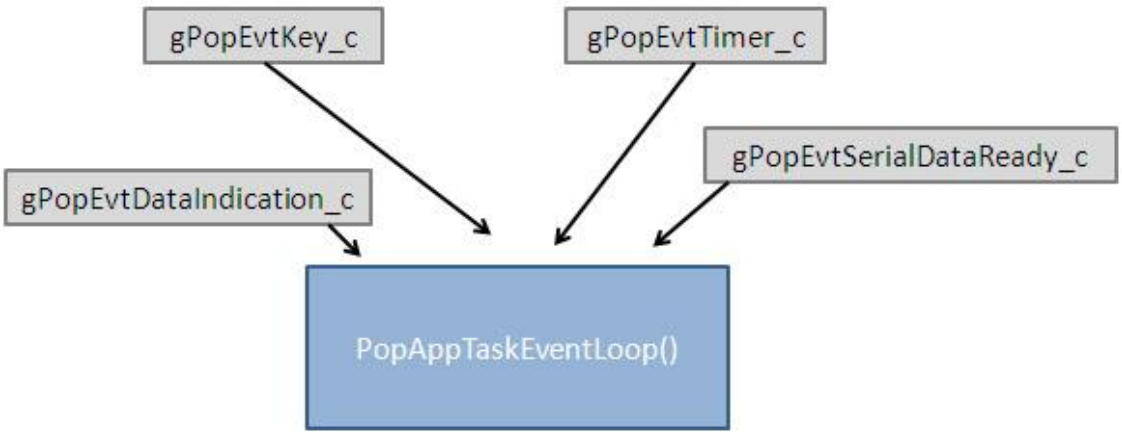

Figura 12 – Eventos gerenciados pela função PopAppTaskEventLoop( ) [2]

Além dos eventos, existem os comandos no protocolo que são iniciados pela aplicação. Podem terminar imediatamente, como alocação de memória, ou demorar algum tempo, como comandos seriais ou de rede. Nesse caso, o protocolo trata de fornecer um evento de confirmação.

Falta detalhar os métodos para o envio e o recebimento de dados através da comunicação sem fio. Para isto, o protocolo utiliza-se do seu conceito de utilizar métodos e eventos. Existe uma função denominada *PopNwkDataRequest* ( ) que é responsável por transmitir os pacotes através da rede(figura 13). E para indicar que um nó está recebendo dados, a aplicação recebe um evento denominado *gPopEvtDataIndication_c*.

Dessa forma, existe sempre uma função responsável por enviar os dados utilizando uma estrutura denominada *sPopNwkDataRequest_t*. O termo *"data request"* é amplamente usado em redes do tipo 802.15.4 e significa que a aplicação está pedindo para enviar dados. Este estrutura possui em seus campos informações como o nó destino, opções de segurança e propagação do dado e por fim o tamanho e o dado em si. O protocolo permite o envio de até 108 bytes de dados propriamente ditos, dependendo das opções de segurança.

Já para o caso de recebimento de dados, o método responsável para isso é o *PopAppDataIndication*, que recebe como parâmetro uma estrutura denominada *sPopNwkDataIndication_t*. O termo *"data indication"* é para indicar que a aplicação está recebendo dados. Este método é executado quando a aplicação recebe o evento *gPopEvtDataIndication_c*. É possível então recuperar o dado, seu tamanho, de que nó ele veio, a forma como foi transmitido, dentre outras informações.

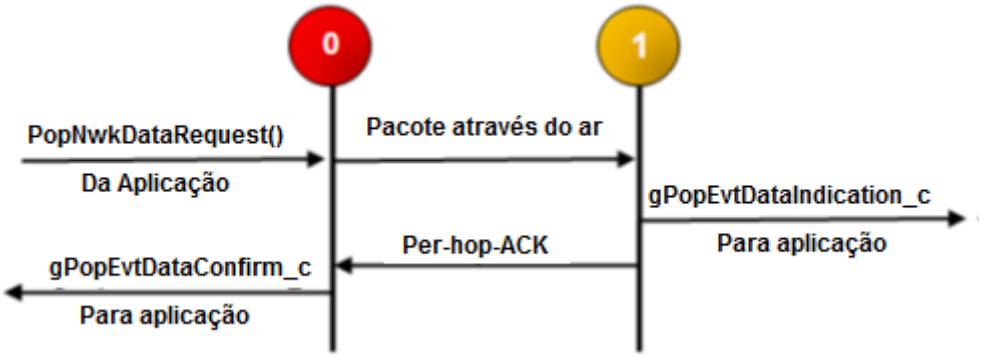

Figura 13 – Método de envio de dados no protocolo PopNet [2]

2.2.2 Endereçamento

O protocolo PopNet adiciona novos identificadores aos já definidos pelo padrão IEEE 802.15.4 para auxiliar no endereçamento da rede:

- **Manufacturer ID**: é um identificador de 16 bits definido pela *San Juan Software* para cada fabricante, para auxiliar os nós a encontrar a rede certa.
- **Application ID**: é um identificador de 16 bits para ser usado dentro de uma determinada aplicação, a fim de definir a função do nó naquela aplicação. Por exemplo, em um sistema de ligar luzes através de um comando sem fio, o nó pode ser o *switch* ou a luz. Nesse caso, cada opção recebe um ID diferente.
- **Hardware ID**: identificador de 8 bits que ajuda a identificar diferentes versões do hardware do sistema.

2.2.3 Unibroadcast

O envio de dados através de uma rede sem fio normalmente ocorre através de dois métodos: *unicast* e *broadcast*(figura 14). O método *unicast* realiza o envio de dados de um nó para outro nó definido, utilizando algum dos identificadores de 16 bits presentes nos protocolos 802.15.4 e PopNet. Já no método *broadcast* um nó envia pacotes por uma porção ou toda uma rede, atingindo vários nós. Cada nó que recebe uma transmissão por *broadcast* retransmite o pacote três vezes. Como é necessário que o nó armazene o pacote para retransmiti-lo, consumindo memória do *buffer*, não é recomendável utilizar este método frequentemente. Existe ainda o método denominado *multicast*, que o envio é para um determinado grupo de nós. O protocolo

PopNet não oferece esse método, mas é possível algo semelhante atribuindo um grupo de identificadores particular e estabelecer o envio somente para aquele identificador.

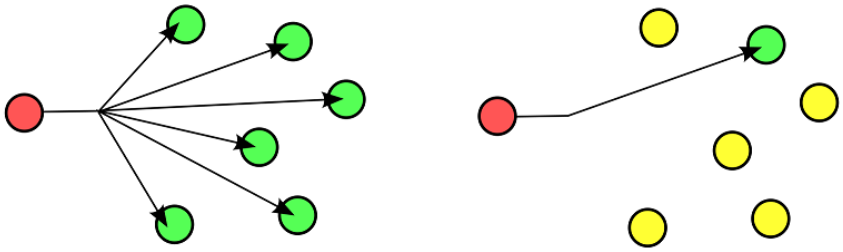

Figura 14 – Métodos de envio Broadcast e Unicast
Fonte:http://blogs.law.harvard.edu/niftyc/archives/date/2009/09

O protocolo PopNet introduz um novo método de envio de dados: unibroadcast. O termo define bem as suas características: o envio de dados de um nó para outro nó definido utilizando o envio *broadcast*. O PopNet usa este método principalmente para descobrir rotas para envio de pacotes em uma rede, o que permite o envio de dados ao mesmo tempo em que a rota a ser percorrida está sendo descoberta.

2.2.4 Uso de Energia

O padrão IEEE 802.15.4 já estabelece que os rádios utilizados em hardwares para aplicações sobre tal protocolo façam uso de pouca energia. No entanto, em determinadas aplicações o envio de informação através do rádio nem sempre é necessário durante todo o tempo. Por isso, o protocolo PopNet oferece um meio de colocar um determinado nó em modo de baixo consumo de energia, onde o rádio e o microcontrolador são colocados em uma espécie de estado de espera.

Para utilizar este modo, basta chamar a função *PopPwrSleep* (*popPwrWakeUpon_t iWakeUpOn, popPwrTimeOut_t iTimeOutMs*) em qualquer momento na aplicação. O dispositivo pode voltar ao seu funcionamento normal através da ocorrência de uma interrupção, ou após determinado tempo. Quando a placa volta ao modo normal, ocorre um evento *gPopEvtPwrWakeUp_c* que deve ser tratado pela aplicação na função *PopAppTaskEventLoop.*

2.2.5 Atualizações "*over-the-air*"

A partir da versão 2.0 do protocolo, surgiu um novo recurso extremamente interessante para aplicações comercias. Chamado de "*Over-the-air upgrades*", este

recurso permite que uma aplicação que esteja sendo executada em uma rede possa ser atualizada em tempo real. É apenas necessário o envio de uma imagem contendo a nova aplicação. A aplicação atual então é capaz de escolher se muda para a sua nova versão. Para que isto ocorra, o protocolo utiliza-se dos identificadores já definidos para que as atualizações possam ir para somente um nó, um determinado grupo ou para toda rede.

Durante a atualização, a rede pode continuar a funcionar e caso ocorra algum problema na atualização, voltar para a aplicação válida anterior.

2.3 I²C – *Inter-Integraded Circuit*

O I²C é um protocolo de comunicação originalmente desenvolvido pela *Philips Semiconductors* na década de 80[6], com o objetivo de reduzir custos em aparelhos de TV. Atualmente é amplamente difundido como uma opção para comunicação serial entre dispositivos diversos. Dentre estes dispositivos podemos citar microprocessadores, microcontroladores, memórias, conversores, sensores, dentre outros.

A interface I²C é uma interface muito simples, por utilizar somente duas linhas de comunicação bidirecionais: o *clock* (SCL) e dados (SCA). Além disso, permite o endereçamento de múltiplos dispositivos (teoricamente até 127 dispositivos) no mesmo barramento. Cada dispositivo recebe um endereço único. Atingem taxas de transmissão desde 100kb/s até 3.4MB/s no modo *High-Speed*.

Para a completa compreensão do protocolo, é necessário entender alguns termos [7]:

- Transmissor: é o dispositivo que envia os dados através do barramento.
- Receptor: é o dispositivo que recebe os dados através do barramento
- *Master* (mestre): é o dispositivo responsável por iniciar e concluir a comunicação, além de gerar o sinal de clock.
- *Slave* (escravo): é o dispositivo controlado pelo mestre.
- *Multi-master* (multi-mestre): o sistema pode possuir mais de um dispositivo mestre. Com isso, pode ocorrer de mais de um dispositivo tentar controlar o barramento em dado instante.
- Arbitragem: procedimento para garantir que, em um sistema multi-mestre, enquanto um dispositivo mestre esteja utilizando o barramento, os outros não poderão interferir no processo.

Um exemplo simples de utilização deste protocolo é a ideia utilizada neste projeto: a comunicação entre o microcontrolador e uma memória EEPROM. Neste caso, o microcontrolador se comporta como o dispositivo mestre e a memória como o dispositivo escravo. O microcontrolador pode ser tanto o transmissor (quando envia dado a serem gravados na memória) como receptor (quando realiza leitura da memória).

De posse destas informações e observando a figura 15, pode-se compreender o funcionamento e conhecer melhor os detalhes do protocolo I²C [8]:

- A leitura da linha de dados (SDA) somente ocorre quando a linha do clock (SCL) encontra-se em nível lógico “1”.
- Para se alterar o nível lógico da linha de dados (SDA) o sinal da linha do clock (SCL) deve estar em “0”.
- Quando o barramento está livre, ambas as linhas encontram-se em nível lógico “1”.

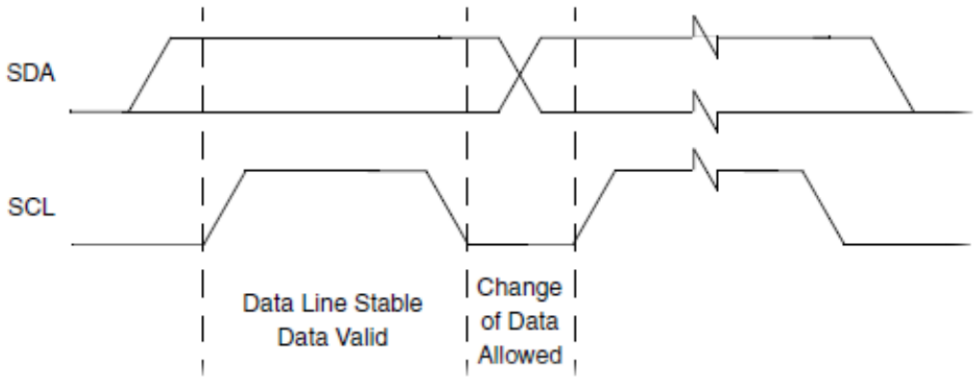

Figura 15 – Momentos de barramento livre e para escrita de dados[7]

Para indicar o início e término de uma transmissão, ocorrem duas condições que violam as regras descritas acima, chamadas de condição *START* e *STOP* (figura 16). A condição *START* que indica o início da transmissão para os dispositivos ocorre quando se força a linha SDA de nível lógico “1” para “0” durante a fase alta do *clock*. Para a condição *STOP*, deve-se forçar o nível lógico da linha SDA de “0” para “1” também durante a fase alta do *clock*.

