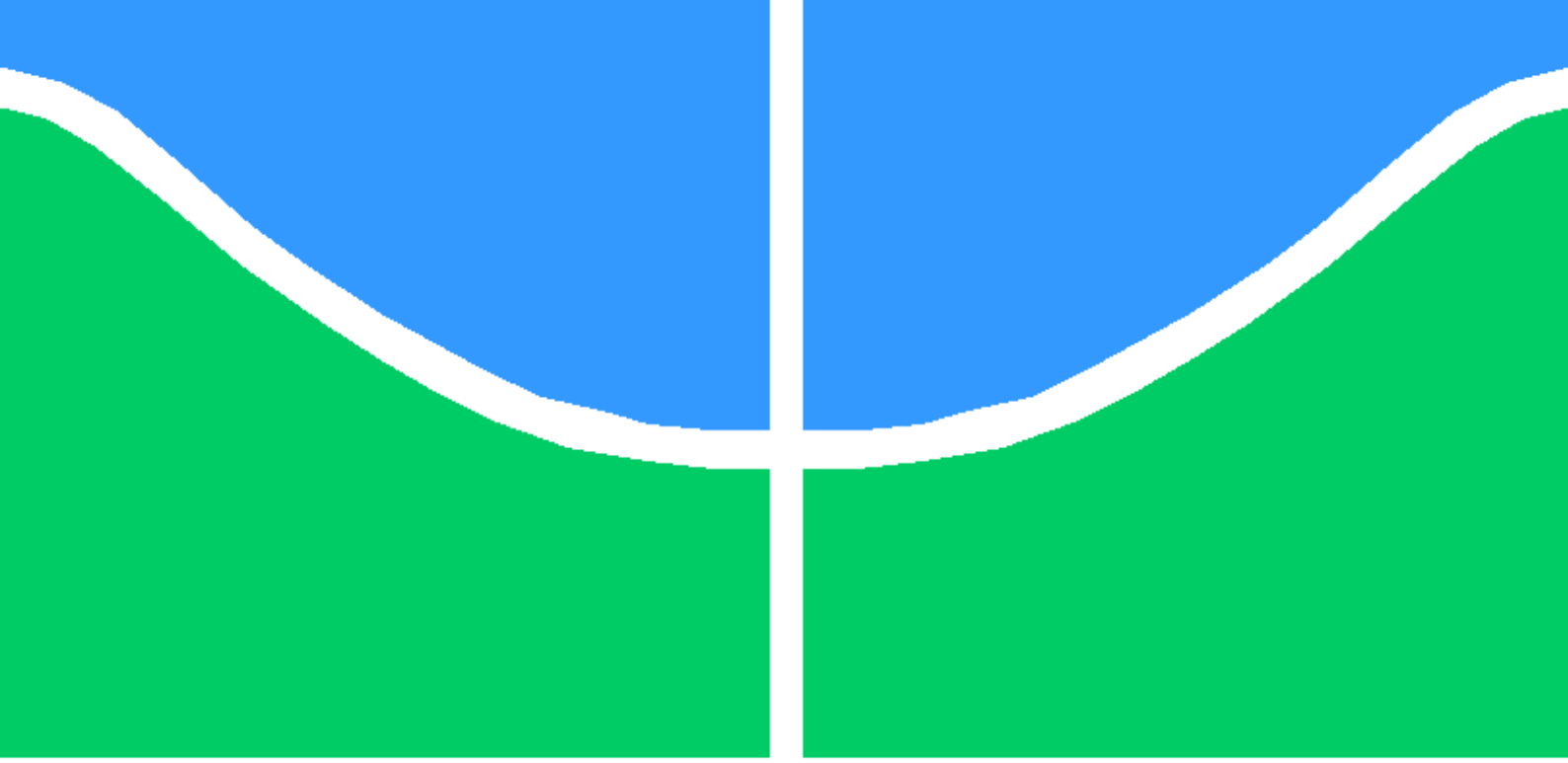

TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO E AVALIAÇÃO DE DESEMPENHO DO PROTOCOLO CSMA/CA IEEE 802.15.4 SOBRE A PLATAFORMA SIMPLICITI E MICROCONTROLADORES MSP430

Reinaldo Gutierrez Pimenta

Rodrigo Lobo Ribeiro de Moraes

Brasília, julho de 2015

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO E AVALIAÇÃO DE DESEMPENHO DO PROTOCOLO CSMA/CA IEEE 802.15.4 SOBRE A PLATAFORMA SIMPLICITI E MICROCONTROLADORES MSP430

Reinaldo Gutierrez Pimenta

Rodrigo Lobo Ribeiro de Moraes

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro Eletricista

Banca Examinadora

Prof. Marcelo Menezes de Carvalho, ENE/UnB _____
Orientador

Prof. José Edil Guimarães de Medeiros, ENE/UnB _____
Examinador interno

Eng. Hermano Albuquerque, Innovix _____
Examinador externo

Dedicatórias

Dedico este trabalho às pessoas que me acompanharam por esta longa e tortuosa jornada de quase 7 anos na Engenharia Elétrica. Aos amigos, que partilharam angústias e vitórias, à minha família, que foi sempre um porto seguro nos momentos de dificuldade e à minha namorada, que com carinho e ternura me trouxe a calma necessária para seguir em frente.

Rodrigo Lobo Ribeiro de Moraes

Dedico este trabalho à minha família, principalmente à minha mãe Julia e ao meu irmão Braulio, aos meus amigos e a todos que ajudaram e fizeram possível este trabalho, sendo essenciais para eu chegar até aqui. Em especial, ao meu pai José Adib e ao meu avô Braulio, que não puderam estar aqui para presenciar esta grande vitória.

Reinaldo Gutierrez Pimenta

Agradecimentos

Agradeço a todos que ajudaram na realização deste trabalho. Em especial ao Rodrigo Lobo pela amizade e por aceitar o desafio de realizar um trabalho cuja área é muito pouco abordada durante o nosso curso de Engenharia Elétrica, ao professor Marcelo, por ter acreditado em nós desde o início, quando tínhamos muito(!) a aprender, ao professor Edil por ter emprestado os kits de desenvolvimento do MSP430 e ao Hermano por nos ter confiado uma parte importante de seu trabalho. Agradeço também ao Goku por salvar a Terra mais uma vez.

Reinaldo Gutierrez Pimenta

Gostaria de agradecer a todos que, de alguma maneira, contribuíram para que eu pudesse trilhar este longo caminho até este momento: aos amigos que fiz durante o curso, que me ensinaram o valor da camaradagem e do trabalho em equipe; ao professor Marcelo, que acreditou em nós e muito nos ensinou durante este ano de orientação; aos meus pais, Paulo e Sílvia, por serem sempre um modelo de caráter, honestidade e perseverança, me orientando com palavras de sabedoria nos momentos de aflição; ao meu irmão Iago, pelo carinho e apoio incondicional, acreditando sempre no meu potencial; à minha namorada Manuela, que com muito carinho e paciência me ajudou a passar pelos momentos mais difíceis; e finalmente ao amigo Reinaldo Pimenta, coautor deste trabalho, obrigado pela confiança em realizar este trabalho em parceria comigo, as dificuldades encontradas certamente fortaleceram nossa amizade..

Rodrigo Lobo Ribeiro de Moraes

RESUMO

Este trabalho apresenta a implementação do protocolo CSMA/CA IEEE 802.15.4, de tempo não compartimentado, sobre a plataforma SimpliciTI, da *Texas Instruments*. O desempenho do protocolo implementado é avaliado sobre uma rede de sensores baseada nos microcontroladores MSP430, e comparado com o desempenho do protocolo CSMA/CA original implementado na plataforma SimpliciTI. Os desempenhos de ambos os protocolos são avaliados sobre redes com diferentes números de nós e taxas de geração de pacotes, e comparados segundo o número de recuos ("*backoffs*") necessários para entrega de pacotes com sucessos, número de pacotes perdidos e vazão efetiva da rede.

ABSTRACT

This work presents an implementation of the non-slotted CSMA/CA MAC protocol of the IEEE 802.15.4 standard over the SimpliciTI plataform, by Texas Instruments. The performance of the implemented protocol is evaluated over a sensor network based on MSP430 microcontrollers, and compared against the peformance of SimpliciTI's original CSMA/CA protocol. Both protocols are evaluated under different number of nodes and data rates, according to the number of backoff stages needed to successfully deliver data packets, the number of data packets delivered unsuccessfully, and network throughput.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Siglas

IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
IoT	Internet das Coisas (*Internet of Things*)
RSSF	Rede de sensores sem fio (*Wireless Sensor Network*)
RF	Rádio Frequência
SAR	Conversor de aproximação sucessiva(*Sucessive approximation converter*)
MSP430	Processador misto de sinais (*Mixed Signal Processor*)
RISC	Computador com conjunto de instruções reduzidas (*Reduced Intruction Set Computer*)
CCA	Avaliação de disponibilidade do canal (*Clear channel assessment*)
SPI	Interface serial periférica (*Serial Peripheral Interface*)
VCO	Oscilador controlado por tensão (*Voltage controlled oscilator*)
DCO	Oscilador controlado digitalmente (*Digitally controlled oscilator*)
FIFO	Pilha (*First in, first out*)
RFID	Identificação por rádio frequência (*Rafio-Frequency Identification*)
TDMA	Time Division Multiple Access (*Acesso múltiplo por divisão de tempo*)
CTS	Livre para enviar (*Clear-to-send*)
RTS	Pronto para enviar(*Ready-to-send*)
CDMA/CA	Acesso múltiplo com verificação de portadora com prevenção de colisão (*Carrier Sense Multiple Access with Collision Detection*)
AP	Ponto de acesso (*Access Point*)
ED	Nó da rede (*End Device*)
API	Interface de programação de aplicações (*Application Programming Interface*)
LPM3	Modo de baixo consumo 3 (*Low Power Mode 3*)
AM	Modo ativo (*Active Mode*)
RSSI	Indicador da força do sinal recebido (*Received Signal Strength Indicator*)
PF	Falhas de envio de pacotes (*Packet Fail*)

Capítulo 1

Introdução

1.1 Motivação

A Internet das Coisas (IoT, ou *Internet of Things*) é um conceito que se tornou realidade há pouco tempo. A partir deste conceito, vários objetos do nosso cotidiano estão permanentemente conectados, trocando informações, dados de sensores e atuadores, extendendo o limite do que conhecemos como conectividade. Pelo simples fato de estes objetos estarem conectados, a presença de fios e cabos para estabelecer a conexão entre eles torna a portabilidade e a versatilidade proibitivas, sendo a comunicação por RF (rádio frequência) a melhor saída para estabelecer uma conexão entre estes objetos. Como o espectro de frequência utilizável de forma viável é limitado, são necessárias diversas técnicas de codificação, transmissão e recepção, tendo como objetivo a economia de energia, melhor utilização do espectro de frequência e um melhor desempenho dos dispositivos em questão.

Uma aplicação muito explorada da Internet das Coisas é a Rede de Sensores Sem Fio (RSSF). Dependendo de sua topologia e aplicação, uma RSSF possui diversos dispositivos, denominados nós, que fazem a medição, avaliação e transmissão de parâmetros relacionados ao ambiente em que se encontram, tais como temperatura, umidade, pressão atmosférica, detecção de movimento, dentre outros. Estes dados podem ser enviados para um ponto de acesso, onde o usuário pode avaliar e determinar alterações no ambiente, dependendo do tipo de aplicação ou da variável do ambiente que se deseja controlar.

O consumo de energia é hoje um fator limitante para o desenvolvimento e implementação de vários sistemas baseados na Internet das Coisas. Como os dispositivos armazenadores de energia possuem um tamanho muito maior do que os outros dispositivos utilizados na composição de nós das redes de sensores sem fio, tais como microcontroladores, transceptores RF, sensores e atuadores, faz-se necessário o estudo de formas e técnicas de economia de energia, além da pesquisa sobre os processos e componentes consumidores de energia.

O objetivo deste trabalho é a implementação, teste e comparação de dois protocolos MAC: o protocolo CSMA/CA disponibilizado na plataforma SimpliciTI, da Texas Instruments, e o protocolo CSMA/CA, definido na norma IEEE 802.15.4, que foi implementado neste trabalho na mesma

plataforma para efeito de comparação e avaliação. A RSSF em questão é montada utilizando as plataformas de desenvolvimento RF-2500, da *Texas Instruments*, utilizando o microcontrolador MSP430 e o transceptor RF CC2500 disponíveis na plataforma para realizar as medições, processamento e envio de dados da rede. A implementação dos dois protocolos nas plataformas da rede é feita a partir da adaptação do código fonte do protocolo de RSSF SimpliciTI, sendo os microcontroladores programados a partir do *software* de desenvolvimento Code Composer Studio versão 5.5, também da *Texas Instruments*. Neste trabalho são realizados testes na RSSF, cujos dados do ponto de acesso são obtidos e processados através de um programa criado em MATLAB. Estes testes avaliam a quantidade de perda de pacotes, assim como a vazão da rede e o número de *backoffs* realizados pela rede.

1.2 Objetivos

Dentre os processos existentes numa RSSF, o envio de dados por rádiofrequência está entre os maiores consumidores de energia. O consumo de energia está diretamente associado ao alcance do transceptor, que, dependendo da aplicação, pode gerar um compromisso entre alcance da rede e vida útil dos nós, elementos críticos para a viabilidade da implementação da RSSF em questão. Como consequência, a perda de pacotes durante o envio de dados é tido como um fenômeno altamente indesejável. Por causa deste problema, é necessário implementar técnicas visando a economia de energia e eficiência na transmissão de dados nesta área crítica.