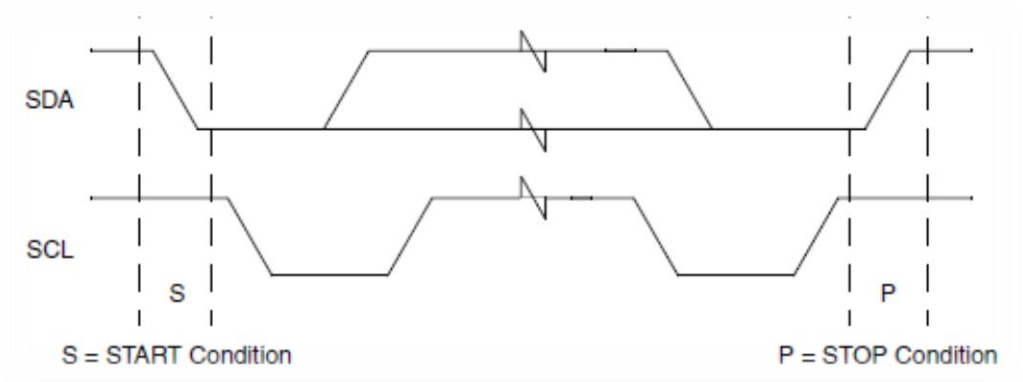

Figura 16 – Geração de condições START e STOP[7]

A transferência de um byte de dados ocorre após a condição de *START* e inicia pelo bit mais significativo. Após o último bit (bit menos significativo), o receptor deve gerar um bit de reconhecimento (*ACK – acknowledge*) em nível lógico “0” na linha SDA antes do nono pulso de *clock* da linha SCL. Este byte de dados pode representar o byte de controle do sistema, um byte de endereçamento do dispositivo ou então um byte de dados propriamente dito.

2.3.1 Byte de Controle

O byte de controle sempre ocorre após uma condição de *START*. Sua função é armazenar informações a respeito do tipo de dispositivo escravo a ser acessado (no projeto uma memória EEPROM), o endereço deste dispositivo entre vários que podem estar acessando o mesmo barramento e se está ocorrendo uma operação de escrita ou leitura. Assim, os 4 primeiros bits armazenam o tipo do dispositivo, seguindo de 3 bits de endereçamento e 1 bit para informar se é uma operação de escrita ou leitura (“1” para leitura, “0” para escrita) (figura 17).

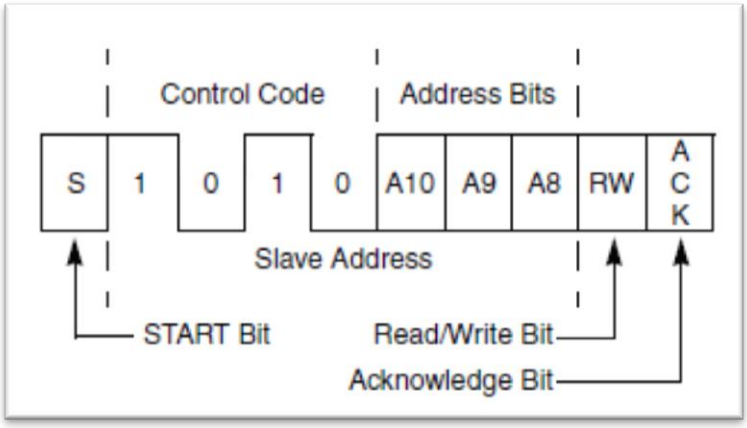

Figura 17 – Estrutura do byte de controle [7]

2.3.2 Byte de Endereçamento

O byte de endereçamento é responsável por determinar endereços físicos dentro do dispositivo a ser acessado. Assim, para o caso de uma memória EEPROM, ele é responsável por armazenar informações sobre em que posição exata da memória a informação está sendo lida ou escrita. Como alguns dispositivos possuem um grande número de endereços internos, podem-se utilizar dois bytes de endereçamento: o primeiro byte armazena a parte alta do endereço, e o segundo byte a parte baixa.

Após os envios destes bytes (sempre seguidos do bit de *acknowledge)* ocorre o envio do byte de dados (figura 18).

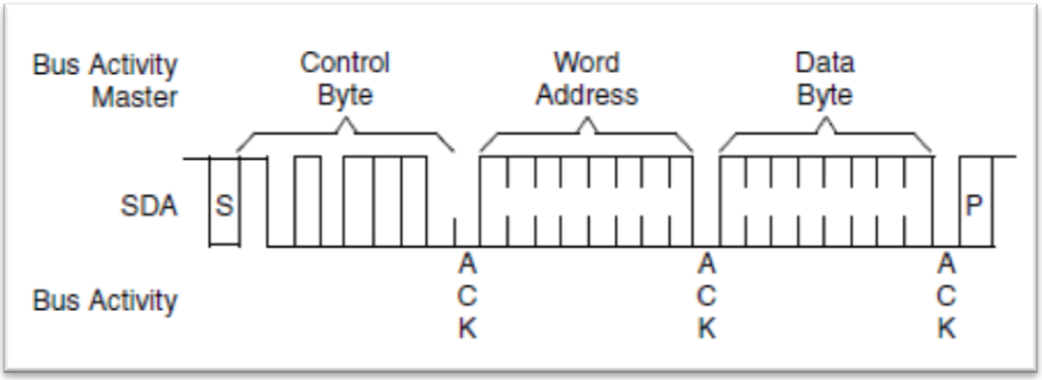

Figura 18 – Operação de escrita com endereço de 8 bits[7]

O protocolo I²C ainda conta com a possibilidade do uso de interrupções.

Todas as manipulações e geração de condições *START* e *STOP* e escrita e leitura dos dados devem ser realizadas pelo microcontrolador com o uso de registradores específicos para o módulo I²C. Maiores detalhes sobre como isto é feito podem ser encontrados no capítulo 3.5.4.

CAPÍTULO 3 - DESENVOLVIMENTO

O projeto descrito visa criar uma rede de sensores de temperatura com comunicação sem fio, aplicável a diversos ambientes e situações. O diferencial de tal projeto está na utilização de um protocolo ainda pouco conhecido, o PopNet. Para tal projeto, existem dois componentes físicos básicos:

Coordenador – Hardware que possui a função de *gateway*, comunicando-se serialmente com um computador através da porta USB, recebendo dados de outros dispositivos sem fio da rede, além de possuir um sensor de temperatura para título de comparação com dados recebidos.

Placas de Aquisição Remota – Dispositivos dotados de sensores de temperatura LM35, memória EEPROM e comunicação sem fio. Armazenam a leitura dos sensores na memória, para posterior transmissão através da comunicação sem fio.

A figura 19 representa o sistema:

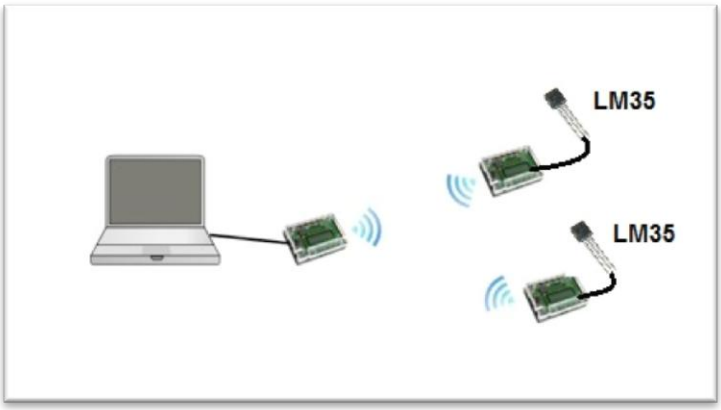

Figura 19 – Rede de sensores de temperatura

3.1 Coordenador

O projeto conta com um coordenador, responsável por ser o elemento principal da rede. É o coordenador que dá início à formação da rede, além de ser o *gateway* do sistema, já que se comunica com um computador através da porta USB, recebendo os dados dos outros sensores remotos. Ele estará sempre conectado a um computador. Para realizar esta função, foi utilizada uma placa produzida pela Freescale: 13213-SRB (Sensor Reference Board) (figura 20).

Figura 20 - Freescale 13213-SRB [9]

A placa 13213-SRB foi escolhida por ser um dispositivo já voltado para aplicações sobre o protocolo 802.15.4. Além disso, é uma das placas suportadas nativamente pelo protocolo PopNet, o que facilita muito sua utilização. Ela conta com [9]:

- Um transmissor de radiofrequência que opera na faixa de 2,4GHz, acoplado ao microcontrolador MC13213, baseado na família HCS08 de microcontroladores.
- Porta USB 2.0, facilitando a comunicação com um computador, além de poder fornecer energia para a placa.
- Quatro botões, além de um botão para reset e quatro LEDs.
- Uma antena F acoplada ao desenho da placa que oferece bom desempenho para transmissões sem fio.
- Conexão BDM para a sua programação (será falado mais adiante).
- Entrada para fonte DC de 5-9V.
- Entrada para 2baterias AA (o que facilita sua mobilidade).

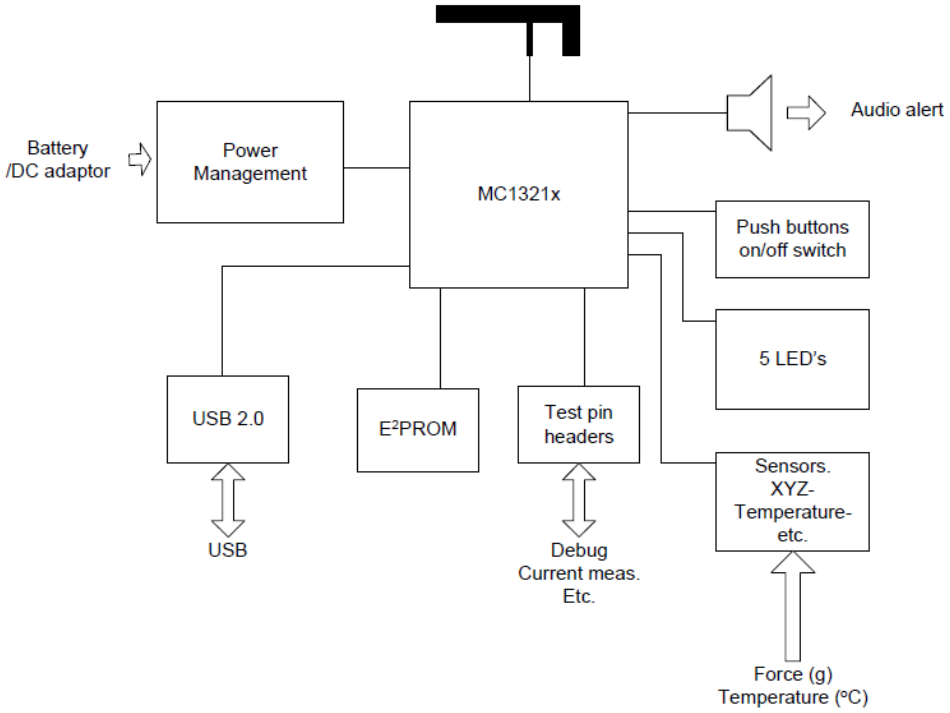

Figura 21 - Diagrama de blocos da placa 13213-SRB [9]

Além destes recursos, a placa 13213-SRB ainda conta com dispositivos acoplados a tal (figura 21). Ela possui uma EEPROM, modelo AT25HP512, conectada diretamente ao microcontrolador. Possui também um acelerômetro capacitivo, modelo MMA7260Q e um indicador de áudio (*buzzer*) controlado via software. Por fim, há um sensor de temperatura LM61B que é o que mais interessa ao projeto, já que serão feitas leituras diretamente dele.

3.1.1 LM61B

O LM61B, fabricado pela *National Semiconductor* [10] é um circuito integrado que funciona como um sensor de temperatura de precisão. Sua tensão de saída é linearmente proporcional à temperatura a ser medida em Celsius. A proporção de sua tensão de saída para o valor da temperatura é de +10mV/ºC e possui um offset DC de +600mV. Isto quer dizer que, à temperatura de 0ºC, a tensão de saída é de +600mV. A faixa de tensões de saída suportadas pelo sensor é de +300mV a +1600mV, o que fornece, portanto, faixas de temperatura de -30ºC a 100ºC.

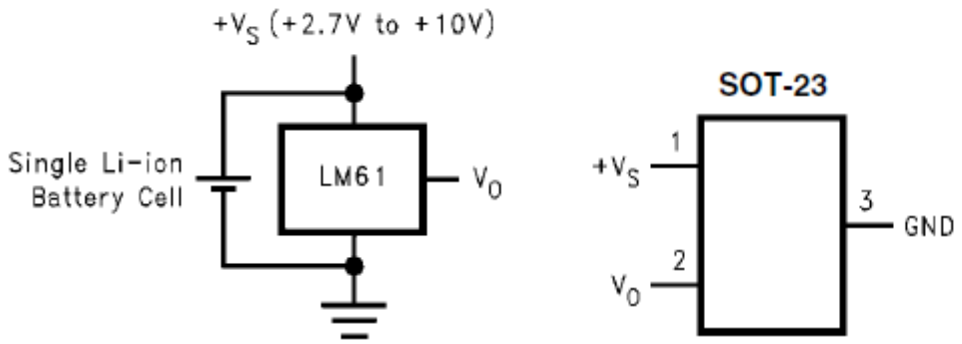

Figura 22 - Circuito Típico e visão superior do sensor LM61B [10]

Devido às propriedades materiais do sensor, ele fornece uma precisão de 3ºC quando a temperatura medida se encontra na faixa de -25ºC a 85ºC. A faixa de tensão de operação deste dispositivo pode variar entre +2.7V e 10V (figura 22). No caso da placa 13213-SRB, é fornecida diretamente pelo microcontrolador.

3.2 Placas de Aquisição Remota

As placas de aquisição remota (figura 23) serão os dispositivos responsáveis por realizar as medições de temperatura. Para tornar o sistema ainda mais portátil e flexível, a ideia é armazenar as medições em uma memória EEPROM até que estas informações possam ser recuperadas pelo coordenador. Isto permite que mesmo que as placas não estejam ao alcance do coordenador, elas possam continuar a realizar normalmente a monitoração de dados, até que possam entrar em contato com o coordenador.