Um técnica essencial para evitar a colisão no envio de pacotes é o protocolo de controle de acesso ao meio (MAC, do inglês *medium access control*), que determina como os sensores devem utilizar o canal comum, compartilhado, da forma mais eficiente possível. Em particular, no contexto deste trabalho, tratamos da classe de protocolos MAC conhecidos como protocolos de acesso aleatório (ou por contenção), baseados na técnica de detecção de portadora e diferentes estratégias de recuo (*backoff*) no caso de colisão de pacotes no canal.

Dentro dos objetivos almejados com este trabalho, estão: a familiarização e aprendizado sobre o funcionamento da plataforma SimpliciTI, disponibilizada pela *Texas Instruments*, para a implementação de uma rede de sensores sem fio; a implementação do protocolo CSMA/CA de tempo não-compartimentado do padrão IEEE 802.15.4 na plataforma SimpliciTI; e a avaliação comparativa de desempenho de uma rede de sensores real, operando segundo o protocolo CSMA original da implementação nativa e o protocolo CSMA/CA da implementação proposta.

1.3 Contribuição

Com a realização do presente trabalho, resultados relevantes foram alcançados. Foi realizado com sucesso o projeto e a implementação de um protocolo CSMA/CA do padrão IEEE 802.15.4 na plataforma SimpliciTI da *Texas Instruments*. além disso, foi feita a avaliação de desempenho dos protocolos CSMA original e CSMA/CA implementado sobre a rede de sensores baseada no microcontrolador MSP430; para que tais testes pudessem ser realizados, foram desenvolvidas rotinas

para interfaceamento e coleta de dados dos sensores através do software MATLAB, da *Mathworks*. Estas contribuições permitiram realizar uma análise de desempenho da rede implementada, além de deixar uma base para que novas implementações e testes futuros possam ser realizados.

1.4 Apresentação do manuscrito

No Capítulo 2 é apresentada uma fundamentação teórica sobre o tema de estudo. São introduzidos conceitos importantes relacionados à Internet das Coisas, Rede de Sensores Sem Fio e o principal conceito deste trabalho, o protocolo de controle de acesso ao meio (MAC). Em seguida, o Capítulo 3 faz um detalhamento da plataforma de aplicação do trabalho, o microcontrolador MSP430 e sua placa de desenvolvimento RF-2500. Além disso, descreve o protocolo de RSSF SimpliciTI, os objetos SimpliciTI e o funcionamento básico da rede implementada por estes objetos. O Capítulo 4 apresenta a configuração dos experimentos, metodologia de aquisição de dados, e resultados obtidos nos experimentos realizados com a operação da rede de sensores segundo os dois protocolos CSMA (SimpliciTI original e modificado, baseado no IEEE 802.15.4). O Capítulo 5 faz um breve resumo do trabalho, suas aplicações diretas e comenta também sobre os experimentos efetuados, tais como os resultados obtidos. Por fim, faz um breve relato de possíveis trabalhos futuros. Os anexos contém material complementar.

Capítulo 2

Fundamentação Teórica

Este capítulo trata dos conceitos explorados neste trabalho, tais como a Internet das Coisas, rede de sensores sem fio e controle de acesso ao meio em protocolos de rede de comunicação. Além disso, apresenta um breve histórico da origem de tais conceitos, bem como suas evolução através dos anos e estados atuais.

2.1 Introdução

No mundo de hoje, a miniaturização e a integração cada vez maior de componentes eletrônicos permitiu a popularização cada vez maior de dispositivos eletrônicos móveis de comunicação sem fio, tais como *tablets*, telefones celulares *smartphones*, computadores *notebooks*, dispositivos vestíveis (*google glass, apple watch*). Com a interação deste tipo de dispositivos eletrônicos, veio a necessidade da comunicação entre eles.

O ar, como meio de transmissão de dados, tem como principal vantagem não necessitar utilizar fios e cabos para realizar a comunicação entre dispositivos, aumentando assim a portabilidade e praticidade de utilização dos mesmos. Porém, há uma série de desvantagens na utilização deste meio de comunicação. Dentre elas, podemos destacar um consumo maior de energia e uma taxa menor para o envio de dados, menor qualidade de serviço, devido a uma maior taxa de erro e possíveis interferências, além de uma maior suscetibilidade a problemas relacionados à segurança de dados.

Para contornar tais problemas, várias organizações, comitês, grupos e institutos de engenheiros juntam esforços para definir padrões e estabelecer metas de desenvolvimentos (*roadmaps*), além de realizar pesquisas acerca de determinado problema a ser solucionado no âmbito de comunicações sem fio. Um dos mais famosos institutos deste tipo é o IEEE, Instituto de Engenheiros Eletricistas e Eletrônicos. Fundado em primeiro de janeiro de 1963, resultado da fusão entre o Instituto Americano de Engenheiros Eletricistas e o Instituto dos Engenheiros de Rádio, é uma associação profissional que, dentre as mais variadas atuações na sociedade, estabelece padrões para as mais diversas aplicações de dispositivos eletrônicos, desde o barramento de circuitos integrados até o

protocolo de redes de comunicação.

A padronização de protocolos de redes de comunicação é essencial para o bom funcionamento de qualquer rede de dispositivos eletrônicos e de computadores, além de toda a Internet. Qualquer dispositivo conectado a uma rede de comunicação, que esteja fora dos padrões estabelecidos, irá causar um mal funcionamento da rede, com diversas implicações ao funcionamento da mesma. Esta padronização é estabelecida em vários *standards*, como por exemplo o IEEE 802.11 (redes *WI-FI*), IEEE 802.15.3 (*Ultra-Wideband*, etc.), IEEE 802.15.4 (*Zigbee*, etc.), IEEE 802.3 (*Ethernet*), dentre outros.

2.2 Internet das Coisas

Internet das Coisas (*Internet of things*, ou *IoT*) é um conceito criado no final dos anos 90 pelos fundadores do *MIT Auto-ID center*, um centro de pesquisas relacionadas ao monitoramento de variáveis de objetos, seres e do ambiente, do MIT, *Massashussets Institute of Tecnology*. O termo "*Auto-ID*" se refere a qualquer grande classe de tecnologia de identificação utilizada na indústria para automatizar, reduzir erros e aumentar eficiência. Estas tecnologias incluem códigos de barras, *smart cards*, sensores, reconhecimento de voz e biométrico [9].

O principal objetivo de pesquisas referentes à Internet das Coisas é o desenvolvimento de redes de comunicação que conectem computadores a objetos, e não apenas o *hardware* (dispositivos *RFID* e nós de rede de sensores sem fio) ou o *software* para operar a rede, mas tudo necessário para criar uma Internet das Coisas, desde *hardware* fabricados a preços acessíveis, *software* e protocolos de operação da rede, e linguagens de programação para descrever objetos de tal modo que computadores possam entender e interpretar o ambiente. É importante frisar que a Internet das Coisas não é uma rede concorrente à Internet atual, e sim a adição de elementos que promovam a localização de objetos, seres e a determinação de variáveis de determinado ambiente, além do compartilhamento de informações pela Internet.

Hoje, existem cerca de 1,5 bilhões de computadores pessoais conectados à Internet e mais de um bilhão de telefones celulares com Internet [9]. A "Internet de computadores pessoais" atual irá progressivamente se transformar na "Internet das Coisas", em que de 50 a 100 bilhões de dispositivos estarão conectados à Internet em 2020. Se forem consideradas não somente comunicações entre máquinas mas comunicações entre todos os tipos de objetos, então o número estimado de objetos conectados à Internet sobe para mais de 100 trilhões [9].

Vários especialistas concordam que os desafios da Internet das Coisas serão profundos e extremamente desafiadores. Entretanto, são criadas as mais variadas perspectivas de pesquisas para a Internet das Coisas. Por exemplo, tem-se o conceito de *smart-dust*, uma "poeira inteligente" composta por milhares ou talvez milhões de dispositivos eletrônicos, capazes de comunicarem entre si e realizar medições das mais diversas variáveis do ambiente, tais como temperatura, umidade, pressão atmosférica, nível de poluição do ar, etc. A Figura 2.1 ilustra um grão de *smart-dust*.

O surgimento da Internet das Coisas provavelmente irá provocar quebras e transformações

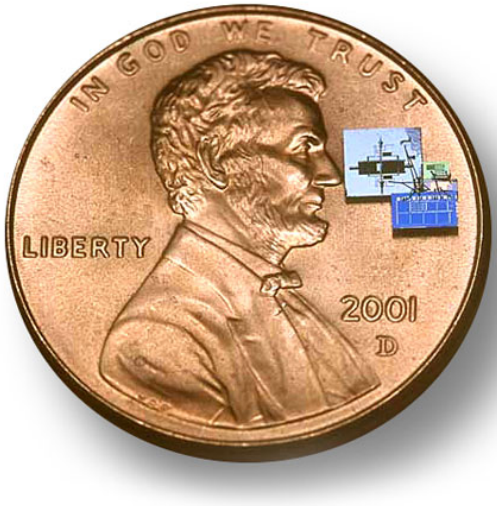

Figura 2.1: *Smart Dust* para a medição de umidade do ar [1].

na indústria, assim como as que anteriormente originaram as maiores revoluções tecnológicas. A Internet das Coisas é um conceito que engloba nanotecnologia, biotecnologia, tecnologia da informação e ciências cognitivas. Pelos próximos dez a quinze anos, é provável que a Internet das Coisas desenvolva rapidamente e molde uma nova "sociedade da informação" e uma "economia do conhecimento", mas a direção e a velocidade com o qual o desenvolvimento irá ocorrer é difícil de se prever.

2.3 Rede de sensores sem fio

A evolução tecnológica e a redução do custo de produção de circuitos eletrônicos, reduzindo a dimensão dos circuitos e dispositivos a níveis microscópicos, permitiu o desenvolvimento dos sistemas micro-eletro-mecânicos (ou *MEMS*) [10], que passaram a integrar diversos dispositivos eletrônicos do uso comum, reduzindo o preço de produção em escala dos produtos eletrônicos e aumentando a capacidade de processamento e operação dos mesmos. Rádios e transmissores que antes eram grandes e pesados, com capacidade de operação limitada hoje são substituídos por dispositivos sem fio com alta taxa de transmissão de dados. Atualmente, a conectividade entre pessoas e dispositivos é um tema que está em constante discussão e estudo. E é neste contexto, dentre muitos tópicos, que encontramos as redes de sensores sem fio, ou RSSFs.

As redes de sensores sem fio costumam ser classificadas como uma categoria de redes ad-hoc, que por sua vez são redes que não dependem de uma infraestrutura pré-estabelecida, operando sem uma topologia específica e podendo realizar comunicação entre os nós de forma centralizada (com ponto de acesso) ou descentralizada (sem ponto de acesso). Uma rede de sensores sem fio é essencialmente composta por um determinado número de nós, que são formados basicamente por um microcontrolador (processador, memória, conversores, além de outros componentes específicos), bateria (para gerar energia elétrica para o nó), rádio (para se realizar a comunicação sem fio entre os nós e/ou entre a interface do usuário, se implementada), sensor (para medir uma ou mais variáveis

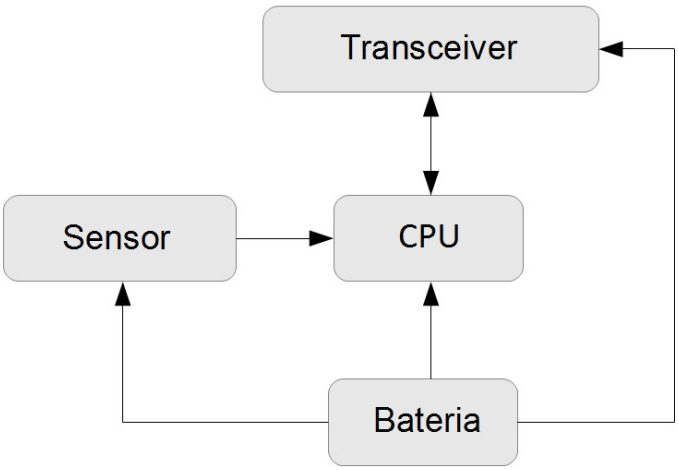

Figura 2.2: Esquemático básico de um nó de uma RSSF.

em questão do ambiente). Na Figura 2.2 está ilustrada a representação dos componentes básicos de um nó de uma rede de sensores sem fio.