As placas são dotadas também de um microcontrolador MC13213, que permite a transmissão sem fio utilizando a frequência de 2.4GHz com uma antena em F impressa diretamente na placa. No entanto, tais placas possuem menos recursos que as 1321-SRB. Elas não contam com botões, porta USB e nem um sensor de temperatura acoplado a placa. Possuem apenas um LED e a conexão BDM para gravação direta no chip. No entanto, contam com uma memória EEPROM, modelo AT24C256, que foi de grande importância para armazenar as leituras de temperatura.

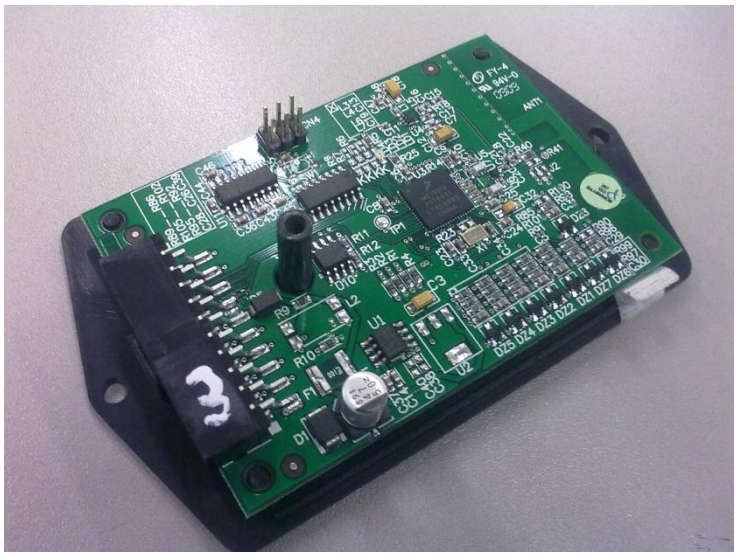

Figura 23–Placa de aquisição remota

A alimentação destas placas foi fornecida por uma fonte DC de 5V. Isto reduziu a mobilidade das mesmas. Isto reforça que as placas de aquisição remota têm como aplicação principal sistemas embarcados, como automóveis e caminhões.

Uma vez que as placas de aquisição remota não possuem sensores embarcados, isto permite que o usuário faça sua opção pelo sensor que lhe convier. Neste trabalho a solução encontrada foi a utilização de sensores de temperatura LM35, que são ligados ao conector de entrada da placa de aquisição remota. Com isso, foi possível realizar as medidas e enviá-las através do rádio transmissor e antenas presentes nas placas.

3.2.1 LM35

O LM35 é outro modelo de circuito integrado da *National Semiconductor* [11] com a função de sensor de temperatura. Seu funcionamento é muito semelhante ao modelo LM61, já que sua tensão de saída também é proporcional à temperatura em graus Celsius. Sua linearidade é de +10mV/ºC, com uma precisão de 0,5ºC à temperatura ambiente (cerca de 25ºC) e possui sensibilidade para medir temperaturas de -55º a +150ºC. Sua tensão de *off-set* é 0mV, o que significa que à temperatura de 0ºC, a tensão de saída será aproximadamente 0mV.

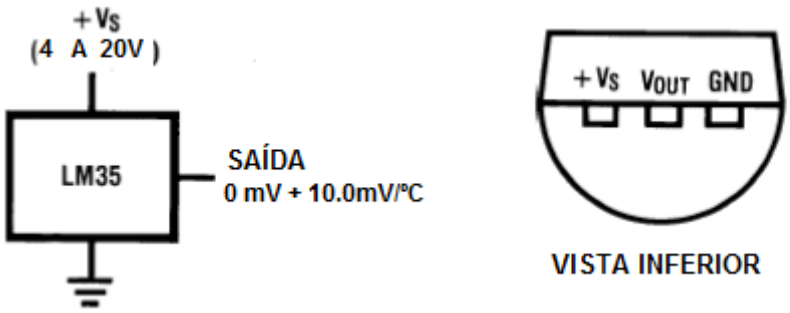

Figura 24 - Circuito típico e visão frontal do modelo adotado do sensor LM35[11]

Para que o circuito seja alimentado corretamente, deve-se fornecer uma tensão na faixa de 4 a 20 Volts (figura 24).

3.2.2 AT24C256

As placas de aquisição remota possuem um módulo de memória EEPROM modelo AT24C256, fabricado pela Atmel [12].

Memórias EEPROM (*Electrically Erasable Programmable Read-Only Memory*) têm como principal característica a possibilidade de se apagar seu conteúdo através de um comando elétrico enviado diretamente para a mesma. Além disto, permite que bytes individuais possam ser apagados e reprogramados sem que afete outras regiões da memória, ao contrário de memórias EPROM. Isto facilita a escrita e a leitura destas memórias, já que um microcontrolador pode ser o responsável por enviar comandos elétricos para sua manipulação.

O modelo AT24C256 possui os seguintes recursos:

- 256Kb de espaço, divididos em palavras de 8 bits cada, totalizando 32.768 palavras.
- Permite até quatro dispositivos compartilhando o mesmo barramento.
- Interface serial utilizando apenas dois fios bidirecionais. O modelo AT24C256 utiliza o protocolo de comunicação I^2C (figura 25).
- Alta durabilidade. Suporta mais de um milhão de ciclos de escrita e os dados podem ficar armazenados por até 40 anos.
- Suporta operação em temperaturas extremas. (de -55ºC a +125ºC)

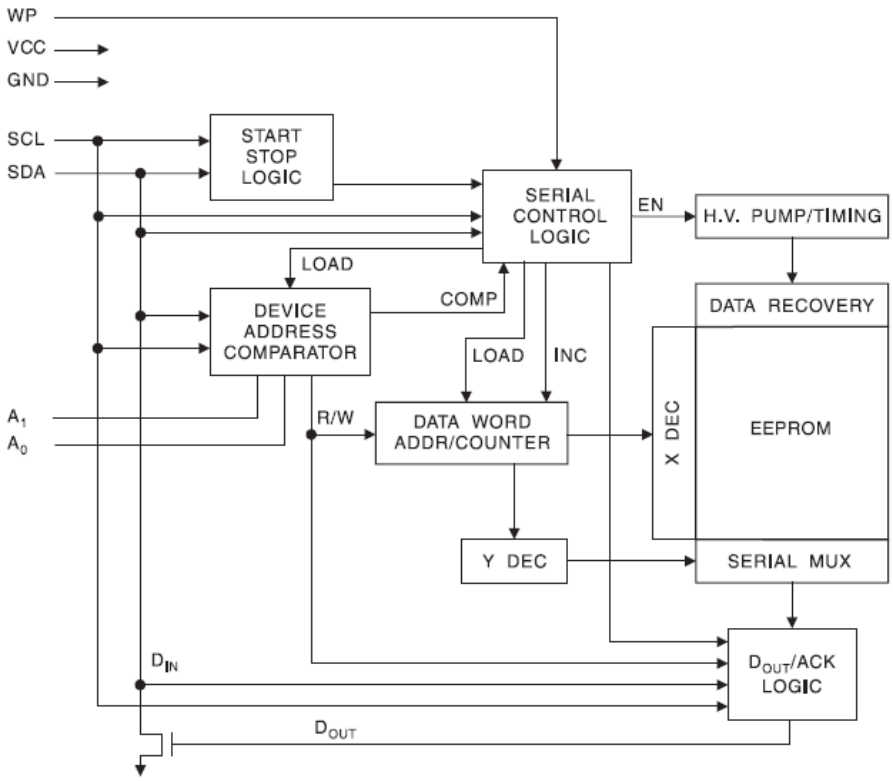

Figura 25 - Diagrama de blocos da memória EEPROM AT24C256 [12]

3.3 MCU 13213

O microcontrolador MC13213 (figura 26) é um microcontrolador da família HCS08 produzido pela Freescale. Ele integra em um mesmo chip um transmissor de rádio frequência MC1320x com um microcontrolador MC9S08GB60 de 8bits[13].

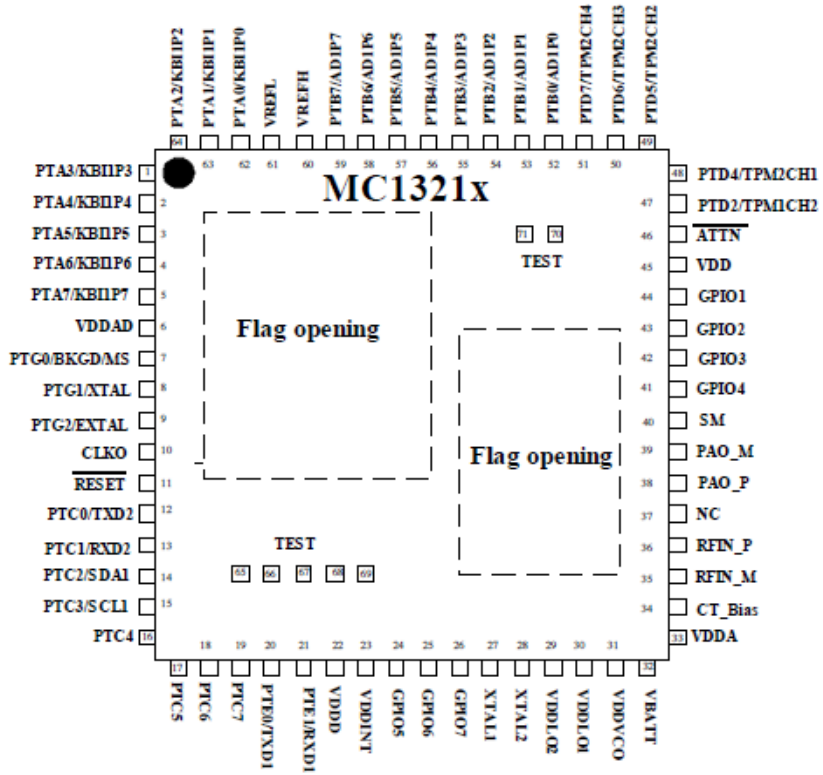

Figura 26 - Pinagem do MC13213 [13]

O transmissor de rádio frequência opera na banda de frequências ISM de 2.4GHz com 16 canais, contando com um amplificador de baixo ruído, com potência nominal de saída de 1mW, além um oscilador integrado de voltagem.

O microcontrolador presente no conjunto provê (figura 27):

- 60KB de memória flash e 4KB de memória RAM.
- Clock da CPU de 40MHz.
- Modo de baixo consumo.
- Interface serial (SPI) conectada ao modem 802.15.4. O microcontrolador é o dispositivo mestre e o modem o dispositivo escravo. Tal interface é usada SOMENTE para a comunicação entre modem e microcontrolador.
- Conversor A/D (ATD) com 10 bits multiplexados em oito canais.
- Interface Serial I²C.
- Módulo de interrupção de teclado (KBI) com 8 entradas.
- Gerador de Clock Interno (ICG)
- Dois módulos de interface serial assíncrona (SCI) para comunicação serial.
- Capacidade de depuração e programação flash através do BDM (Background Debug Module)

- 32 pinos de entrada/saída para uso geral
- Dois *timers* de 16 bits.

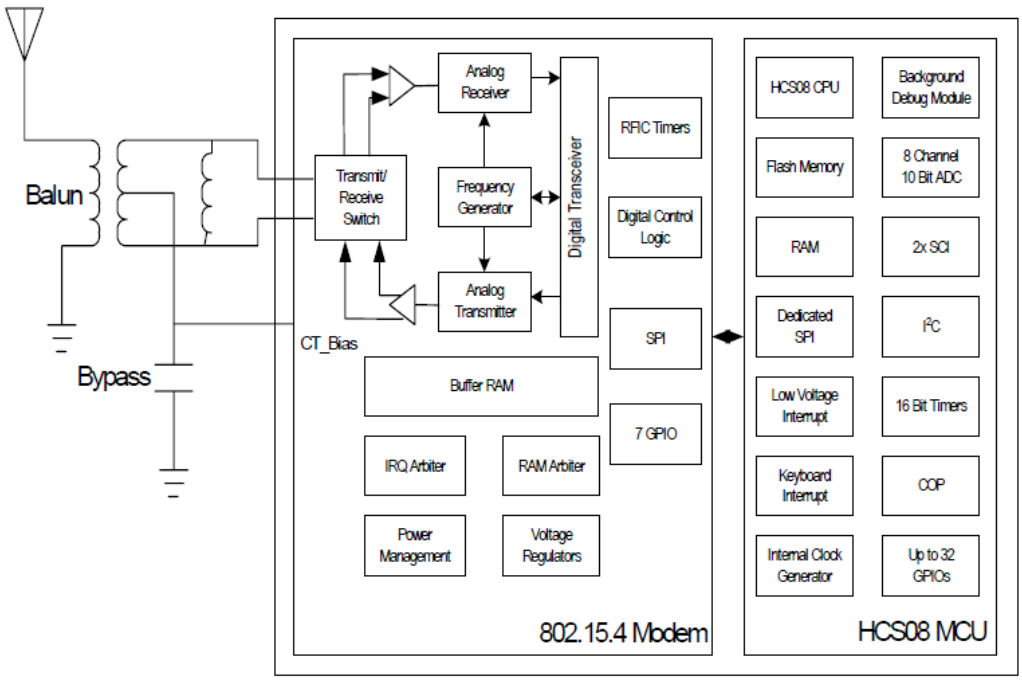

Figura 27 - Diagrama de blocos do MCU 13213 [13]

Dentre os recursos presentes no MC13213, serão melhores descritos o conversor ATD e a interface serial I²C. O conversor ATD será utilizado para realizar a conversão dos sinais analógicos provenientes dos sensores de temperatura para um sinal digital com a temperatura. Por fim, a interface serial I²C será a interface utilizada para a comunicação com a memória EEPROM modelo AT24C256, descrita anteriormente. Maiores detalhes sobre os registradores e métodos destes dois módulos estarão no capítulo 3.5.4 e 3.5.5.