A depender da complexidade da rede em questão, os nós podem variar a sua função e capacidade de processamento, além do tipo de sensor que cada nó possui. As redes de sensores sem fio combinam o monitoramento de determinadas variáveis físicas em um ambiente com eletrônica embarcada e comunicação sem fio entre os nós. Estes podem estar equipados com diferentes tipos de sensores, de acordo com a finalidade que a rede foi projetada. Sensores comumente utilizados são sensores acústicos, sísmicos, de infravermelho, de temperatura, de umidade, de luminosidade, de nível de volume, entre outros. A flexibilidade oferecida por redes de sensores as tornam muito atrativas para determinadas aplicações onde não é possível ou desejável a conexão física via cabo entre os sensores e o ponto de acesso, em aplicações onde uma grande área deve ser monitorada ou em aplicações em locais de difícil acesso, por exemplo. Atualmente encontramos RSSFs sendo utilizadas para aplicações como:

- Sistemas de automação e controle, controlando variáveis de processos através do processamento das grandezas medidas e atuação sobre o sistema;
- Contenção de danos em eventos climáticos, onde a rede de sensores monitora a situação de vulcões, mares e ventos, permitindo um processamento desta informação para reduzir danos com intempéries e catástrofes climáticas;
- Segurança e monitoramento de patrimônio, utilizando sensores como tags RFID para controle de estoque;
- Engenharia biomédica, para monitoramento de sinais biológicos em pacientes com sensores e equipamentos implantados, sem a necessidade de uma sonda invasiva para realizar a medida ou colher os dados;

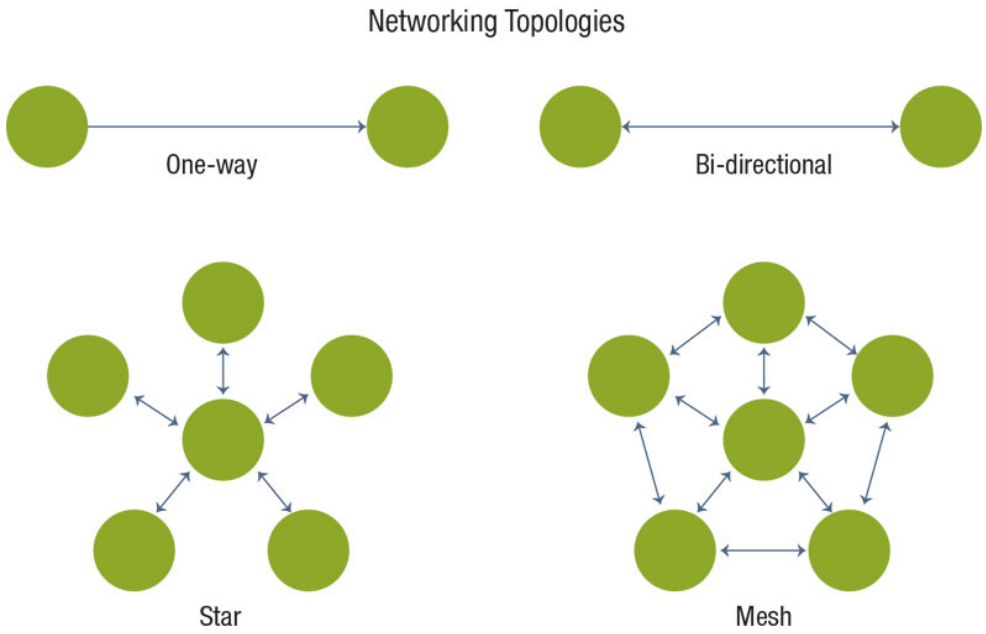

Figura 2.3: Topologias de redes de sensores sem fio [2].

- Aplicações militares, em sistemas de defesa para detecção de inimigos ou para detecção de elementos nocivos, como radiação ou gases venenosos;

Como todo aparato tecnológico, as redes de sensores sem fio apresentam vantagens e desvantagens na sua implementação, que devem ser levados em conta na hora de decidir se utilizar uma RSSF é a solução mais adequada para a realização do sistema. Das vantagens de se utilizar comunicação sem fio, temos a flexibilidade e facilidade de instalação da rede, além da possibilidade de montar diversas topologias, de acordo com a aplicação. Já em relação a desvantagens atreladas a esta tecnologia, podemos apontar as questões de segurança da informação do sistema, maior susceptibilidade à interferência, além da baixa taxa de transmissão de dados em comparação com redes cabeadas.

As RSSFs aparecem como solução para situações onde uma grande área precisa ser monitorada, como parques, estradas e lavouras, por exemplo. Nestes ambientes, é impraticável montar uma arquitetura de rede de monitoramento onde todos os sensores se comunicam com os pontos de acesso via cabo; nesse contexto, a rede de sensores sem fio apresenta-se vantajosa, por se adaptar às condições do ambiente, podendo ser instalada em ambientes que uma rede com cabos não poderia ser instalada. As redes de sensores sem fio podem ser configuradas em diversas topologias de transmissão de dados, de acordo com as necessidades da aplicação. Em redes menores, onde os sensores não se encontram muito distantes do ponto de acesso, é possível configurar uma topologia em estrela, com fluxo de dados *single-hop*, onde o sensor envia seus dados diretamente para o *gateway*. Redes maiores, por sua vez, tem seus sensores muito distantes do ponto de acesso, fazendo-se necessário que a topologia de malha seja utilizada, com fluxo de dados multi-hop, onde os nós se comunicam, transmitindo a informação até o ponto de acesso. Na Figura 2.3, temos uma ilustração destas possíveis topologias:

A utilização de redes de comunicação sem fio está sujeitas a algumas desvantagens, que devem ser levadas em conta de acordo com a aplicação. Redes sem fio estão mais susceptíveis a interferência, uma vez que a transmissão se dá utilizando o ar como meio de transmissão. Como os dados são transportados por ondas eletromagnéticas, dependendo do que há no ambiente da rede, é possível que a transmissão seja prejudicada por conta da interferência com outras ondas eletromagnéticas geradas por linhas de transmissão ou outros aparelhos eletrônicos; em comparação, redes de transmissão de dados a cabo apresentam uma robustez em relação a interferência durante a transmissão.

As RSSFs, por terem limitações de hardware (memória, processamento e rádio transmissor), fornecem taxas de envio de pacotes relativamente baixas. Para aplicações que visam monitoramento simples, sem uma urgência de atuação em resposta a uma leitura do sensor, esse tópico não se mostra como um problema; porém, em redes onde é necessário um monitoramento e atuação em tempo real, este tópico pode ser um ponto crítico [11]. Outro ponto crítico relativo a redes de sensores sem fio é o consumo de energia de seus nós, pois seu tamanho e sua vida útil são fatores determinantes para o sucesso da implementação desse tipo de rede. O compromisso relacionado a estas características está sempre presente em praticamente todas as etapas da criação de uma rede de sensores sem fio, uma vez que definem os fatores limitantes da utilização da rede, quase sempre definidos pelo custo e pela tecnologia empregada. A autossuficiência energética das redes varia de semanas a anos, a depender de alguns fatores, como a configuração de coleta e processamento dos dados pelo nó, por exemplo. Essa vida útil dos sensores depende principalmente do perfil de consumo de energia de cada sensor da rede [3]. Dependendo da aplicação da rede, os sensores não podem ser acessados ou estão em locais de difícil acesso, como sensores em fornos industriais, por exemplo. Nestes casos, a manutenção da rede para substituição de baterias torna-se inviável.

2.4 Protoclo de Controle de Acesso ao Meio (MAC)

As RSSFs têm sua operação baseada em comunicação sem fio entre os nós e o ponto de acesso, ou simplesmente entre os nós, de acordo com a topologia utilizada na implementação do sistema. A comunicação sem fio apresenta dois grandes desafios a serem contornados: eficiência na transmissão e eficiência energética.

Comparada a redes com fios, é possível observar que a capacidade de transmissão de dados de RSSFs é inferior, pois a transmissão pelo ar implica em enfrentar uma série de fatores que acabam prejudicando a transferência de dados entre dois pontos, como a dispersão e atenuação do sinal ao longo da transmissão, que implicam na perda de dados e na redução da frequência de transmissão de dados. Em RSSFs observa-se a ocorrência de muitas colisões de pacotes, principalmente quando a rede possui muitos nós ou está congestionada. Além deste fator, há também a interferência de outros sinais eletromagnéticos que estejam sendo transmitidos na mesma faixa de frequência da rede. A propagação guiada (via cabo) permite uma taxa de transmissão de dados mais elevada, além de não apresentar o problema de interferência, atenuação ou dispersão do sinal ao longo da transmissão.

No quesito de eficiência energética, as RSSFs precisam por definição consumir pouca energia, pois os sensores devem ser espalhados ao longo da área desejada e na maioria das vezes não podem ser conectados à rede elétrica para serem alimentados, dependendo de fontes independentes, como pilhas e baterias [12]. Dessa maneira, a administração da energia disponível para o processamento e transmissão de dados é fundamental para aumentar a vida útil da rede. Em comparação, as redes cabeadas não apresentam esta preocupação, pois normalmente os modems e pontos de acesso estão normalmente ligados à rede elétrica.

Para atenuar estes pontos que prejudicam a utilização das redes sem fio, é possível alterar o protocolo de controle de acesso ao meio, do inglês *Medium Access Control* (MAC), de maneira a melhorar o desempenho energético e o desempenho de transmissão durante a operação, modificando a maneira como os dados devem ser enviados, assim como controlando de maneira inteligente o módulo de transmissão, que é o componente com maior custo energético de operação da rede. Em redes que possuem um meio compartilhado, o protocolo de controle de acesso ao meio é um componente importante para evitar que ocorram colisões de dados transmitidos [3], coordenando o envio dos dados de acordo com a disponibilidade de dados a serem enviados, assim como a disponibilidade do meio livre para transmissão. Em relação à questão de conservação de energia, deve-se observar os períodos de dormência e de atividade dos nós, o ciclo de atividade, uma vez que otimizando este parâmetro é possível reduzir o consumo de energia de cada nó da rede. A literatura apresenta diversas abordagens de desenvolvimento destes protocolos, baseados em múltiplo acesso por divisão de tempo, *Time Division Multiple Access* (TDMA), ou em contenção.

Há protocolos, como o S-MAC (*Sensor MAC*) [3] e o T-MAC (*Timeout MAC*) [13], que coordenam a transmissão de pacotes de maneira síncrona, enviando e/ou recebendo dados durante uma janela de tempo pré-determinada. Na abordagem do TDMA tradicional, o tempo de transmissão da rede é dividido em *frames*, ou quadros, que são subdivididos em *slots*; cada nó tem o acesso a um *slot* de tempo dedicado, em que pode enviar e/ou receber dados, utilizando toda a largura de banda do canal para realizar a transmissão. Nos períodos em que não está designado para transmitir ou receber dados, o nó é colocado em estado de dormência. Esta estratégia requer um sincronismo entre os nós, que muitas vezes é prejudicado pelo atraso natural que ocorre entre os nós dos sensores. Dessa forma, para reduzir este efeito, estas abordagens normalmente incorporam ao protocolo pacotes de sincronização, de pequena dimensão, que envia as informações do nó emissor e o tempo restante até que ele vá para o estado de dormência. Para ilustrar uma situação de transmissão de dados em uma rede TDMA, a Figura 2.4 mostra uma situação onde tem-se um receptor e três emissores, onde o receptor tem seu *frame* dividido em três processos (sincronização, RTS (*ready-to-send*) e CTS (*clear-to-send*) e os emissores funcionam da seguinte maneira: o primeiro apenas envia o pacote de sincronização SYNC, o segundo apenas envia o pacote RTS e recebe o pacote CTS, e o terceiro realiza a sincronização com o pacote SYNC e realiza a transmissão com a troca de pacotes RTS/CTS com o receptor [3].

O TDMA só pode ser utilizado em RSSFs se a alocação dos canais for dinâmica, onde cada canal é alocado somente a um nó quando houver dados para transportar. Dessa maneira, um frame do esquema de acesso TDMA para RSSFs deve possuir uma parte reservada no header do pacote para agendamento dos nós, para que a sequência de transmissão da rede seja definida [12].

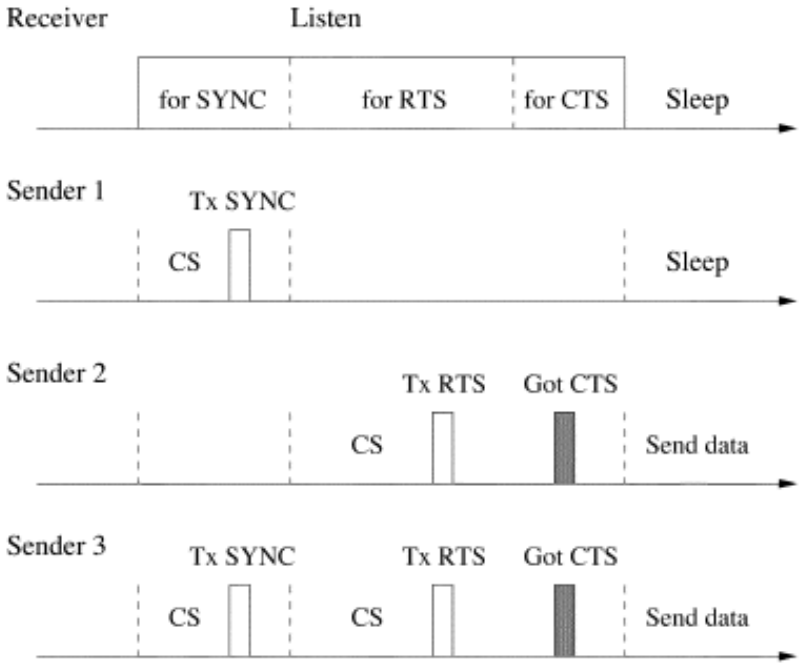

Figura 2.4: Diagrama temporal de funcionamento de rede baseada em TDMA. [3]

O controle de acesso ao meio através de contenção, proposto em protocolos como B-MAC [14], X-MAC [15] e o WiseMAC [16], é uma abordagem assíncrona, não necessitando que os nós da rede estejam sincronizados entre si, permitindo que cada nó possua seu próprio duty cycle independente. Nestas abordagens, as colisões de dados são evitadas através do mecanismo de *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA), onde o nó analisa o canal para verificar se há dados sendo transmitidos; caso o canal esteja ocupado, o nó adormece por um período aleatório e ao fim deste tempo analisa novamente o canal. Caso o canal continue ocupado, este procedimento é repetido até que o canal esteja livre, o nó em questão vença a contenção e os dados possam ser enviados. A Figura 2.5 mostra um fluxograma que exemplifica o funcionamento do protocolo MAC baseado em contenção.

Como analisado neste tópico, o controle de acesso ao meio é de fato um dos mecanismos mais utilizados para implementar melhorias no desempenho das RSSF. A literatura apresenta diversas abordagens para atacar os principais problemas na transmissão de dados sem fio modificando a estrutura de transmissão dos dados e implementando diferentes estratégias referentes à percepção do meio de transmissão. De acordo com a finalidade da rede ou com o objetivo do projeto, é possível melhorar o desempenho de transmissão da rede, reduzindo colisões de dados, assim como melhorar o consumo de energia dos nós, controlando o ciclo de atividade e otimizando o tempo de operação de cada nó.