3.4 BDM – Background Debug Mode

Para a gravação e depuração dos projetos nas placas dotadas do microcontrolador MC13213, foi utilizado um módulo BDM USB Multilink fabricado pela *P&E Microcomputer Systems, Inc* [14] (figura 28). O módulo permite realizar a gravação dos códigos do projeto nas placas de destino e assim testar e depurar passo-a-passo as aplicações criadas. O BDM permite até dois *breakpoints* (pontos de parada no código).

Figura 28 - BDM produzido pela PE Micro. Fonte: www.pemicro.com

Para o correto funcionamento, basta plugar o conector USB do módulo BDM a um computador e seu cabo em forma de fita em um conector de 6 pinos respeitando a pinagem descrita na figura 29:

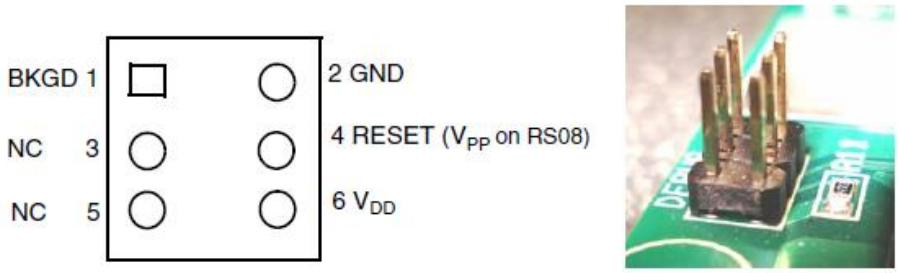

Figura 29 - Pinagem do conector para gravação através do BDM [14]

3.5 Desenvolvimento do Software

A versão do PopNet disponibilizada para o desenvolvimento do projeto foi a PopNet SDK [3]. Esta versão conta com aplicações-exemplo e o pacote de suporte a placas (BSP), que contém as bibliotecas para portar o PopNet para os principais microcontroladores da plataforma Freescale, como MC1319x, MC1321x e MC1322x.

O primeiro requisito para que fosse possível desenvolver uma aplicação sobre o protocolo PopNet foi a instalação do software Freescale BeeKit. Isto ocorreu porque o instalador do PopNet adicionou duas "*codebases*" (uma codebase para a família HCS08 e outra para a família ARM7) na mesma pasta onde o software Beekit possui outras *codebases*. Além disso, foi instalada também toda a documentação do protocolo.

Concluída a instalação, é necessário abrir o software Beekit e selecionar a *codebase* do PopNet (figura 30). Surgem então opções de aplicações já prontas em

PopNet, além da possibilidade de iniciar uma a partir do zero. Os exemplos presentes foram de grande ajuda no desenvolvimento do projeto, fornecendo condições para entender o funcionamento e a estrutura de um código construído sobre o protocolo PopNet. Existem cinco exemplos prontos [15]:

- PopAppOnOffLight – aplicação mais simples, é considerado o "*Hello World*" do protocolo, para ligar LEDs ou lâmpadas à distância.
- PopAppMaze – um jogo em que vários jogadores podem jogar a distância utilizando se botões ou acelerômetro das placas.
- PopAppTest – exemplo para testar manualmente uma rede e sua entrega de pacotes.
- PopAppGeneric – exemplo genérico para desenvolver qualquer aplicação, possui somente o básico para constituir uma rede PopNet.
- PopAppTracking – exemplo mais completo, com opções de monitoramento e rastreamento, é uma aplicação pronta.

O projeto foi construído e modificado sobre o exemplo do *PopAppOnOffLight*.

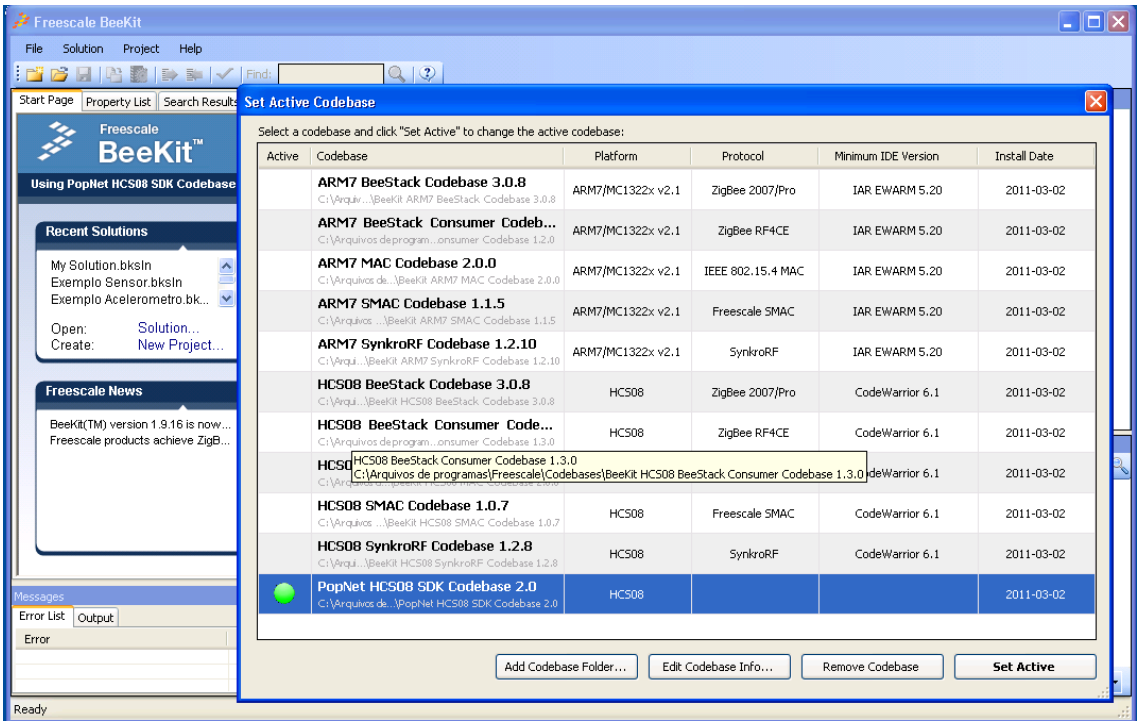

Figura 30 - Seleção da codebase PopNet no software BeeKit

Já no próprio BeeKit, é possível definir uma série de parâmetros e propriedades no projeto (que são basicamente *#defines* em C). As propriedades do projeto vão desde as definições de canais utilizados para a transmissão, endereços dos nós, propriedades da rede formada até a ativação ou desativação de recursos

como interface serial, LEDs, LCDs, etc. e propriedades do *kernel* do protocolo. Uma coleção de projetos no BeeKit forma uma solução. Esta solução deve então ser exportada para que possa ser editada, depurada e compilada na IDE CodeWarrior.

3.5.1 CodeWarrior IDE

CodeWarrior é uma IDE (*Integrated Development Environment*) para o desenvolvimento de softwares para aplicações embarcadas. A versão utilizada para o desenvolvimento do projeto foi a CodeWarrior *Special Edition*, de licença gratuita, versão 6.3, destinada aos microcontroladores da família HCS08. Tal ambiente de desenvolvimento inclui as funções de edição, compilação e depuração no código. Para que ocorra a gravação e a depuração do software nos microcontroladores HCS08, é necessário um BDM para a comunicação (descrito anteriormente).

A figura 31 exibe o ambiente de desenvolvimento do CodeWarrior. O acesso aos diversos arquivos de um projeto é realizado de forma simples através do menu à esquerda. O código do arquivo fica totalmente disponível para edição na janela principal.

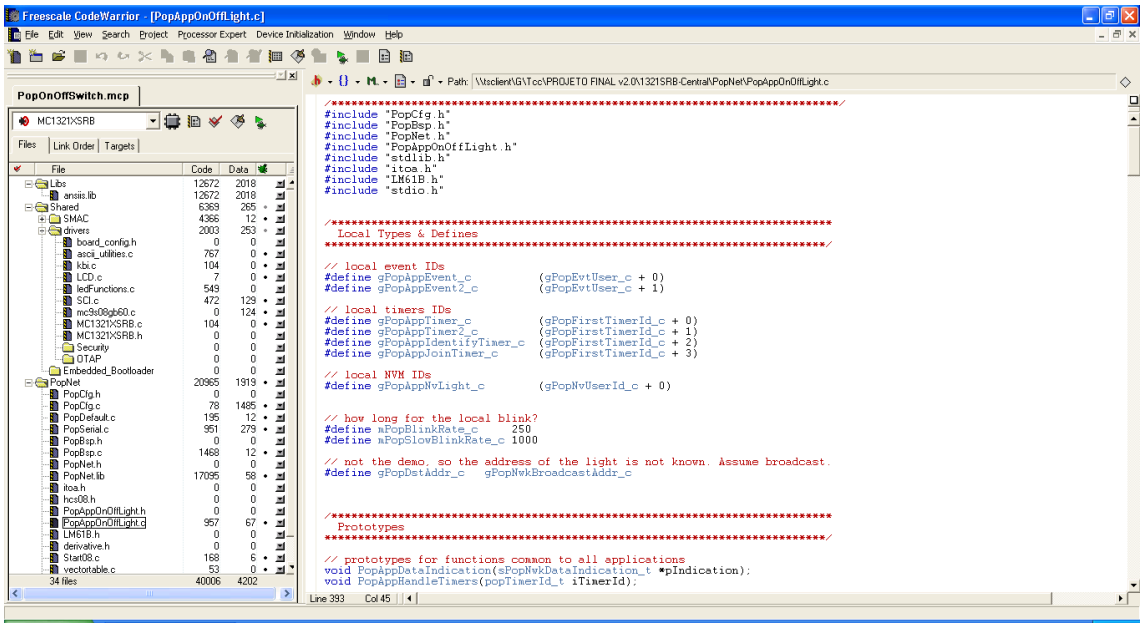

Figura 31 - Ambiente CodeWarrior

Para compilar o projeto, basta ir ao menu “Project – Make” ou através da tecla F7. Para que ocorra a depuração e gravação nas placas, basta ir ao menu “Project – Debug” ou tecla F5. No momento que isto ocorre, uma nova janela é aberta (figura 32)

com as opções para a gravação no microcontrolador. Nesta janela se seleciona o BDM a ser utilizado.

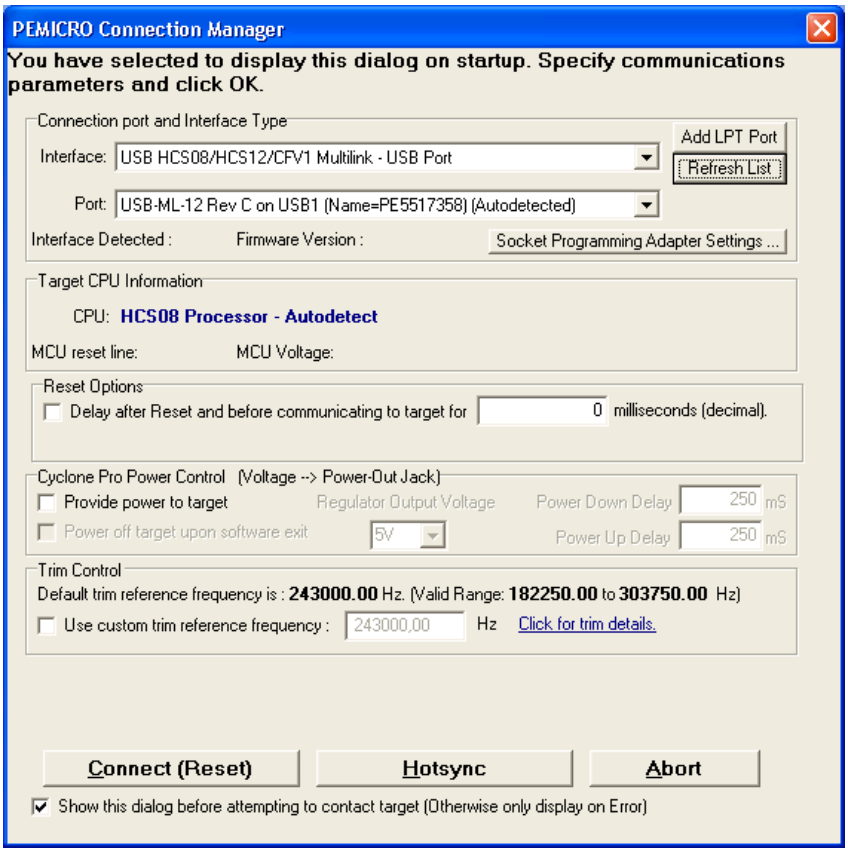

Figura 32 –Gravação no microcontrolador

Após a correta gravação da aplicação no microcontrolador, é possível realizar o monitoramento dos registradores, memória e outros recursos do microcontrolador conforme figura 33. Ainda é possível realizar a execução passo a passo, interromper a execução, reiniciá-lo, além de poder inserir breakpoints (pontos de parada) no código.

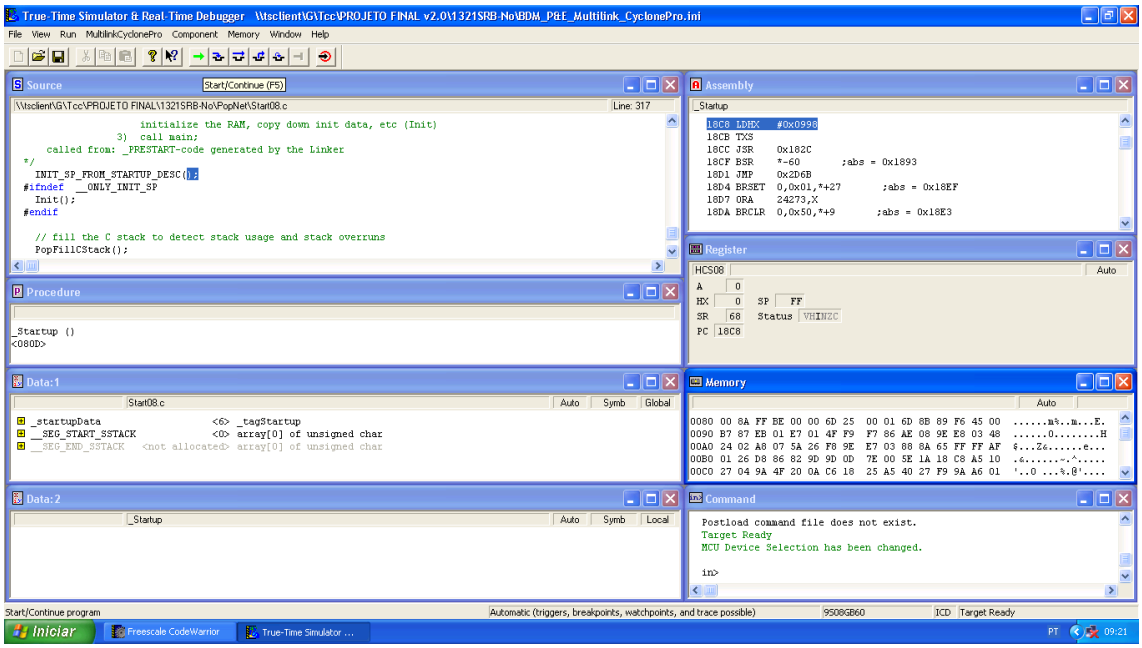

Figura 33 –Monitoramento dos recursos do microcontrolador

3.5.2 Estrutura de uma aplicação em PopNet

De posse das ferramentas necessárias para o desenvolvimento e gravação de aplicações baseadas em PopNet sobre as placas disponíveis para o projeto, agora é a hora de apresentar a estrutura de uma aplicação desenvolvida sobre o protocolo PopNet.

Basicamente todo projeto desenvolvido em cima do PopNet apresenta a seguinte estrutura de arquivos, apresentada na figura 34:

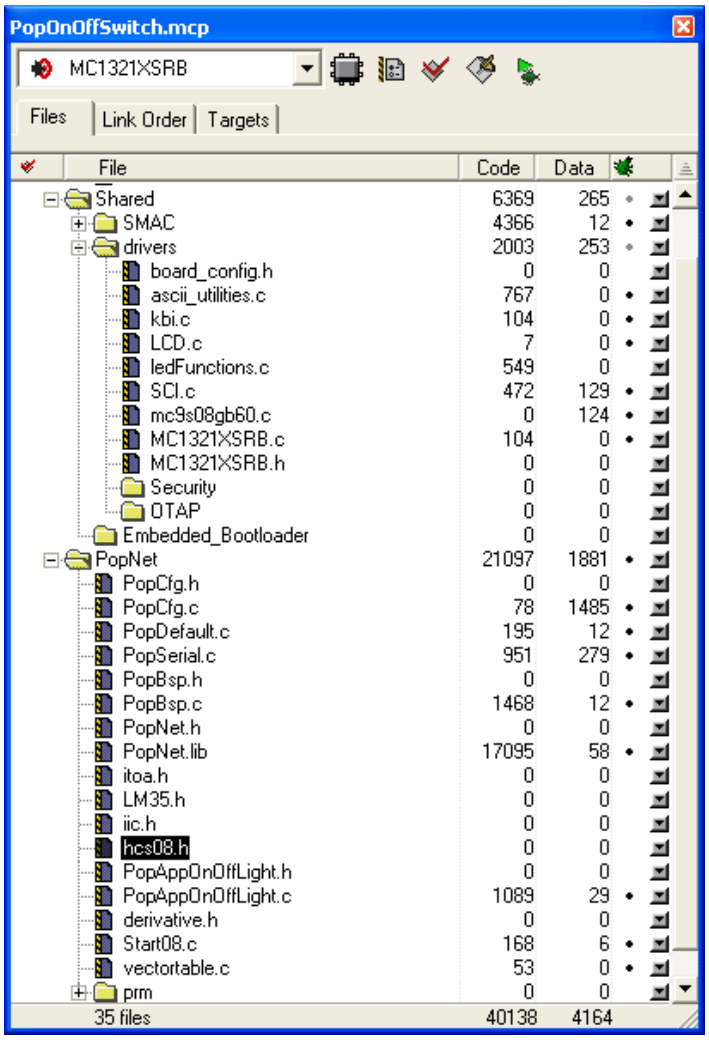

Figura 34 - Estrutura de Arquivos do projeto PopNet

A pasta chamada "*drivers*" contém os drivers para comunicação com os periféricos do microcontrolador. Os arquivos mais importantes nesta pasta são:

- *mc9s08gb60.c*: este arquivo contém a nomenclatura dos registradores presentes no microcontrolador, para que possam ser referenciados por outros arquivos, como no header para a comunicação I²C e conversor A/D.
- *MC1321XSRB.h*: este *header* define as funções dos pinos de entrada/saída do microcontrolador. Este arquivo foi modificado para adequar a pino referente ao LED das placas de aquisição remota.

Já para a pasta PopNet, a tabela 1 descreve as funções de cada arquivo:

Tabela 1 - Estrutura de arquivos em um projeto PopNet

ARQUIVO	FUNÇÃO

PopCfg.h	Este arquivo é responsável pelos parâmetros e definições da rede. Ao realizar mudanças nos parâmetros no BeeKit, é este arquivo editado.
PopCfg.c	Atribuição das variáveis presentes no arquivo PopCfg.h.
PopDefault.c	Arquivo para restaurar configurações iniciais do projeto, liberar memória, etc.
PopSerial.c	Arquivos com configurações e *drivers* para a comunicação serial do PopNet.
PopBsp.h	Interface entre a placa utilizada e o PopNet.
PopBsp.c	Realiza a conexão dos drivers com o PopNet. Funções para ligar LED, escrever em LCDs, etc.
PopNet.h	Definição de variáveis utilizadas pelo protocolo, formato dos pacotes, registros, etc.
PopNet.lib	Arquivo não editável.
Itoa.h	Rotina de conversão de inteiro para *string* para uso na serial.
LM35.h	*Header* para manipulação do módulo ATD e medição de temperaturas. No caso do coordenador, LM61B.h
Iic.h	*Header* para manipulação do módulo I²C, com rotinas de escrita e leitura na memória EEPROM.
PopAppOnOffLight.h	*Header* da aplicação. Define atributos específicos da aplicação
PopAppOnOffLight.c	É a aplicação propriamente dita. O usuário deve escrever o código da aplicação neste arquivo.
Start08.c	Código em *assembly* responsável pela inicialização do microcontrolador.
Vectortable.c	Gerencia a tabela de interrupções.

A ideia da aplicação é a possibilidade de se acessar diretamente um determinado nó, utilizando o método de envio de dados *unicast* para recuperar os dados dos sensores nele armazenado. Para que isto fosse possível, foi fundamental a utilização de uma variável definida no header *PopCfg.h*, *gPopNodeAddr_c*. Com ela, é possível atribuir um endereço único a cada nó na rede PopNet. Assim, bastou que durante a chamada dos métodos de envio e recebimento de dados, se passasse esta variável como parâmetro. Assim, as ações foram direcionadas somente ao nó que possui o *gPopNodeAddr_c* desejado.

Retomando os conceitos já descritos no capítulo 2.2 no que tange ao assunto de envio e recebimento de dados, é importante ressaltar como isto foi realizado no projeto. O coordenador enviou através do *PopNwkDataRequest* () o endereço único *gPopNodeAddr_c* para encontrar a placa da qual deseja obter os dados. Como auxílio para monitorar o funcionamento da rede, este método enviou comandos para acender o LED da placa desejada.

Já as placas de aquisição remota utilizaram o método *PopNwkDataRequest* () para enviar os dados de temperaturas lidos das memórias EEPROM diretamente para o endereço da placa coordenadora (no caso do projeto, “0”).

Para o método *PopAppDataIndication* de recepção dos dados, o coordenador recebeu os dados de temperatura através da comunicação sem fio para exibi-los no computador. Já nas placas de aquisição remota o método recebia o comando para acender seu LED e em seguida já invocava o método para envio dos dados ao coordenador. Isto garantiu que somente a placa que recebeu o pedido da coordenadora enviasse os dados.

3.5.3 Considerações do Projeto

Foi necessário adicionar ao projeto os dois *headers* referentes às operações de leitura do sensor de temperatura e manipulação da memória EEPROM. Com isso, bastou chamar as funções por ele definidas no código principal para que a aplicação funcionasse corretamente.

Para que fosse possível observar o conceito de redes mesh oferecido pelo protocolo, utilizou-se de dois eventos presentes no protocolo: *gPopEvtDataPassThru_c* e *gPopEvtNwkRoutePassThru_c*. Estes eventos são gerados por um determinado nó toda vez que este participar do roteamento e da transmissão de dados entre dois nós distintos. Isto pode ocorrer quando os nós estejam fora de alcance para comunicação ou então impedidos de se comunicarem

por obstáculos presentes entre os nós. Assim, no código foi definido que toda vez que estes eventos ocorrem, o LED da placa que está roteando e retransmitindo os dados é ativado.

Durante o desenvolvimento do projeto, algumas considerações tiveram de ser tomadas. A principal envolve a questão de intervalos de leitura dos sensores. Ora, uma rede de sensores voltada para o monitoramento nem sempre precisa que as temperaturas sejam monitoradas em tempo real. Com a utilização da memória e sua capacidade máxima, teve-se de pensar em medir as temperaturas apenas em determinados intervalos de tempo, para fornecer maior autonomia ao sistema. Assim, foi definido que uma leitura do sensor de temperatura ocorreria a cada 1 minuto, aproximadamente.

Para armazenar as temperaturas na memória, definiu-se que o valor da temperatura ocuparia 8 bits, que é o espaço de uma palavra na memória EEPROM. Os intervalos de leitura foram de -128 a 127ºC (com precisão de 1ºC), que abrangem as faixas de temperaturas mensuráveis pelos sensores LM61B e LM35. Como a memória EEPROM AT24C256 presente nas placas possui 256Kb de espaço, divididos em 32.768 de palavras de 8 bits, temos, portanto, 32.768 posições para armazenar leituras de temperatura.

Com a definição de intervalo de leitura de 1 minuto temos autonomia nas placas para que sejam armazenadas leituras durante 32.768 minutos. Isto significa que as placas podem armazenar leituras sem a necessidade de transmitir os dados para a central por cerca de 530 horas, ou seja, aproximadamente 22 dias. Tais características são importantes em sistemas de monitoramento com dispositivos em constantes deslocamentos, como veículos automotivos e caminhões frigoríficos, por exemplo.

Para que fosse possível a leitura das temperaturas e posterior gravação na memória apenas de um em um minuto foi utilizado o timer do protocolo PopNet. Para utilizá-lo, basta utilizar a função *PopStartTimer ()*, com o parâmetro de tempo em milissegundos. No momento que o timer atinge o tempo máximo a ser medido, ele gera um evento *gPopEvtTimer_c,* que invoca o método *PopAppHandleTimers*. Dentro deste método, então, é possível ordenar nova leitura do sensor e posterior gravação em uma posição da memória. Para provocar um loop de ações de 1 em 1 minuto, basta chamar então novamente a função *PopStartTimer ()*. Como o timer do PopNet é em nível de software, as medições de tempo não são em tempo real. Isto ocorre devido à ocorrência de eventos e momentos de maior processamento durante suas contagens, mas seu contador oferece uma boa estimativa para a aplicação.

3.5.4 Implementação do I²C

O funcionamento do protocolo I²C já foi explicado no capítulo 2.3. Para utilizar-se da ideia deste protocolo, utiliza-se de registradores presentes no microcontrolador para gerar as condições necessárias para a escrita e a leitura dos dados. Existem cinco registradores que garantem o funcionamento desta interface serial (em anexo):

- **IICC** – Registrador responsável pelo controle da interface I²C
- **IICS** – Registrador responsável por definir o status da interface no momento.
- **IICF** – Registrador responsável por definir a frequência para as operações de escrita e leitura.
- **IICA** – Registrador que armazena o endereço da interface I²C para o modo escravo.
- **IICD** – Registrador responsável por escrever e ler bytes do buffer da interface.

Para configurar a interface I²C então, foi adicionado ao projeto um *header* em C chamado *iic.h* obedecendo aos seguintes passos [7]:

- Definir a velocidade de comunicação desejada, alterando o registrador IICF.
- Habilitar o módulo I²C, setando o bit IICEN do registrador IICC.
- Setar o bit MST do registrador IIC para o modo mestre. Isto provoca uma condição de START no barramento.
- Setar o bit TX do registrador IIC para habilitar transmissão.
- Escrever o byte de endereçamento no registrador IICD, o que provoca a comunicação com o dispositivo escravo (no caso a memória EEPROM). Aguardar a transferência ser completamente, que ocorre com a *flag* TCF=1 no registrador IICS.
- O programa deve verificar se houve recebimento da *flag* RXAK (do registrador IICS), que reconhece a transmissão. Se RXAK=0, o escravo recebeu a informação. Caso contrário, o mestre provoca uma condição de *STOP* e reinicia a comunicação.
- O mestre agora pode enviar ou receber bytes de dados. Isto depende do bit R/W enviado no byte de endereçamento. Para R/W=0, o mestre enviará dados para o escravo. Caso contrário, o mestre receberá dados do dispositivo escravo.
- Para transmitir dados, o mestre deve fazer com que TX = 1, escrevendo o dado no registrador IICD. Após o byte ser transmitido, o escravo enviará um bit ACK

ou NACK. No término da transmissão, o mestre deve gerar uma condição *STOP*, através de MST = 0.