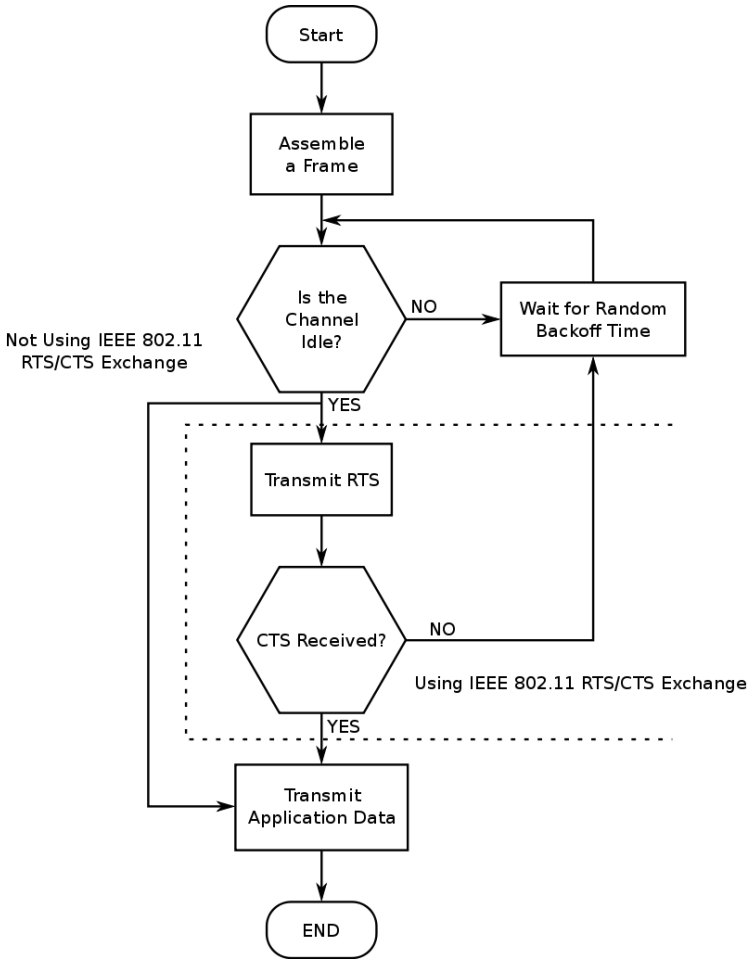

Figura 2.5: Fluxograma de funcionamento de rede baseada em contenção. [4].

Capítulo 3

Plataforma de Aplicação do Trabalho

Este capítulo faz uma introdução e uma breve análise de cada elemento da plataforma de aplicação deste trabalho. Traz também uma descrição detalhada do microcontrolador MSP430, assim como da sua placa de desensolvimento RF2500. Além disso, faz uma descrição do protocolo de redes de sensores sem fio simpliciTI e dos protocolos de acesso ao meio implementados no mesmo.

3.1 Introdução

O trabalho vigente possui uma grande variedade de opções de plataformas para sua implementação. Dentre elas, a mais adequada é a placa de desenvolvimento RF2500, da *Texas Instruments*. Além de possuir todos os componentes necessários para a implementação da infraestrutura de um nó de rede de sensores sem fio, possui um microcontrolador voltado para o baixo consumo, característica essencial para o projeto de uma RSSF com um tempo de operação aceitável. O transceptor RF da placa é o CC2500, também da *Texas Instruments*. Este transceptor RF possui um consumo relativamente baixo, operando na faixa de 2,4 GHz. Sua programação é relativamente simples, realizada por uma placa *debugger* USB.

3.2 MSP430

O MSP430 [17] (*Mixed Signal Processor*) é um microcontrolador de 16 bits de processamento de sinais mistos. É projetado e fabricado pela *Texas Instruments*. Por possuir arquitetura *RISC* (*Reduced Intruction Set Computer*, ou Computador de conjunto reduzido de instruções), favorece um conjunto simples e pequeno de instruções que levam aproximadamente a mesma quantidade de tempo para serem executadas. Esta série de microcontroladores é voltada para o baixíssimo consumo de energia, sendo ideal para aplicações onde a demanda por energia deve ser a menor possível, como por exemplo redes de sensores sem fio. A Figura 3.2 mostra uma placa de desenvolvimento eZ430, contendo um microcontrolador MSP430 modelo F2103, além de sua placa de

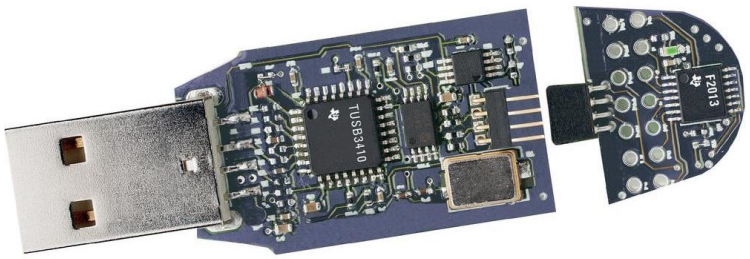

Figura 3.1: Placa de desenvolvimento eZ430-F2013 [5].

programação e *debugging* USB.

Por ser um processador de sinais misto, o MSP430 oferece uma vasta gama de periféricos para atender os mais variados tipos de aplicações. Pode-se citar a presença de conversores analógico-digital de 10 e 12 bits do tipo SAR (*sucessive approximation converter*, ou conversor de aproximações sucessivas) e Sigma-Delta de 16 bits, de conversores digital-analógico de 12 bits, passando por amplificadores operacionais, comparadores, *drivers* de LCD, portas digitais de entrada e saída de aplicações gerais, além de outros tipos de periféricos. Na Figura 3.2 são mostrados os periféricos mais comuns presentes no MSP430.

Por ser um microcontrolador de baixo consumo, o *clock* do MSP430 é relativamente reduzido, podendo ser configurado de 1 MHz até 25 MHz, a depender do modelo. Para aplicações onde o consumo de energia é um fator decisivo, é quase obrigatório que o *clock* seja configurado em um valor relativamente baixo, entre 1 e 8 MHz. O MSP430 possui a opção de ter diferentes fontes de osciladores como *clock* do sistema. Pode-se utilizar cristal externo ao microcontrolador ou um oscilador controlado digitalmente (*Digitally Controlled Oscilator*, ou DCO). A vantagem do DCO está em gerar clocks mais altos e a taxas variadas, além de possuir um tempo de estabilização relativamente pequeno.

Algumas séries do MSP430 (RF430) possuem um transceptor RF integrado, facilitando assim o projeto de sistemas eletrônicos que utilizam comunicação sem fio, havendo apenas a necessidade de uma antena e a realização de componentes passivos para a realização do casamento de impedância, não havendo necessidade de circuitos adicionais para comunicação entre o transceptor RF e o microcontrolador.

3.3 Placa de desenvolvimento RF-2500

A placa de desenvolvimento RF-2500 da *Texas Instruments* é uma placa de avaliação e desenvolvimento de sistemas embarcados com capacidade para comunicação sem fio. Esta placa é

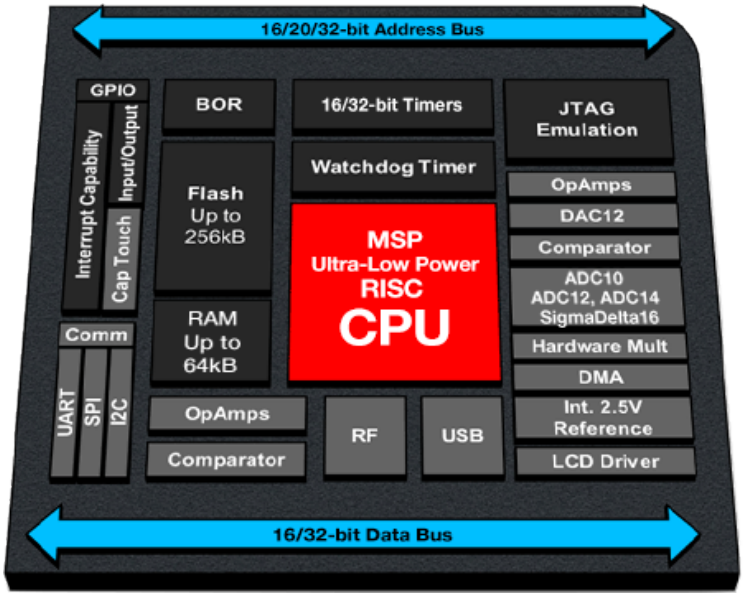

Figura 3.2: Periféricos mais comuns do MSP430 [5].

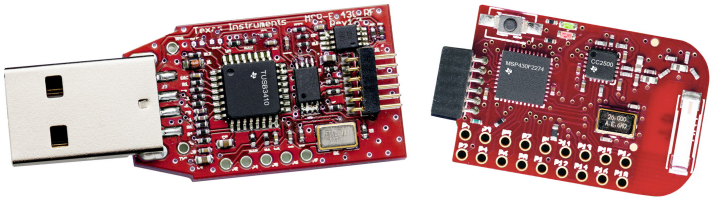

Figura 3.3: Placa de desenvolvimento RF2500 [5].

composta pelo microcontrolador MSP430 modelo F2274 de encapsulamento do tipo RHA, leds verde e vermelho, o transceptor RF CC2500, também da *Texas Instruments*, um cristal de 26 MHz e uma antena de cerâmica, além de conectores e componentes passivos (resistores, capacitores e indutores). A Figura 3.3 mostra a placa de desenvolvimento RF2500 e o seu *debugger* USB.

A programação do microcontrolador é feita pela placa *debugger* USB, conectando-a na interface *spy-by-wire* da placa RF2500. O *debugger* USB, além de ser responsável pela programação do microcontrolador, é também responsável por realizar os *breakpoints* pré-definidos antes da programação do mesmo, para propósitos de análise de funcionamento do código compilado, além de ser essenciais para o *debug* de qualquer problema na execução do programa pelo microcontrolador. Quando definidos *breakpoints* na aplicação do MSP430, a mesma é pausada em cada breakpoint, facilitando assim a resolução de problemas e o refinamento do código. O *debugger* USB também realiza comunicação serial entre o computador e o microcontrolador, podendo enviar e receber dados a uma velocidade fixa de 9600 Bauds por segundo através de um dispositivo UART (*Universal Asynchronous Receiver/Transmitter*, ou Receptor/Transmissor universal assíncrono) integrado.

Para os testes da aplicação de RSSF, é disponibilizada uma placa de bateria, no qual possui espaço para a instalação de duas pilhas do tipo AAA, além de um *jumper* como dispositivo liga-desliga.

3.3.1 MSP430F2274

O MSP430F2274 [18] é o modelo do microcontrolador incorporado à placa de desenvolvimento F2274. As principais características deste modelo são

- Variação de alimentação: 1,8 a 3,6V
- Consumo ultra baixo de energia (270 μA a 1 MHz, 2,2V; 0,7μA em modo *standby*)
- Arquitetura *RISC* de 16 bits
- Fontes selecionáveis de *Clock* de cristal externo de 32 KHz, oscilador de baixo consumo, *DCO* com frequências pré-calibradas com erro de até $\pm 1\%$
- Dois *timers* (Timer_A e Timer_B) de 16 bits
- Interface universal de comunicação serial compatíveis com os padrões UART, IrDA, SPI síncrono e I2C
- Conversor AD de 10 bit de até 200 ksps
- Dois amplificadores operacionais configuráveis
- 32 KB + 256 B de memória flash
- 1 KB de memória RAM

A placa RF2500 ainda disponibiliza conexão a diversas portas do microcontrolador, sendo possível o interfaceamento de outros componentes, tais como sensores, tanto analógicos quanto digitais, fontes externas de energia, dispositivos de posicionamento, dentre outros. Estas portas ainda contam com sinal de *clock* (gerado pelo microcontrolador), saída de referência de terra, saída e entrada de conversores analógico-digital e digital-analógico, portas digitais de entrada e saída de aplicações gerais, etc. A seleção do tipo de porta e a aplicação da mesma é configurado via software.

3.3.2 CC2500

O transceptor RF CC2500 [19] é um transceptor de baixo custo e baixo consumo, projetado para aplicações de baixa potência. O circuito é projetado para operar na faixa de 2400 a 2483,5 MHz. O CC2500 provê ainda um extenso suporte por *hardware* de manejo de pacotes, acumulação de dados, transmissões em sequência, CCA (*clear channel assessment*), indicação da qualidade do enlace, *wake-on-radio*, dentre outros.

Seus parâmetros principais de operação e os 64 *bytes* de sua pilha FIFO (*first in, first out*) de recepção/transmissão podem ser controlados via interface serial periférica (SPI, ou *Serial Peripheral Interface*). Em um sistema típico, o CC2500 é usado em conjunto com um microcontrolador.

Dentre suas características, pode-se citar as mais importantes:

- Alta sensibilidade (-104 dBm a 2,4 kBaud, 1% de taxa de erro de pacote)
- Baixo consumo de energia (13,3 mA em modo RX, 250 kBaud, entrada bem acima da sensibilidade limite)
- Faixa de frequência de 2400 a 2483,5 MHz
- Taxa de transmissão programável de 1,2 a 500 kBaud
- Interface SPI eficiente; todos os registradores podem ser programados pela SPI
- Saída RSSI (*Received Signal Strength Indicator*, indicador da potência do sinal recebido) digital
- Filtro de banda de frequência programável
- Indicador de detector de portadora (CS) programável

O arquivo de configuração e inicialização do CC2500 foi criado a partir do *software* SmartFR Studio, da *Texas Instruments*.

3.4 Protocolo de RSSF SimpliciTI

O protocolo SimpliciTI da *Texas Instruments* [20] é um protocolo de redes de rádio frequência de baixo consumo destinado a redes sem fio simples e pequenas. O protocolo foi projetado para ser de fácil implementação com o menor uso possível dos recursos do microcontrolador. O protocolo atende prontamente os microcontroladores MSP430 e vários modelos de transceptores RF.