- Para receber dados, o mestre deve fazer com que TX = 0 e ler o registrador IICD. Esta primeira leitura (denominada *dummy*) é apenas para iniciar a recepção de um novo dado.
- O *flag* TCF deve ser setado e lê-se novamente o registrador IICD.
- Com o dado corretamente recebido, o dispositivo mestre deve configurar o bit TXAK = 0, para emitir o bit ACK. Para encerrar a comunicação, o mestre deve setar o bit TXAK, o que provoca uma condição NACK no barramento. O mestre também pode encerrar a transmissão fazendo MST = 0;

3.5.3 Conversor A/D ATD

O módulo de conversão analógico-digital presente no microcontrolador é bastante simples. Ele constitui-se basicamente de um conversor A/D SAR de 10 bits, um multiplexador analógico com suporte para oito canais e um sistema de controle. O módulo pode realizar conversões sinalizadas e não sinalizadas e definir o alinhamento do resultado à esquerda ou direita [8]. O módulo possui cinco registradores (ver anexo), dos quais três são utilizados para o gerenciamento, configurações e controle do sistema e dois registradores para armazenagem das leituras do resultado da conversão.

A operação do módulo é iniciada simplesmente pela escrita no registrador ATDSC. A conversão utiliza cerca de 30 ciclos de *clock* do conversor para ser concluída. Além disso, utilizando o bit ATDCO do registrador ATDSC, é possível realizar uma conversão simples (uma conversão por disparo, precisa reescrever no registrador ATDSC para nova conversão) ou conversão contínua.

Os resultados da conversão são armazenados nos registradores ATDRH e ATDRL. Existe a possibilidade de diferentes formatos de saída, desde valores de 8 bits com a presença ou não de sinal (que utilizam somente o registrador ATDRH), até valores de 10 bits com ou sem sinal e com alinhamento à direita ou a esquerda dos registradores.

Assim, para o correto funcionamento dos sensores, basta configurar os registradores de acordo com o canal de entrada em que o sensor está alocado e escolher o formato de saída mais apropriado para a aplicação desejada. A seguir, a partir da leitura das tensões de saída do sensor e da relação tensão de saída por graus Celsius, obter o resultado final já na escala Celsius.

3.5.6 Interface Serial

Para que a central transmita os dados recebidos por ela para o computador, foi utilizado as rotinas de interface serial já definidas no protocolo PopNet: *PopSerialSend* e *PopSerialReceive*. Para ativar a interface serial no projeto, é necessário invocar o método *PopSerialRegister*. Devido à grande quantidade de informações a serem transmitidas, definiu-se também os parâmetros de tamanho do buffer de recepção e transmissão. Estes parâmetros, denominados *gPopSerialRxBufferSize_c* e *gPopSerialTxBufferSize_c,* estão presentes no *header PopCfg.h* e tiveram seus valores alterados para 132 bytes, tamanho máximo permitido pelo protocolo. O *baud rate* da comunicação também é definido neste arquivo através do parâmetro *gPopSerialBaudRate_c*, cujo valor foi de 38400bps. O protocolo já oferece os drivers prontos para comunicação entre o computador e a porta USB presente na placa 1321-SRB, emulando uma porta RS232.

O programa utilizado para receber os dados através da interface serial foi o AccessPort, desenvolvido pela SUDT [16]. O AccessPort é um programa gratuito e sua interface pode ser vista na figura 35.Ele possibilita definir as velocidades de transmissão de dados, envio e recepção de caracteres (em formato ASCII ou hexadecimal), envio de dados através de um arquivo, seleção da porta de comunicação a ser utilizada, etc. Um recurso interessante é a possibilidade de salvar um arquivo em formato texto com os dados exibidos pela interface. Isto é útil ao projeto já que pode servir como um *log* do sistema.

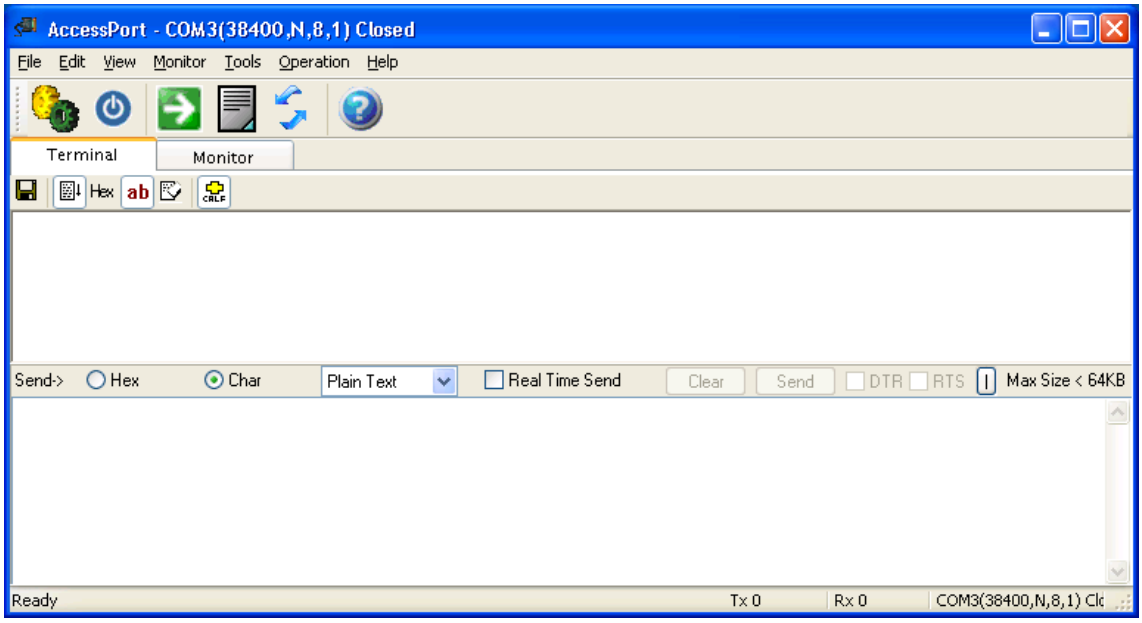

Figura 35 –Interface do programa AccessPort

CAPÍTULO 4 - RESULTADOS

O primeiro passo para a obtenção de medidas corretas no sistema foi a calibração dos sensores utilizados. Para isso, os sensores (um a um) foram comparados com a temperatura fornecida por um termômetro de mercúrio de referência, à temperatura ambiente, onde se observou pequenas diferenças com a referência, as quais foram julgadas desprezíveis para o trabalho (em torno de 1 grau Celsius).

Ao estabelecer uma conexão serial entre o computador e o coordenador da rede, é necessário definir os parâmetros de comunicação no programa AccessPort, através das opções do programa. Estes parâmetros são a porta de comunicação (COM) utilizada e o *baud rate*(velocidade de transmissão de dados) desejado. Feito isto, o programa recebeu os dados pela interface serial e exibiu a seguinte entrada (figura 36):

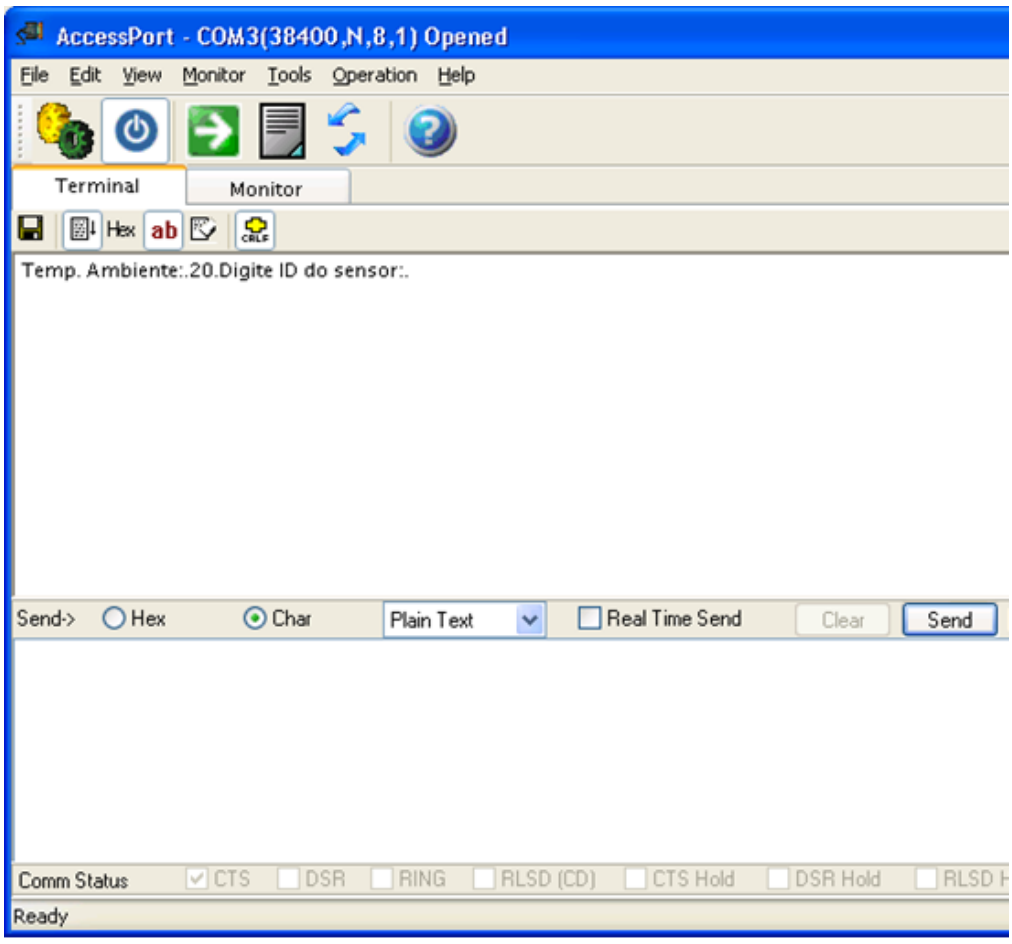

Figura 36 – Dados enviados pelo coordenador

Esta foi a mensagem enviada pelo coordenador, que exibe a temperatura medida pelo sensor LM61B presente na placa. O coordenador aguarda a entrada do usuário, feita diretamente pelo teclado. Bastou digitar o número atribuído como identificador a uma das placas de aquisição remota (este valor foi definido no projeto de 00 a 99 e corresponde ao atributo *gPopNodeAddr_c*) na janela inferior e clicar no botão "*Send*".

Se a placa de aquisição remota possuir o identificador digitado pelo usuário, ela enviará os dados já gravados em sua memória EEPROM através da transmissão sem fio. Vale lembrar que as medições foram realizadas de um em um minuto, mas no momento da solicitação e envio dos dados, foi exibido todo o conjunto de medidas já gravadas na memória da placa de aquisição remota naquele instante. Assim, após cerca de 10 minutos, a leitura da placa de aquisição remota com o endereço *gPopNodeAddr_c* igual a 3 retornou 10 entradas, conforme mostra a figura 37:

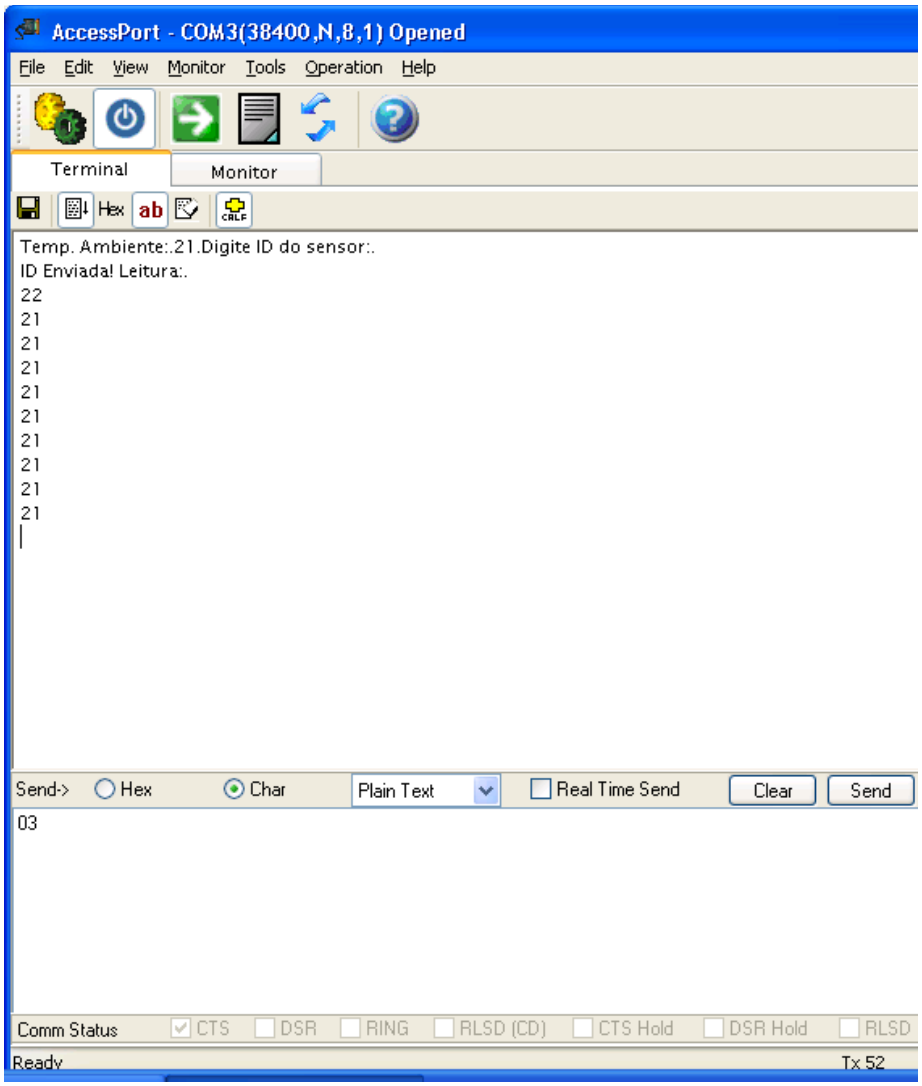

Figura 37 – Leituras realizadas pela placa de aquisição remota

Para se observar alterações nas leituras dos sensores, o sensor da placa de aquisição remota foi aquecido, e esta alteração pôde ser observada nas leituras realizadas novamente (figura 38):

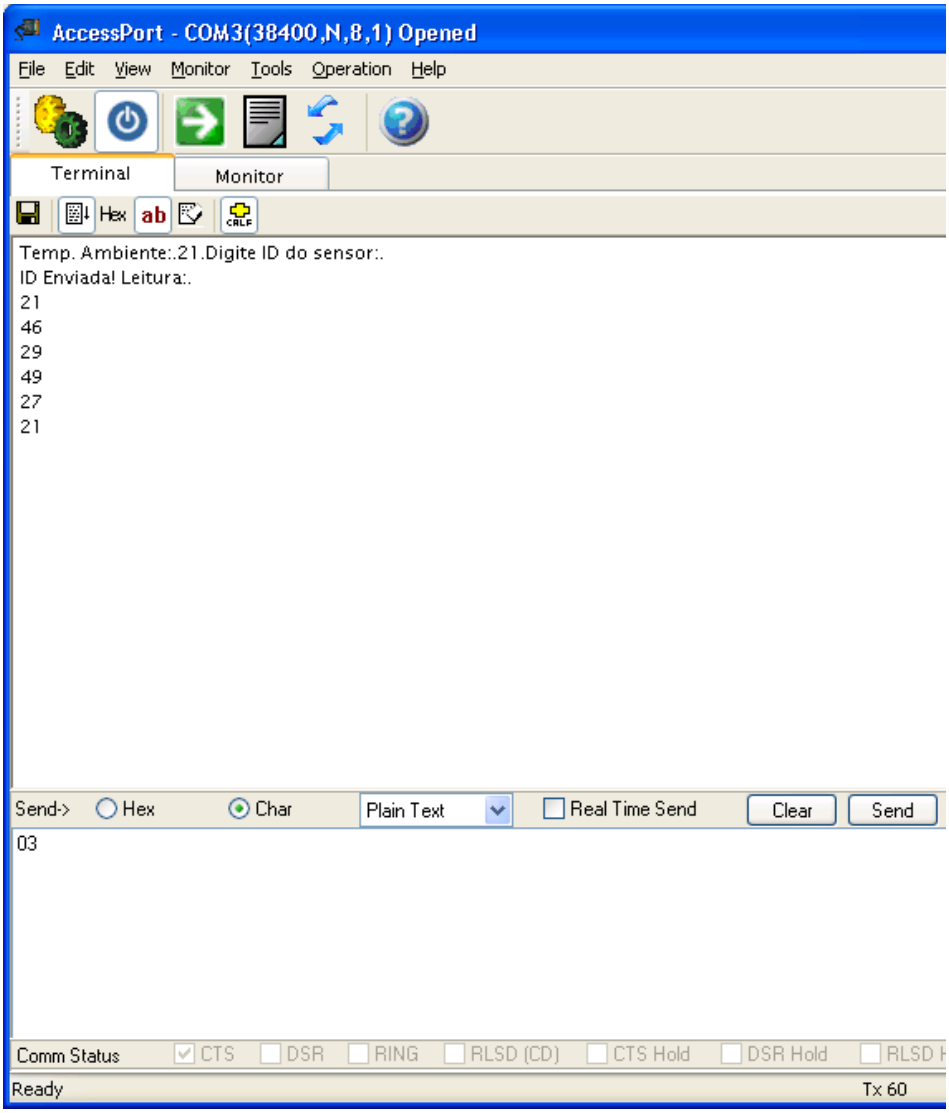

Figura 38 - Leituras dos sensores aquecidos

Para realizar leituras de outra placa de aquisição remota após o envio das informações, bastaria inserir novamente o identificador da placa e clicar no botão "*send*".

Se desejar salvar estas informações em uma espécie de log, o programa AccessPort permite salvar as leituras como um arquivo simples no formato *txt*. Para isto, basta clicar no ícone em forma de disquete.

No intuito de se observar um dos grandes atributos do protocolo PopNet, a formação de redes *mesh*, foram realizados testes com obstáculos entre o coordenador e o nó desejado para comunicação. Com um nó, visível diretamente por ambas as placas, foi possível observar o funcionamento da rede *mesh*, com o recebimento e envio de dados pelo mesmo. Isto pôde ser comprovado pois em todas as vezes que havia uma solicitação de dados por parte da central ao nó desejado, a placa intermediária ativava seu LED, indicando que havia ocorrido o roteamento e a passagem de dados por ela. Quando a placa intermediária foi retirada ou desligada, o coordenador não conseguiu obter as leituras de seu nó alvo.

Os principais problemas encontrados no projeto referem-se a manipulação da interface serial para a exibição dos resultados. Como a quantidade de dados pode ser grande, o buffer da interface serial pareceu não comportar tamanha quantia. Com isso, surgiram problemas em determinados momentos como a não exibição total dos dados, além da não exibição na ordem correta de chegada dos dados.

Outro problema foi a impossibilidade de se realizar testes em um ambiente mais prático, a fim de se observar melhor o funcionamento do sistema e as medições dos sensores nas faixas extremas de medição (temperaturas negativas e/ou temperaturas muito altas).

CAPÍTULO 5 - CONCLUSÕES

A ideia deste trabalho foi utilizar os conhecimentos adquiridos durante a graduação e complementá-los com a pesquisa e compreensão de novos conceitos e recursos.

A utilização de sensores para realizar funções de monitoramento em dispositivos e ambientes sempre será necessária, o que torna o tema do trabalho bastante importante. Além disso, unir este tema à comunicação sem fio oferece uma gama ainda maior de aplicações práticas e facilmente adaptáveis a qualquer ambiente.

A comunicação sem fio é a grande tendência nos últimos anos. Com ela foi possível desenvolver aplicações flexíveis e adaptáveis a qualquer ambiente, reduzindo o uso de fios, tornando diversos sistemas mais portáteis. E com esta tecnologia sendo cada dia mais usada, o surgimento de padrões e protocolos que tratam deste tipo de comunicação tem aumentado.

Assim, o objetivo central deste trabalho foi o estudo de um protocolo ainda pouco usado e explorado, o PopNet. Com uma documentação bastante completa, exemplos inclusos no kit de desenvolvimento e a proposta de ser um protocolo voltado para rede de sensores, o PopNet mostrou-se uma excelente alternativa para o desenvolvimento de aplicações na área. Mesmo com uma curva de aprendizado rápida, o PopNet oferece inúmeros recursos e possibilidades na formação da rede. A proposta de o protocolo ser baseado em redes mesh pôde ser comprovada com a realização de experimentos com situações em que a comunicação entre dois nós fosse impossibilitada por barreiras físicas, sendo necessário um terceiro nó para o roteamento dos dados.

Paralelamente ao estudo do protocolo, o aprendizado para a leitura de sensores mostrou-se prático. É importante ressaltar que os sensores apresentaram leituras com uma boa precisão, mesmo estes tendo um baixo custo.

Por fim, a ideia de este trabalho ser focado em aplicações onde há deslocamentos constantes (monitoramento de veículos e caminhões frigoríficos) possibilitou o estudo e uso de memórias EEPROM. Com ela, foi possível armazenar as leituras dos sensores, mesmo quando estes se encontravam fora do alcance de um coordenador. O protocolo de comunicação I^2C mostrou-se interessante já que utiliza apenas duas linhas bidirecionais para a comunicação.

Os problemas que surgiram no trabalho ocorreram somente na exibição dos dados. A formação da rede e o envio das informações foram realizados com sucesso.

As placas utilizadas e os recursos presentes no microcontrolador MC13213 foram suficientes para a realização do projeto.

Apesar de a aplicação parecer simples, sem grande apelo visual ao usuário, ela se torna funcional a diversos ambientes onde a medição de temperatura é importante. O foco das redes pessoas sem fio baseadas no protocolo IEEE 802.15.4 é justamente esse: simplicidade das informações e dos recursos disponíveis.

5.1 Trabalhos Futuros

O projeto possibilita a expansão e adição de novos recursos:

- Maior aprofundamento dos recursos e métodos do protocolo PopNet, melhorando a formação e as configurações da rede.
- Adicionar outras espécies de sensores, já que existem oito portas para utilização do conversor A/D. Isto permitiria que a aplicação abrangesse um número ainda maior de ambientes e situações.
- Estudar melhor os métodos de consumo de energia oferecidos pelo PopNet.
- Desenvolvimento de uma interface gráfica ao usuário (acesso visual às placas disponíveis, gráficos de temperatura), podendo armazenar as leituras automaticamente em um banco de dados.
- Otimizar a organização das operações de escrita e leitura na memória, adicionando cabeçalhos para indicar as datas de início e término das leituras dos sensores.
- Integração do sistema com a Internet, a fim de possibilitar o controle ainda mais remoto do sistema.
- Utilização de outro microcontrolador com um custo/benefício maior.
- Realizar uma comparação com o protocolo ZigBee, que possui características e recursos semelhantes.
- Desenvolver uma aplicação específica, com a possibilidade de testes no ambiente desejado.

CAPÍTULO 6 - BIBLIOGRAFIA

[1] – What is Sensor Technology? Disponível em:<http://www.dcu.ie/~best/st.htm>. Acesso em 30 Maio 2011.

[2] – IEEE Organization. Disponível em:<http://www.ieee.org/about/index.html>. Acesso em 30 Maio 2011.

[3] –San Juan Software. PopNetDeveloper's Guide v2.0. 2010.

[4] –IEE Computer Society. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs).2006. Disponível em <http://standards.ieee.org/about/get/802/802.15.html>. Acesso em 30 Maio 2011.

[5] –Engen,Sinem C. ZigBee/IEE Summary. 2004. Disponível em:<http://staff.ustc.edu.cn/~ustcsse/papers/SR10.ZigBee.pdf>. Acesso em 30 Maio 2011.

[6] - Philips. Interface Products Business Line Specialty Logic Product Line I2C Logic Family Overview. 2004. Disponível em <http://www.nxp.com/acrobat_download2/various/philips_i2c_logic_overview.pdf>>. Acesso em 03 Junho 2011.

[7] – Freescale Semiconductor. How to Use IIC Module on M68HC08, HCS08, and HCS12 MCUs. 2007. Disponível em: <http://cache.freescale.com/files/microcontrollers/doc/app_note/AN3291.pdf>. Acesso em 30 Maio 2011.

[8] – Pereira, Fábio. Microcontroladores HCS08: Teoria e Prática. 1ª ed. Editora Érica Ltda., 2005. 204p.

[9] – Freescale Semiconductor. 13213 Evaluation Kits User's Guide. 2007. Disponível em: <http://cache.freescale.com/files/rf_if/doc/user_guide/13213EVKUG.pdf>. Acesso em 30 Maio 2011.

[10] – National Semiconductor. Datasheet LM61. 2010. Disponível em <http://www.national.com/mpf/LM/LM61.html#Overview>. Acesso em 30 Maio 2011.

[11] – National Semiconductor. Datasheet LM35. 2000. Disponível em: <http://www.national.com/mpf/LM/LM35.html#Overview>. Acesso em 30 Maio 2011.

[12] - Atmel. Datasheet AT24C256. 2007. Disponível em: <http://www.atmel.com/dyn/resources/prod_documents/doc0670.pdf>. Acesso em 30 Maio 2011.

[13] –Freescale Semiconductor. MC13211/212/213 ZigBee™- Compliant Platform 2.4 GHz Low Power Transceiver for the IEEE® 802.15.4 Standard plus Microcontroller Reference Manual. 2010. Disponível em: <http://www.freescale.com/files/rf_if/doc/ref_manual/MC1321xRM.pdf>. Acesso em 30 Maio 2011.

[14] - Freescale Semiconductor. Introduction to HCS08 Background Debug Mode. 2006. Disponível em: <http://www.freescale.com/files/microcontrollers/doc/app_note/AN3335.pdf>. Acesso em 30 Maio 2011

[15] – San Juan Software. PopNet Application User's Guide v2.0. 2010.

[16] – SUDT AccessPort Debugger for RS232/422/485 Serial Port. Disponível em <http://www.sudt.com/en/ap/index.html>. Acesso em 24 Junho 2011.

CAPITÚLO 7 - ANEXOS

Todas as figuras e descrições dos registradores foram obtidas no manual [13] e no livro [8].

7.1 Registradores I²C

As figuras abaixam mostram os registradores presentes no módulo I²C do microcontrolador utilizado e as funções de cada um de seus bits.