Apesar do modesto requerimento mínimo, o protocolo SimpliciTI suporta nós em uma topologia de rede do tipo *peer-to-peer*, além de contar com a opção do ponto de acesso guardar e enviar outras mensagens. O protocolo conta também com a opção de utilizar extensores de alcance na rede.

O protocolo SimpliciTI pode ser usado numa grande variedade de aplicações de baixo consumo, incluindo sensoriamento de alarme e segurança (detectores de fumaça, detecção de quebra de janelas, sensores de monóxido de carbono, sensores de luz, dentre outros), medições automatizadas (medição de níveis de gás e de água), automação residencial (dispositivos de sensoriamento de ambiente, controle de portões de garagem), e RFID (identificação por rádio frequência) ativo. O protocolo utiliza menos de 8 KB de memória Flash e menos de 1 KB de memória RAM, dependendo da aplicação.

O protocolo [21] suporta duas topologias básicas: *peer-to-peer* pura e topologia em estrela, cujo ponto de acesso é o par de todos os nós. O ponto de acesso é usado basicamente para gerenciamento da rede. Ele suporta aplicações tais como o desligamento de nós por determinados períodos de tempo, gerenciamento da rede em termos de permissões de acesso à mesma, permissões de conexão, etc. O ponto de acesso possui também a funcionalidade de um nó, ou seja, pode interfacear sensores e atuadores na rede. Na topologia estrela, o ponto de acesso funciona como o *sink* da rede. A Figura 3.4 ilustra uma topologia possível para a rede SimpliciTI.

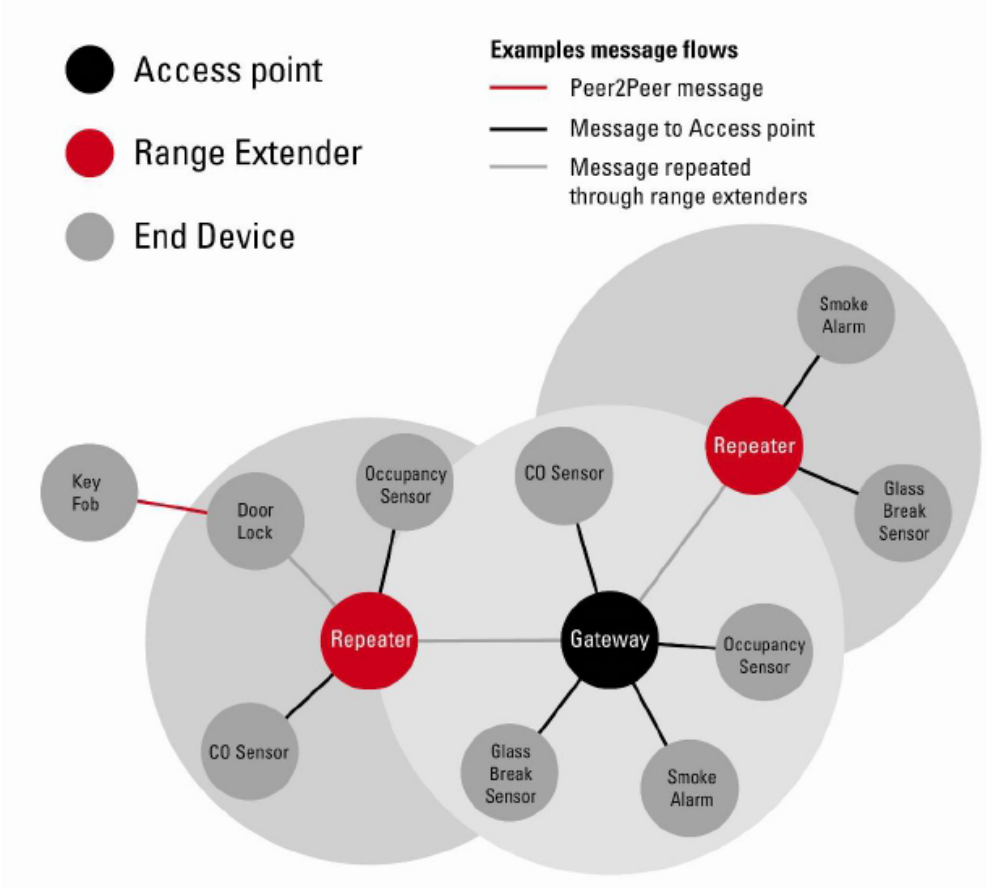

Figura 3.4: Exemplo de topologia da rede SimpliciTI [6].

O protocolo realiza somente conexões com topologias do tipo *peer-to-peer*. O protocolo não possui mecanismos formais de roteamento. O ponto de acesso (se existente na rede) oferece o suporte necessário para gerenciar a rede. Este suporte inclui acesso à rede e funções de gerenciamento de rede, como por exemplo troca de canal.

O funcionamento do protocolo é feito em um pequeno número de chamadas realizadas pela API. A associação entre duas aplicações é feita durante o funcionamento do nó. O processo de ligação cria uma conexão baseada em quais conjuntos de aplicações que podem enviar mensagens. Quando uma conexão é estabelecida, é definida como uma conexão bidirecional.

3.4.1 Objetos SimpliciTI

Os objetos da rede SimplicitTI são objetos definidos pelo seu *software*. Os três objetos SimpliciTI são: nós (*end devices*), pontos de acesso (*access points*) e extensores de alcance (*range extenders*). Apenas a programação diferencia os objetos. Como consequência, múltiplos objetos possam ser criados utilizando somente uma única plataforma de *hardware*. Por exemplo, uma plataforma que funciona como um ponto de acesso pode funcionar também como um nó. Entretanto, a maioria dos nós funcionará somente como um nó, e não como um ponto de acesso e um nó.

3.4.1.1 Ponto de acesso (*Access Point - AP*)

O ponto de acesso (AP), quando existente na rede, funciona como como um sorvedouro. Por isso, são dispositivos que estão permanentemente ligados. Somente um ponto de acesso é permitido por rede. Uma plataforma pode atuar tanto como ponto de acesso como um nó. Esta plataforma pode interfacear sensores e atuadores e atuar como ponto de acesso simultaneamente. Pontos de acesso recebem todos os pacotes num certo alcance. Além disso, os pontos de acesso reenviam pacotes não destinados a si mesmos para extender o alcance dos nós da rede.

3.4.1.2 Nó (*End Device - ED*)

Os nós são os dispositivos mais simples da rede simpliciTI. Eles são conectados à maior parte dos sensores e atuadores da rede. As plataformas que funcionam somente como nós podem ser alimentadas por baterias, diferente dos pontos de acesso e de extensores de alcance. Além disso, possuem a capacidade de desligarem por um determinado período de tempo, afim de economizar energia, aumentando a vida útil do nó.

3.4.1.3 Extensor de alcance (*Range Extender - RE*)

Os extensores são destinados a aumentar o alcance de comunicação da rede. São dispositivos que estão sempre ligados. Sua principal função é repassar pacotes, aumentando efetivamente a esfera de influência do nó de origem do pacote em questão. O protocolo SimpliciTI atualmente suporta até quatro extensores de alcance por rede.

3.4.2 Funcionamento básico da rede

O funcionamento básico da rede se dá primeiramente pela inicialização do ponto de acesso, transmitindo uma *splash screen* para o computador a ele conectado, através da porta COM (emulada através da porta USB cujo ponto de acesso está conectado). O ponto de acesso gerencia a rede e está sempre ligado, recebendo dados de um ou mais nós da rede SimpliciTI. Os nós da rede estão conectados aos sensores que são ligados à rede e passam a maior parte do tempo no modo de baixo consumo 3 (LPM3, ou Low Power Mode 3) [7]. Neste modo, a CPU, o DCO e todos os geradores de *clocks*, exceto o gerador de *clock* auxiliar ACLK (responsável pela contagem do TIMER A, que ativa a interrupção responsável por deixar o microcontrolador no modo ativo (AM, ou *active mode*), são desligados. A Figura 3.5 mostra a taxa de consumo de cada modo de operação do MSP430.

Logo depois, o ponto de acesso é inicializado como o *sink* da rede. Depois de completo o procedimento de inicialização, o ponto de acesso utiliza o conversor analógico digital ADC10 para medir a temperatura ambiente uma vez por segundo para transmiti-la ao computador por meio da porta serial. Enquanto isso, o ponto de acesso "escuta" continuamente por novos nós querendo se conectar à rede e por pacotes de nós já conectados. Usando os dois LEDs como indicadores, o ponto de acesso notifica o usuário sobre duas atividades: o LED vermelho indica a transmissão

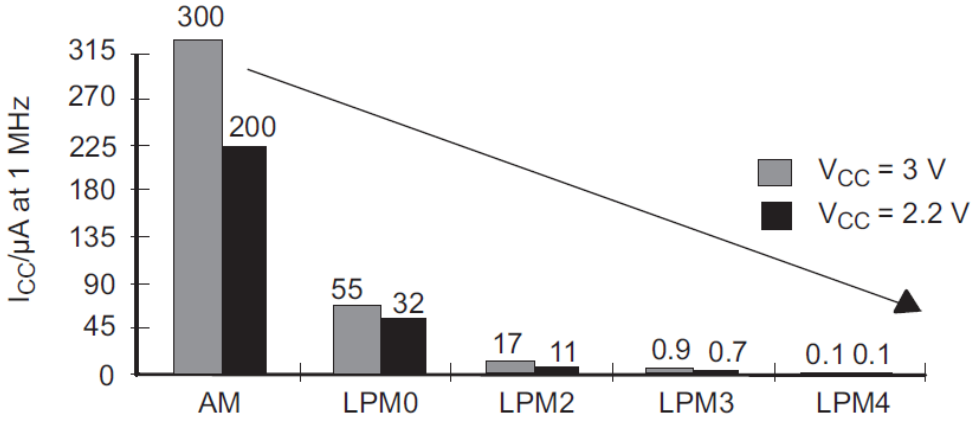

Figura 3.5: Modos de operação do MSP430 e suas respectivas taxa de consumo de energia [7].

```
Node:HUB0,Temp: 91.2F,Battery:3.5V,Strength:000%,RE:no
$HUB0, 89.6F,3.5,000,N#
```

Figura 3.6: Dados enviados ao computador pela porta serial [7].

dos dados do ponto de acesso ao computador, enquanto que o LED verde indica o recebimento do pacote de um dos nós da rede.

Assim que o nó é ligado, começa imediatamente a procurar por um ponto de acesso para se conectar. Enquanto procura, ambos os LEDs piscam, indicando a procura. Assim que é conectado ao ponto de acesso, o nó tenta se conectar à rede, piscando o LED vermelho para indicar a tentativa de conexão. Se não pode se conectar ao ponto de acesso, ele continua a piscar o LED vermelho. Uma vez conectado à rede através do ponto de acesso, todos os LEDs desligarão e o nó entrará em LPM3, piscando o LED verde somente quando ativo (em AM).

Em plena operação, o ponto de acesso envia ao computador um conjunto de caracteres que informam sobre a medição realizada pelos sensores de um determinado nó, além de informações acerca do funcionamento da rede. A Figura 3.6 ilustra um exemplo de dados enviados ao computador. A primeira linha ilustra os dados enviados ao computador no modo *verbose*. Neste modo, todos os dados são amplamente nomeados, sendo:

- Node: identificação do nó em questão. HUB0 indica o ponto de acesso. O número do nome de cada nó é dado de acordo com a ordem de conexão do nó à rede.
- Temp: temperatura medida pelo nó. Apertando a tecla "C" no teclado do computador, a temperatura é vista em graus Celcius. Apertando "F", a temperatura volta a ser mostrada em Farenheit.
- Battery: indica o nível da tensão da bateria do nó em questão. A tensão é aferida pelo conversor ADC10 do nó.
- Strength: indicador da força do sinal recebido (RSSI) pelo transceptor CC2500. É dado em

porcentagem para melhor visualização. O RSSI do ponto de acesso é sempre 0%, por não haver comunicação entre nós.

No modo *Minimal* (acessível apertando a tecla "M" do teclado do computador) os dados enviados pela porta serial são minimizados para reduzir o tempo de envio ao computador. Este modo está ilustrado na segunda linha da Figura 3.6. Para retornar ao modo *verbose*, basta apertar a tecla "V". As informações são as mesmas do modo *verbose*, e são separadas por vírgula, além de cada linha de informação começar com $ e terminar com #.

3.5 Procedimentos de backoff implementados

O procedimento de *backoff* ocorre quando, através do *CCA*, o transceptor verifica se o canal a ser utilizado para o envio de pacotes está ocupado. Neste momento, dependendo do procedimento, o nó realiza uma espera de tempo aleatório cujos valores mínimo e máximo são definidos pelo próprio procedimento de *backoff* implementado. Além disso, ele ainda define o máximo número de vezes que o nó irá repetir a espera pelo canal livre para tentar a retransmissão de um dado pacote de dados.

Os procedimentos de *backoff* implementados sobre o protocolo de acesso ao meio (ou MAC, *Medium Access Control*) de redes SimpliciTI são o principais aspectos deste trabalho. Além do procedimento de *backoff* do próprio SimpliciTI já implementado, foi implementado também o procedimento de *backoff* do protocolo IEEE 802.15.4. Por serem vitais para a garantia do envio de pacotes ao ponto de acesso, os procedimentos de *backoff* são de extrema importância para o bom funcionamento da rede, além de serem responsáveis por boa parte da economia de energia dos nós.

Uma visão detalhada de ambos os procedimentos está descrita nas seções seguintes.