7.1.1 Registrador IIC1C

	7	6	5	4	3	2	1	0
R	IICEN	IICIE	MST	TX	TXAK	0	0	0
W						RSTA		
Reset	0	0	0	0	0	0	0	0

Figura 39 - Registrador IIC1C

- **IICEN**: Ativação do módulo I²C. “0” para interface I²C desabilitada. “1” para habilitar.
- **IICIE**: Habilitação de interrupções. “0” para interrupções desabilitadas. “1” para habilitar
- **MST**: Seleção do modo mestre. “0” para modo escravo. “1” para modo mestre. A transição de “0” para “1” gera condição de *START* e “1” para “0” gera condição de *STOP*.
- **TX:** Seleção do modo de transmissão. “0” para modo de recepção. “1” para modo de transmissão.
- **TXAK:** Seleção do bit de reconhecimento ACK (*Acknowledge*). “0” para reconhecimento. “1” para não reconhecimento.
- **RSTA:** Bit para reinício. Setar este bit permite que múltiplos bytes de dados possam ser enviados sem a necessidade de gerar condições de *STOP* ao fim de cada dado.

7.1.2 Registrador IIC1S

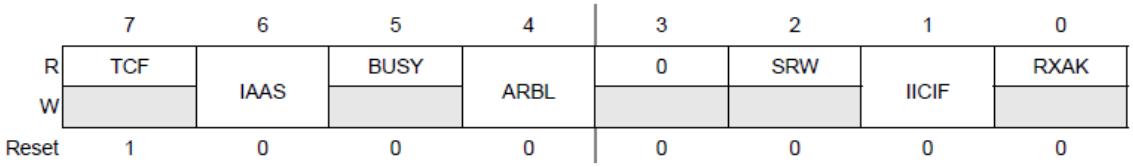

Figura 40 - Registrador IIC1S

- **TCF:** *Flag* para sinalizar transferência completada. A *flag* é apagada toda vez que ocorre uma leitura ou escrita no registrador IIC1D.
- **IASS:** Este bit é setado quando no modo escravo, o escravo recebe um dado (presente no registrador IIC1A) com o endereço coincidente ao seu.
- **BUSY:** Indica o estado do barramento. “1” para barramento livre e “0” para barramento ocupado
- **ARBL:** Indica a perda de arbitragem na linha. Só é utilizado em situações com vários dispositivos-mestre. “0” para operação normal, “1” para perda de arbitragem.
- **SRW:** Indica o tipo de operação que está ocorrendo no barramento, para o modo escravo. “0” para leitura de dados, “1” para escrita de dados.
- **IICIF:** sinaliza a existência de interrupções no barramento. “0” para nenhuma interrupção e “1” para interrupções pendentes.
- **RXAK:** indica o reconhecimento do bit ACK. “0” para reconhecimento recebido, “1” para não reconhecimento.

7.1.3 Registrador IIC1F

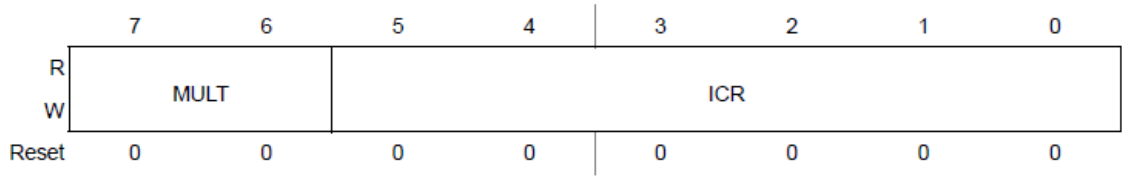

Figura 41 - Registrador IIC1F

- **MULT:** Fator de multiplicação do *clock* I²C
- **ICR:** usado para definir o *clock* do barramento de acordo com a taxa de transmissão de dados. O campo ICR define o divisor do *clock* (SCL) e o valor SDA *hold*. O valor do divisor do *clock* multiplicado por MULT fornece a taxa de transmissão do barramento I²C. Os valores de ICR de acordo com as configurações seguem a tabela XX:

Tabela 2 - Tabela para definir ICR do registrador IIC1F

ICRTAP2-TAP1 (hex)	SCL Divider	SDA Hold Value
00	20	7
01	22	7
02	24	8
03	26	8
04	28	9
05	30	9
06	34	10
07	40	10
08	28	7
09	32	7
0A	36	9
0B	40	9
0C	44	11
0D	48	11
0E	56	13
0F	68	13
10	48	9
11	56	9
12	64	13
13	72	13
14	80	17
15	88	17
16	104	21
17	128	21
18	80	9
19	96	9
1A	112	17
1B	128	17
1C	144	25
1D	160	25
1E	192	33
1F	240	33

TAP2-TAP1ICR (hex)	SCL Divider	SDA Hold Value
20	160	17
21	192	17
22	224	33
23	256	33
24	288	49
25	320	49
26	384	65
27	480	65
28	320	33
29	384	33
2A	448	65
2B	512	65
2C	576	97
2D	640	97
2E	768	129
2F	960	129
30	640	65
31	768	65
32	896	129
33	1024	129
34	1152	193
35	1280	193
36	1536	257
37	1920	257
38	1280	129
39	1536	129
3A	1792	257
3B	2048	257
3C	2304	385
3D	2560	385
3E	3072	513
3F	3840	513

7.1.4 Registrador IIC1A

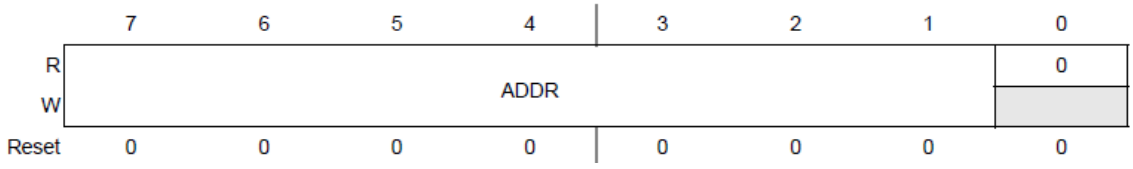

Figura 42- Registrador IIC1A

O registrador IIC1A armazena o endereço da interface I²C se estiver operando no modo escravo.

7.1.5 Registrador IIC1D

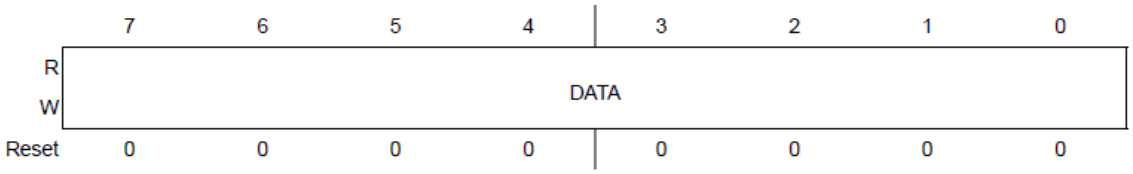

Figura 43 - Registrador IIC1D

Este registrador é responsável por armazenar os dados a serem escritos ou lidos na interface I²C.

7.2 Registradores ATD

7.2.1 Registrador ATDC

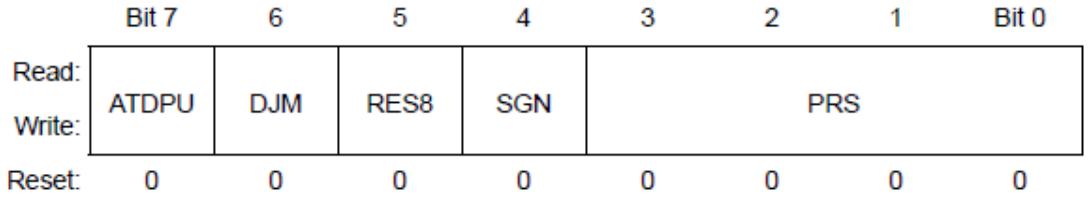

Figura 44 - Registrador ATDC

- **ATDPU**: Bit para ativar a operação do módulo ATD. “1” para ATD ativo, “0” para desabilitado.
- **DJM**: Este bit determina como um resultado de 10 bits ficará nos registradores ATDRH e ATDRL. “0” para 10 bits justificados à esquerda e “1” justificados à direita.
- **RES8**: Seleciona se o resultado da conversão será armazenado em 8 ou 10 bits. “0” para 10 bits, “1” para 8 bits.
- **SGN**: Define se o resultado possuirá sinal ou não. “0” para resultado não sinalizado, “1” para sinalizado.
- **PRS**: Utilizado para definir o *clock* do módulo ATD.

7.2.2 Registrador ATDSC

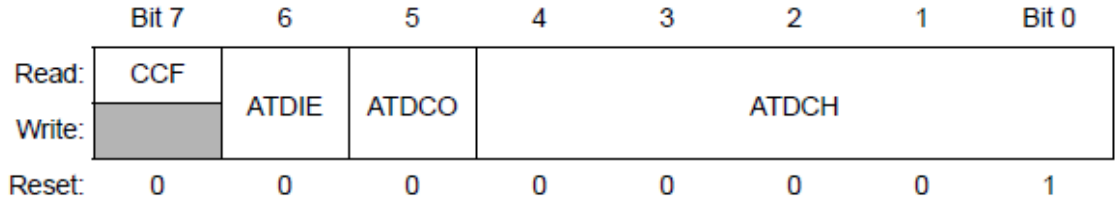

Figura 45- Registrador ATDSC

- **CCF**: sinaliza se a conversão foi completada. O CCF é apagado toda vez que há uma leitura ou escrita no registrador ATDSC. “0” para conversão não completada e “1” para conversão completada.
- **ATDIE**: Habilita interrupções. “0” para interrupções desabilitadas e “1” para habilitá-las
- **ATDCO**: Define se a conversão será simples ou sempre contínua. “0” para conversão simples e “1” para conversão contínua.
- **ATDCH**: Define o canal de entrada para ser convertido. Desde o canal AD0 ao AD7 (presentes no microcontrolador).

7.2.3 Registradores ATDRH e ATDRL

Os registradores ATDRHL e ATDRL são registradores simples, de 8 bits cada, voltados para a armazenagem dos resultados de conversão. A utilização do registrador ATDRL só ocorre se a conversão utilizar 10 bits.

7.2.4 Registrador ATDPE

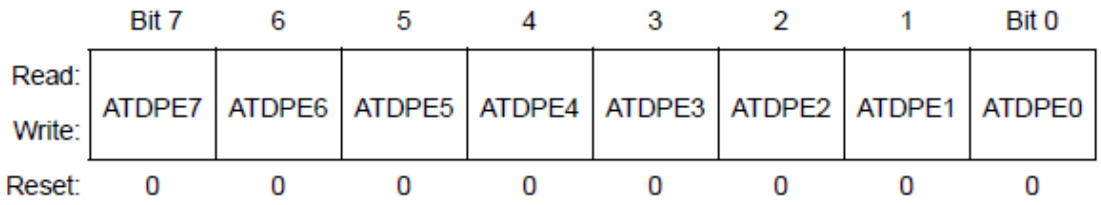

Figura 46 - Registrador ATDPE

Este registrador é responsável por definir a função do pino de entrada. O pino setado no bit “0” funciona como uma porta de entrada/saída digital, não permitindo conversões. O pino setado no bit “1” assume a função analógica.

7.3 Glossário PopNet

A tabela 3 abaixo apresenta os termos mais comuns e utilizados para o correto entendimento de um projeto baseado no protocolo PopNet [3].

Tabela 3 - Glossário PopNet

Termo	**Descrição**
ACK	Abreviação para acknowledgement. Usado durante o método de envio Unicast para verificar se o pacote foi recebido pelo nó
Beacon	Pacote enviado por um nó na rede que possui informações como MAC Address e PAN ID. É usado para o nó entrar na rede desejada.
BeeKit	Software da Freescale utilizado na criação de projetos PopNet
Broadcast	Método de enviar dados para uma porção ou toda rede. Eles não possuem ACK.
BSP	Board Support Package. É formado por rotinas que interagem com hardware específico como LEDs e Timers.
Data Confirm	É o resultado proveniente de um Data Request
Data Indication	É o mesmo que o recebimento de um dado.
Data Request	É o mesmo que transmissão de um dado.
Hop	Define o próximo dispositivo que receberá um pacote.
MAC Address	Identificador de 64 bits que identifica unicamente um dispositivo.
Mesh	Topologia de rede que aumenta o alcance e a confiabilidade utilizando diversos hops para a transmissão de dados.
Multi-hop	Enviar um dado utilizando-se de vários nós da rede.
Node (Nó)	Um dispositivo com rádio na rede.
Node Address	Também chamado de Short Address, é um identificador único de 16 bits do nó.
NVM	Memória não volátil (Non-volatile memory). É um armazenamento que não é apagado mesmo após resetar ou reiniciar o dispositivo.
PAN ID	Identificador único de uma rede de área pessoal.
Platform	Família de rádios e microcontroladores, como Freescale MC1321x (Baseado na família HCS08) ou Freescale MC13224 (Baseado na família

	ARM7).
Project	Conjunto de arquivos-fonte e bibliotecas que constituem uma aplicação, a ser importada para o CodeWarrior.
Property	É basicamente um #define. É uma opção que pode ser modificada durante o uso do BeeKit e que será executada apenas em tempo de compilação. Utilizada para configurar uma aplicação PopNet.
Radius	O número de hops que um pacote utiliza para chegar até o nó destino.
Route	O caminho que um pacote toma através de uma rede mesh multi-hop.
Solution	Coleção de um ou mais projetos criados no BeeKit
Unibroadcast	Conceito único do protocolo PopNet, onde um envio unicast é feito através do mecanismo broadcast. Muito utilizado para descobrir rotas, onde o dado já é enviado durante o processo de roteamento.
Unicast	Envio de dados de um nó para exatamente outro nó.