3.5.1 Protocolo CSMA/CA do SimpliciTI

Esquema nativo do SimpliciTI, o protocolo CSMA/CA se baseia em uma série de ações relativamente simples com o objetivo de se ter uma otimização maior no consumo de energia.

O protocolo CSMA/CA do protocolo SimpliciTI é realizado a partir das seguintes etapas:

1. O nó realiza o CCA, verificando a disponibilidade do meio para enviar um pacote.
2. Se o meio estiver livre, o nó envia o pacote. Caso contrário, espera um tempo aleatório determinado pela função *Mrfi_RandomBackoffDelay()*, definido entre 1 e 16 μs.
3. Finalizado o tempo de espera, o nó verifica mais uma vez através do CCA a disponibilidade do meio para envio de pacotes. Se o meio estiver livre, o nó enviará o pacote. Caso contrário, realizará outra vez o primeiro item. Há um número máximo de tentativas de envio do pacote. Caso o número de tentativas seja maior que este número, o nó não tentará mais enviar o pacote e o descartará. Nos testes realizados o número máximo de tentativas de envio é igual a quatro.

A função *Mrfi_RandomBackoffDelay()* é responsável pelo cálculo do número aleatório entre 1 e 16 e pela espera, em microssegundos, do valor do número aleatório. Este valor aleatório é distribuido uniformemente. A função utiliza uma função geradora de números pseudo-aleatórios (*MRFI_RandomByte()*). Por ser pseudo-aleatório, o valor do Byte calculado por esta função irá se repetir a cada 256 Bytes. Com isso, uma certa sequência de 256 valores do tempo de espera irá se repetir continuamente.

3.5.2 Protocolo CSMA/CA do IEEE 802.15.4

O procedimento de *backoff* do protocolo IEEE 802.15.4 é nitidamente mais complexo que o procedimento do protocolo SimpliciTI. Como consequência, possui uma robustez muito maior em relação à perda de pacotes. Por ser a solução proposta pelo trabalho presente, a implementação deste protocolo CDMA/CA é comparada com o protocolo CSMA/CA nativo do SimpliciTI.

A Figura 3.7 mostra o fluxograma do protocolo CSMA/CA do tipo *non-slotted* do protocolo IEEE 802.15.4. No CSMA/CA do tipo *non-slotted* não há a definição de uma janela de contenção, ou seja, o envio de pacotes é realizado de maneira assíncrona.

Primeiramente, são definidas as variáveis NB (*número de backoffs*) e BE (expoente do *backoff*, número mínimo definido pelo protocolo IEEE 802.15.4), iguais a zero e três, respectivamente. Em seguida, o nó espera por um número aleatório de períodos de *backoff* no intervalo de $2^{BE} - 1$, dado em microssegundos. Com isso, o nó realiza a operação de detecção de portadora (CCA, *Clear Channel Assessment*), checando pela disponibilidade do meio de comunicação. Se o meio estiver livre, o nó simplesmente transmite o pacote e termina o procedimento de *backoff*. Caso contrário, a variável NB é acrescida de uma unidade, enquanto que a variável BE é acrescida de uma unidade até um número máximo igual a oito, definido também pelo protocolo. Em seguida, a variável NB é comparada com o número máximo de *backoffs*, definido pelo protocolo como um máximo de cinco *backoffs* possíveis. Se for mais que o número máximo, o nó acusará falha no envio do pacote e o descartará. Caso contrário, ele voltará para a etapa de espera por um número aleatório de períodos de *backoff*.

Para se gerar o número aleatório utilizado pelo procedimento de backoff foi utilizada a função *TI_getRandomIntegerFromVLO()* [22], uma função criada pela própria *Texas Instruments* para se gerar um número aleatório de 16 bits. Este número é gerado a partir do oscilador de baixa frequência (VLO) e do oscilador do microcontrolador. Estes osciladores são dois sistemas geradores de *clock* independentes, tendo cada um sua fonte de oscilação. Por isso, a borda de subida do sinal de cada um varia de forma diferente uma da outra. Esta característica pode ser explorada para se gerar uma sequência de bits aleatórios. Além disso, o número a ser gerado é desconhecido, mesmo se sabendo o número gerado anteriormente. Esta técnica de geração de números aleatórios é mais robusta que a técnica utilizada no protocolo SimpliciTI, embora consuma mais energia por ter um número maior de instruções a serem executadas.

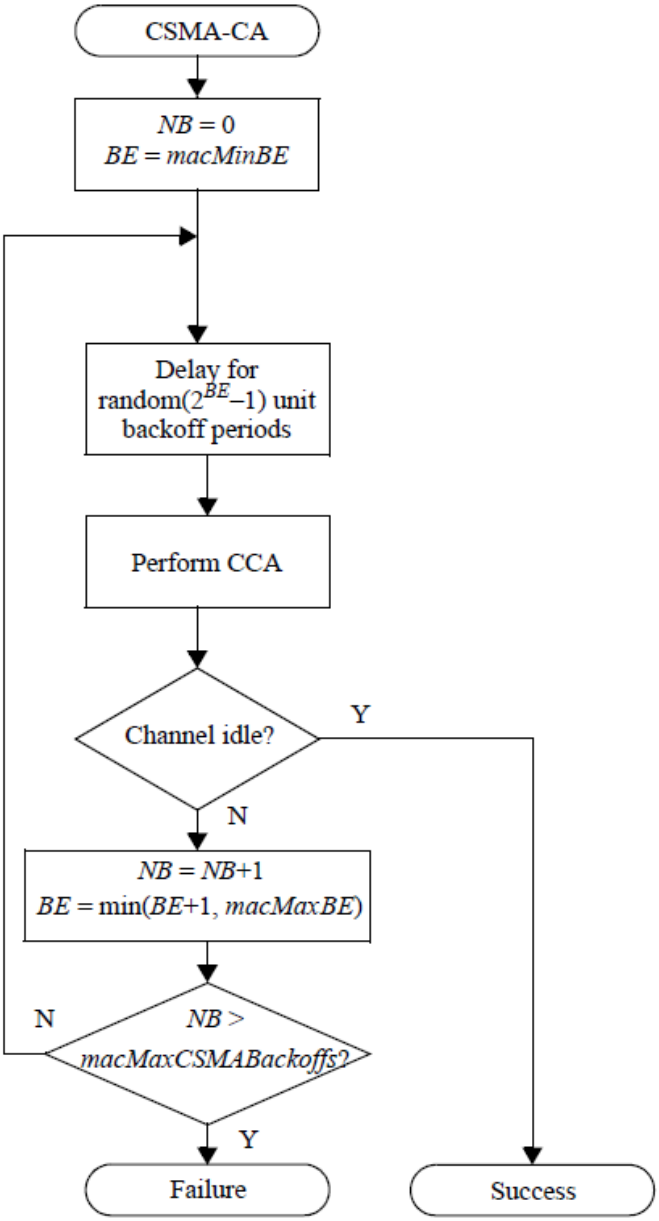

Figura 3.7: Protocolo CSMA/CA do protocolo IEEE 802.15.4 para o caso de tempo não compartimentado [8].

Capítulo 4

Avaliação de Desempenho

O capítulo presente aborda a metodologia utilizada para a execução dos experimentos referentes aos procedimentos de backoff dos protocolos SimpliciTI e IEEE 802.15.4. Além disso, faz a avaliação e comparação entre os procedimentos adotados, bem como uma análise qualitativa sobre as ferramentas utilizadas para tais experimentos.

4.1 Introdução

A plataforma SimpliciTI traz consigo um protocolo de comunicação próprio, descrito anteriormente neste trabalho. Ao estudar-se a estrutura da camada de rede nativa da plataforma, percebeu-se que alguns pontos poderiam ser aperfeiçoados, principalmente no que diz respeito ao protocolo de controle de acesso ao meio. Foi observado que o protocolo MAC nativo do SimpliciTI é baseado em contenção, porém a janela de contenção máxima implementada não era tão grande, 16 μs; este fato implica em uma maior perda de dados para uma rede muito congestionada ou com muitos nós disputando o meio, uma vez que quando o número máximo de tentativas de envio é extrapolada, o pacote é descartado e o contador de tentativas é reiniciado.

Buscando na literatura sobre protocolos de controle de acesso ao meio, constatou-se uma possível opção para implementação alternativa, adequando-se ao hardware oferecido pela plataforma: IEEE 802.15.4. Este protocolo opera de maneira semelhante ao SimpliciTI, porém com uma janela de contenção variável e um número máximo de tentativas maior, o que teoricamente garante uma maior confiabilidade na entrega de pacotes, mesmo com uma rede mais congestionada ou com um número maior de nós.

Neste trabalho, foi implementado na plataforma da *Texas Instruments* o protocolo de controle de acesso ao meio baseado nas diretrizes do protocolo MAC IEEE 802.15.4, para fins de comparação de desempenho com o protocolo nativo SimpliciTI.

4.2 Experimentos e aquisição de dados via MATLAB

O estudo do protocolo de comunicação SimpliciTI nativo da plataforma apresentou alguns pontos que foram julgados negativos para a performance da rede. A maneira como o sistema reagia ao meio congestionado, no caso o mecanismo de determinação da janela de contenção e do tempo aleatório de espera para nova tentativa de envio de pacotes, levantou suspeitas em relação ao desempenho da rede para situações de maior estresse na comunicação, como uma rede com muitos nós ou com um tráfego intenso de dados. Tendo em vista este problema, buscou-se na literatura alguma alternativa que pudesse ser implementada na plataforma e que tivesse uma resposta melhor ao meio congestinoado. Assim, a implementação proposta foi baseada no funcionamento do protocolo CSMA/CA do padrão IEEE 802.15.4, que prevê um incremento exponencial no tempo de espera aleatório a cada nova tentativa de envio de pacotes pelos nós. Para observar a performance dos dois protocolos, foi determinado um procedimento de teste para avaliação do protocolo nativo e do protocolo proposto. A ideia do teste é analisar a quantidade de tentativas de reenvio de pacotes realizadas pelos nós, além da perda de pacotes observada na transmissão. Assim, foram determinados os parâmetros que seriam variáveis e constantes na observação do funcionamento dos sistemas. Para observarmos a taxa de sucesso de recebimento de pacotes pelo ponto de acesso, a vazão, foi mantido fixo em 1000 o número de pacotes a serem enviados por cada nó durante cada experimento. As variáveis controladas durante os experimentos foram as seguintes:

- Número de nós da rede em cada experimento;
- Taxa de envio de pacotes por nó em cada experimento.

As taxas escolhidas para a realização dos experimentos foram as seguintes: 2, 5, 10, 40 e 120 pacotes por segundo. Os pacotes da rede estavam possuíam 19 Bytes, sendo 10 Bytes referentes à carga útil, parcela do pacote que carrega a informação de fato, e 9 bytes referentes ao cabeçalho do pacote.

Como havia somente 10 sensores disponíveis, foram realizados experimentos variando o número de sensores nas seguintes configurações: 2, 5, 7 e 10 sensores. A taxa de geração de pacotes de cada nó também era uma variável que podia ser ajustada; como não havia mais nós disponíveis para estressar a rede, este foi um artifício encontrado para colocar a rede em uma situação com tráfego mais intenso de dados. As taxas escolhidas para realização dos experimentos foram as seguintes: 2, 5, 10, 40 e 120 pacotes por segundo. A cada teste, cada nó envia 1000 pacotes ao ponto de acesso.

Com a realização dos experimentos, buscou-se observar alguns aspectos. Primeiramente, observar a resposta de cada protocolo individualmente em relação à variação do número de nós na rede e à variação da taxa de envio de pacotes. Depois, comparar o rendimento dos protocolos para as mesmas condições, a fim de determinar qual dos protocolos apresenta melhor rendimento para operação em situações críticas, com tráfego intenso ou rede congestionada.

4.2.1 Realização dos experimentos

Para realizar os experimentos, foi necessário gravar o programa com o protocolo a ser testado em cada um dos dez nós para cada taxa de dados a ser analisada. Para isso, foi utilizado o programa *Code Composer Studio* versão 5.5, da *Texas Instruments*. Não foi utilizada uma versão mais recente do programa por motivos de compatibilidade do código do protocolo SimpliciTI. Dessa forma, a variável que guardava o valor da taxa de envio de pacotes podia ser modificada e então a rede passava a operar com a taxa desejada. Ao total, foram realizados quarenta experimentos para cobrir todas as possibilidades, combinando as quatro configurações de nós e as cinco taxas de envio de pacotes. O experimento consistiu em ligar a rede em cada uma das 40 configurações desejadas e colher os dados de tentativas de reenvio de dados dos nós. A cada teste realizado, o ponto de acesso era conectado ao computador e em seguida cada sensor era conectado à rede; dessa forma, a aquisição dos dados era realizada após cerca de 1 minuto da ativação de todos os nós, para garantir a conexão dos mesmos com o ponto de acesso. Foi utilizado um computador com o software Matlab instalado para gravar os dados enviados pelos nós; os nós foram dispostos ao redor do computador de maneira aleatória, alcançando distâncias de até 3 metros do ponto de acesso, de maneira a obter uma topologia do tipo estrela para a rede. Esta distância máxima foi alcançada devido à disponibilidade de espaço no ambiente de teste, que foi a casa de um dos autores do trabalho. A documentação da *Texas Instruments* indicou que não há uma distância máxima teórica para funcionamento dos nós, mas que resultados empíricos apontaram uma distância máxima de 50 metros entre nós com funcionamento razoável. Como as implementações apresentavam topologia de rede em estrela, buscou-se manter o ponto do acesso centralizado em relação aos sensores durante os experimentos, realizando uma distribuição aleatória dos sensores no ambiente. É válido ressaltar que o ambiente dos experimentos estava carregado de dispositivos sem fio que operavam na mesma faixa de frequência da RSSF a ser testada. A Figura 4.1 mostra a captura de tela da aplicação *WiFi Analizer*, quando executada no ambiente do experimento. Esta condição deu ao experimento uma característica próxima da condição de operação real de uma RSSF, onde muitas vezes os sensores precisarão disputar o meio com outros dispositivos que operam na mesma banda de frequência. Realizar o experimento em "condições ideais" poderia mascarar o real desempenho dos protocolos em questão.

Primeiramente foram realizadas as medições com a implementação do *backoff* do protocolo IEEE 802.15.4; a área de trabalho do MATLAB referente ao experimento, o registro de dados da leitura da porta serial e o número de *backoffs* realizados por cada nó foram salvos ao final de cada experimento para análise posterior. O número de backoffs contabilizado é referente ao último pacote enviado com sucesso, pois o pacote seguinte que chega com sucesso traz a informação do número de tentativas de envio do pacote anterior. Quando o número de tentativas de envio excede o máximo estipulado, o pacote é descartado e o próximo pacote que chega com sucesso traz a informação de que o pacote anterior foi descartado. Ao finalizar esta bateria de experimentos, passamos para as medições com a implementação nativa do SimpliciTI; assim como na bateria anterior, a área de trabalho do MATLAB, o registro de dados da leitura da porta serial e o número de *backoffs* realizados por cada nó foram salvos. Os dados obtidos nos experimentos com a implementação do *backoff* do protocolo IEEE 802.15.4 representaram o número de backoffs realizados por cada nó

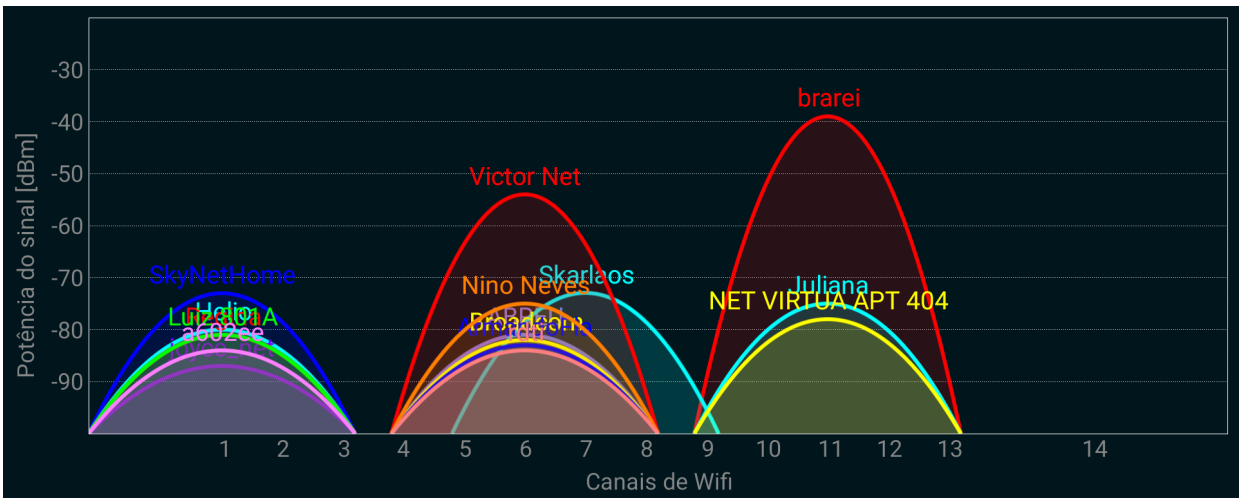

Figura 4.1: Redes WI-FI disponíveis e suas respectivas potências no ambiente dos experimentos. Imagem obtida a partir do programa *WI-FI Analyzer* para sistemas operacionais Android.

ao longo da transmissão de 1000 pacotes, podendo assumir valores de 0 a 5; quando o número de *backoffs* NB ultrapassa 5, esta variável é zerada pelo código implementado nos sensores, indicando que houve perda do pacote em questão, e com o consequente incremento da variável que indica o número de falhas por nó. No caso da implementação nativa do SimpliciTI, a variável em questão CR (CCAs restantes) é decrementada, iniciando com valor 4; quando ela é menor que zero, seu valor é ajustado para 4 novamente e a variável que guarda o valor de falhas de envio de pacotes por nó é incrementada.

A aquisição dos dados dos sensores durante os experimentos foi realizada utilizando o software MATLAB, pois foi necessário buscar uma plataforma que pudesse receber e gravar os dados recebidos dos sensores. O MATLAB oferece uma plataforma que possibilita a leitura de dados da porta serial do computador; esta característica foi de extrema importância para a etapa de aquisição de dados, uma vez que os dados recebidos dos sensores pelo ponto de acesso eram transmitidos ao computador através da porta USB. Cada nó enviava uma mensagem com uma identificação própria (número do nó correspondente), temperatura medida em graus Celsius, nível de tensão da bateria em Volts, nível de potência do sinal em porcentagem em relação à potência total de envio, número de *backoffs* (NB) realizados e número de falhas (PF, ou *packet fail*) de envio. A Figura 4.2 contém um exemplo de uma mensagem enviada pelo nó 2 durante um experimento com implementação IEEE 802.15.4. A mensagem lida pela porta serial carrega as informações do pacote enviado anteriormente pelo nó; dessa forma, quando ocorre a leitura de NB = 1, por exemplo, quer dizer que o pacote anterior foi enviado com sucesso após ter que recuar uma vez um intervalo de tempo aleatório devido ao canal estar ocupado.

$0002, 22.2C, 3.1, 032, NB:0, PF:0.

Figura 4.2: Exemplo de uma mensagem enviada pelo nó 2 durante um experimento com implementação IEEE 802.15.4.

Um programa em MATLAB foi desenvolvido para gravar os dados lidos da porta serial durante o experimento e separar a informação referente ao número de *backoffs*, no caso da implementação IEEE 802.15.4, ou ao número de *CCA Retries* (tentativas de avaliação de disponibilidade do canal), no caso da implementação nativa do SimpliciTI. Estes dados, individualizados para cada nó em cada experimento, foram gravados em arquivos de texto para análise posterior. O ponto de acesso recebia as informações dos nós e as passava através da porta serial para o computador na forma de *string*, descrita na figura 4.2. O código do MATLAB analisava cada *string* enviada pelo ponto de acesso, gravando apenas a informação relativa ao número de *backoffs* ou ao número de *CCA Retries* referentes ao teste realizado, gerando como saída arquivos de texto com somente a informação do número de tentativas de reenvio de cada um dos nós da rede. Estes dados foram posteriormente alocados em um arquivo Excel para geração de gráficos para facilitar a observação e análise dos resultados obtidos. Com os resultados dos testes, são abordadas duas perspectivas distintas. A primeira visa observar o comportamento isolado de cada uma das implementações de acordo com a variação do número de nós e de taxa de transmissão de dados; já a segunda busca realizar uma comparação de desempenho das duas implementações, analisando o comportamento das duas implementações para as mesmas condições de experimentos.

4.3 Comparação e discussão dos resultados

A primeira análise a ser feita é a comparação de desempenho dentro de cada uma das implementações. As Figuras 4.3 e 4.4 apresentam os resultados referentes ao número de recuos realizados pelos protocolos IEEE 802.15.4 e SimpliciTI, respectivamente,para diferentes taxas de envio de pacotes e para diferentes números de nós na rede.

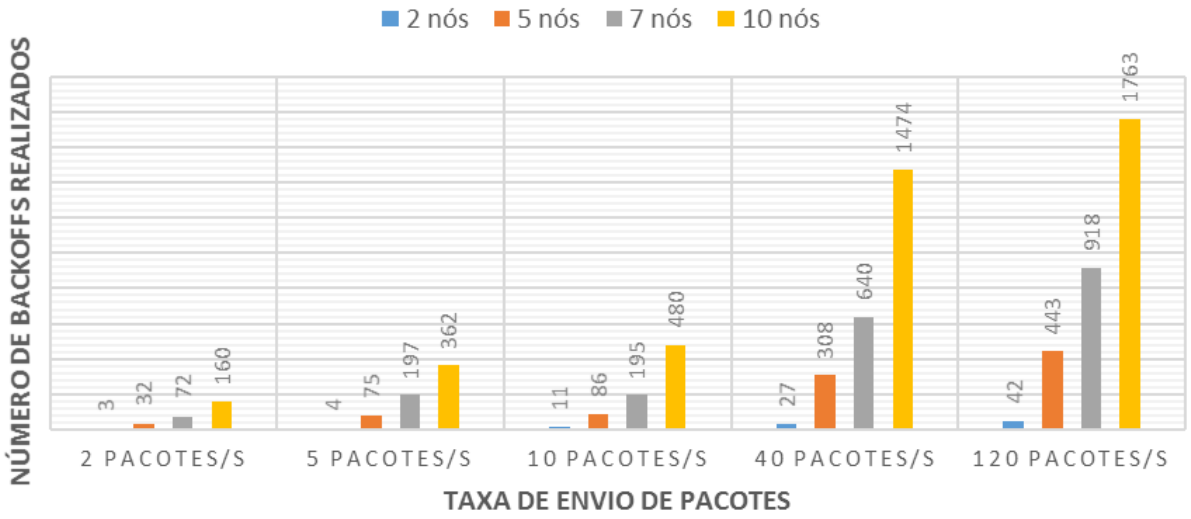

Figura 4.3: Número de *backoffs* realizados em cada experimento com o procedimento de backoff do protocolo IEEE 802.15.4.

Analisando os dois gráficos, percebe-se que a resposta das duas implementações para baixas

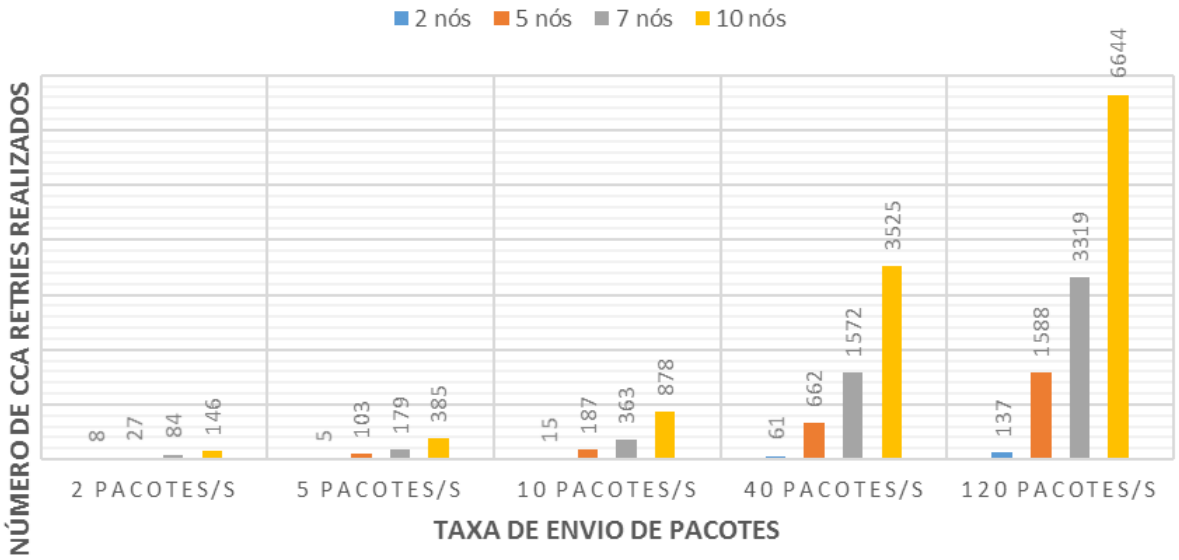

Figura 4.4: Número de *backoffs* realizados em cada experimento com o procedimento de backoff do protocolo SimpliciTI.

taxas de transmissão de pacotes, no caso 2 e 5 pacotes por segundo, são muito parecidas. Nesta configuração, mesmo com dez nós, a rede não se encontra congestionada o bastante para que seja evidenciada uma diferença no desempenho dos dois protocolos, logo as duas implementações garantem a transmissão de todos os dados sem ter de realizar o procedimento de *backoff* muitas vezes para efetuar a transmissão de um pacote. A título de ilustração, de dez mil pacotes enviados, houve apenas 385 tentativas de reenvio por canal congestionado no pior cenário da implementação SimpliciTI neste conjunto de parâmetros considerado (10 nós, 5 pacotes por segundo), sem registro de perda de pacotes; na implementação IEEE 802.15.4, para as mesmas condições, observou-se 362 tentativas de reenvio, também sem registro de perda de dados.

Nos mesmos gráficos foi verificado que, para taxas de envio de pacotes maiores, no caso de 10, 40 e 120 pacotes por segundo, cenários que representam um canal mais congestionado com um maior tráfego de dados e maior chance de o canal estar ocupado, as duas implementações começaram a apresentar comportamentos diferentes. Com o aumento na taxa de geração de dados, era esperado que o número de *backoffs* realizados pela rede aumentasse, a fim de que colisões e perda de dados fossem evitadas. Observou-se que a implementação SimpliciTI teve uma resposta pior em relação à implementação IEEE 802.15.4, em relação ao número de *backoffs* realizados. Na implementação SimpliciTI, para os experimentos com 5, 7 e 10 nós para as taxas de 40 e 120 pacotes por segundo, houve um grande número de tentativas de reenvio de pacotes. A implementação IEEE 802.15.4, por sua vez, apresentou um número grande de *backoffs* realizados, porém quase quatro vezes menor que o número de *backoffs* realizados pelo SimpliciTI no pior cenário.

Complementar à análise do número de *backoffs* realizados, a observação do número de falhas de envio de pacotes durante a transmissão é de extrema importância para a análise de desempenho da rede, pois uma rede que apresenta perda de dados apresenta baixa confiabilidade. Para ilustrar

melhor a diferença de comportamento dos dois protocolos, as Figuras 4.5, 4.6 e 4.7 comparam o número de *backoffs* realizados por cada protocolo para as três taxas de envio de pacotes que apresentaram resultados críticos.

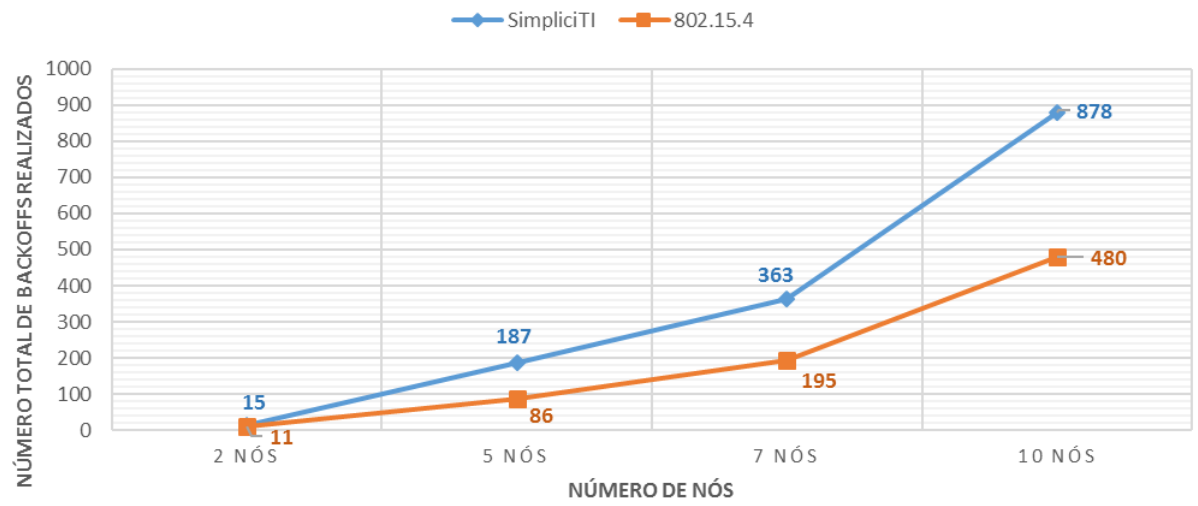

Figura 4.5: Comparativo entre o número de backoffs realizados a uma taxa de 10 pacotes por segundo.

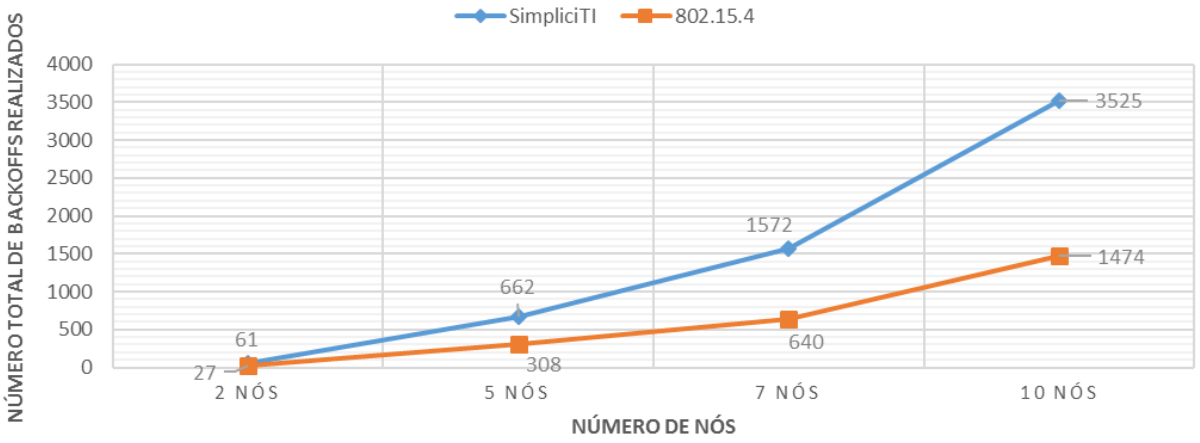

Figura 4.6: Comparativo entre o número de backoffs realizados a uma taxa de 40 pacotes por segundo.

Da análise de dados obtidos através dos experimentos, foi constatado que apenas na implementação SimpliciTI houve perda de dados nas situações de rede congestionada, onde havia mais nós e a taxa de transmissão de dados era alta. Este comportamento dos dados condiz com o procedimento de backoff implementado no SimpliciTI; a janela fixa aumenta as chances de um mesmo número de slots de tempo ser sorteado por dois ou mais nós durante a contenção, quando a rede em questão está com muitos nós ou com uma taxa de envio de dados elevada. Como o número de tentativas era limitado a 4 tentativas, é esperado que o SimpliciTI descarte pacotes por não

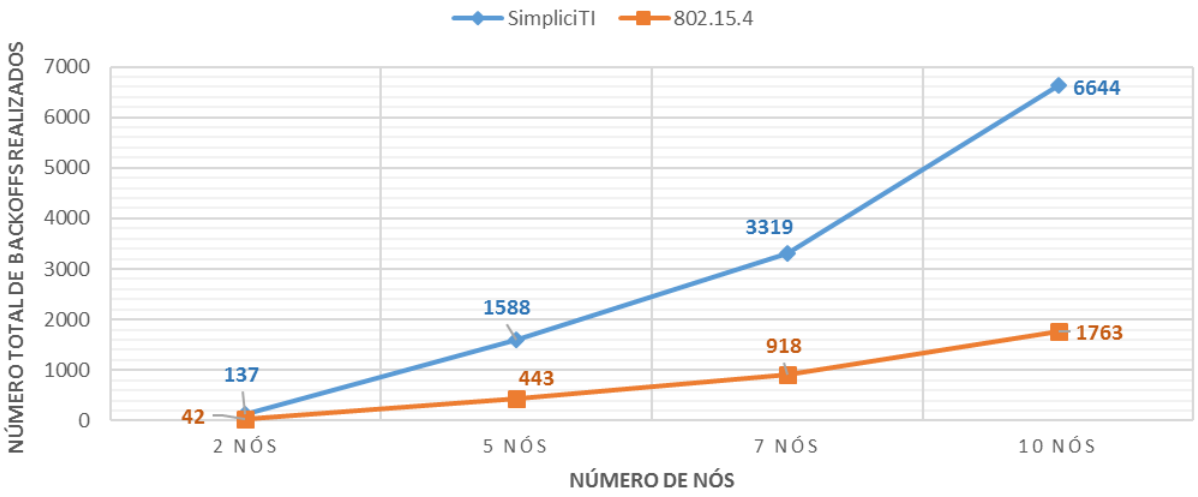

Figura 4.7: Comparativo entre o número de backoffs realizados a uma taxa de 120 pacotes por segundo.

conseguir fazer com que os nós vençam a contenção. As Figuras 4.8 e 4.9 apresentam os resultados das perdas de dados observadas.

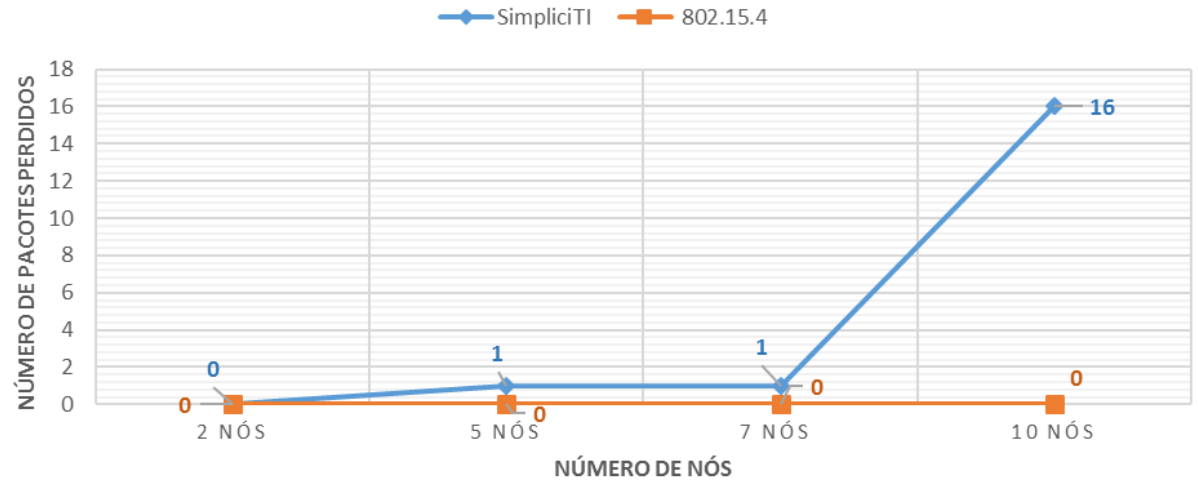

Figura 4.8: Número de pacotes perdidos a uma taxa de transmissão de 40 pacotes por segundo.

Em nenhum cenário a implementação do procedimento de *backoff* do protocolo IEEE 802.15.4 apresentou falhas no envio de seus pacotes, respondendo bem às adversidades da rede e conseguindo enviar seus pacotes dentro da contenção. Foi possível observar também que, para situações com maior estresse na rede, ou com um meio muito disputado, a implementação SimpliciTI teve muita dificuldade em lidar com a contenção, levando à perda de dados. Para tornar mais explícita a relação entre o desempenho desta implementação, a taxa de envio de dados e o número de nós da rede, comparou-se as perdas de pacote para as diferentes taxas de envio, como mostra o gráfico 4.10.

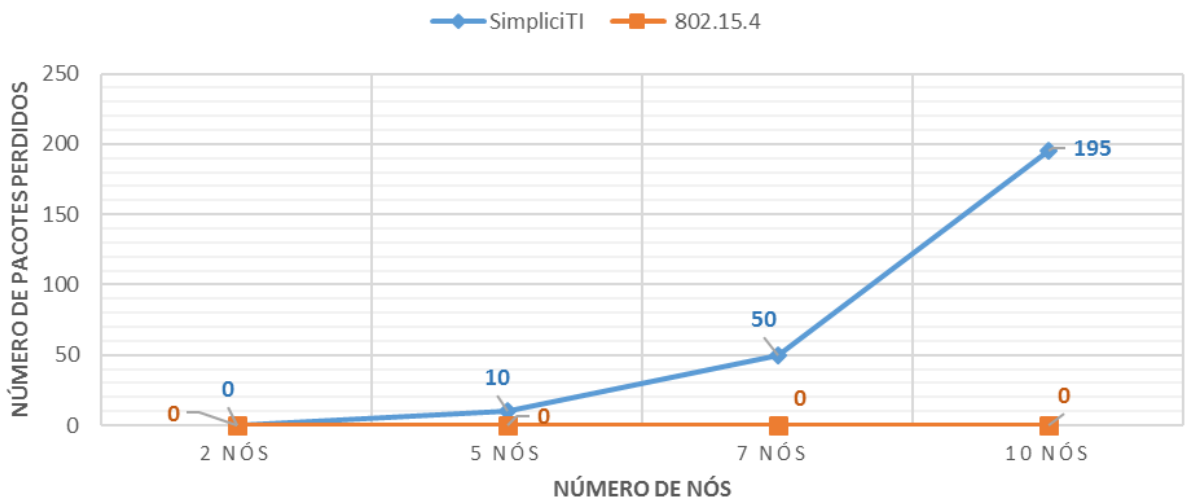

Figura 4.9: Número de pacotes perdidos a uma taxa de transmissão de 120 pacotes por segundo.

Dentro dos parâmetros que determinam a qualidade de uma RSSF, temos o conceito de vazão, que consiste na taxa efetiva de transmissão de dados até o ponto de acesso. Da configuração da rede, tem-se que cada pacote possui 19 Bytes, que equivalem a 152 bits. Para analisar o vazão da rede nos casos onde houve perda de dados, a seguinte análise será feita: transforma-se a taxa de transferência de pacotes de pacotes por segundo para bits por segundo, multiplicando a taxa em questão pelo tamanho do pacote em bits; descobre-se a porcentagem da perda de pacotes em relação ao envio total de pacotes de cada experimento e multiplica-se este fator percentual pela taxa de envio de pacotes. As Figuras 4.11 e 4.12 apresentam uma comparação entre a vazão efetiva e dos protocolos SimpliciTI e IEEE 802.15.4 para os casos em que as taxas de geração de pacotes é de 40 e 120 pacotes/s, respectivamente. Observas-se que a vazão do CSMA baseado no IEEE 802.15.4 mantém a vazão efetiva igual à taxa de geração de bits (18240 b/s) indiferente ao aumento do número de nós, enquanto que a versão original do SimpliciTI já apresenta deterioração na vazão, com uma perda de 2% para uma rede pequena (10 nós), a uma taxa de geração de bits relativamente baixa (18240 bits/segundo).

Com a análise destes dados, pode-se concluir que, apesar do baixo número de nós disponíveis para se verificar o desempenho das implementações para redes mais extensas, ainda assim foi possível estressar a rede, congestionando o meio ao aumentar o tráfego de pacotes; desta forma, a implementação do *backoff* do protocolo IEEE 802.15.4 pôde ser testada e avaliada para fins de análise de desempenho e comparação com a implementação nativa SimpliciTI.

Ao estudar o protocolo de comunicação nativo do sistema da *Texas Instruments*, percebeu-se que o processo de geração do *backoff* no protocolo de controle de acesso ao meio implementado não estava otimizado, pois em caso de o meio estar ocupado na iminência de envio de pacote, o nó em questão elegia um intervalo de tempo aleatório, dentro de uma janela de tamanho fixo, para esperar e então analisar o meio novamente para tentar reenviar o pacote. O tamanho da janela é relativamente pequeno, além de o sistema estar ajustado para realizar apenas mais 4 tentativas

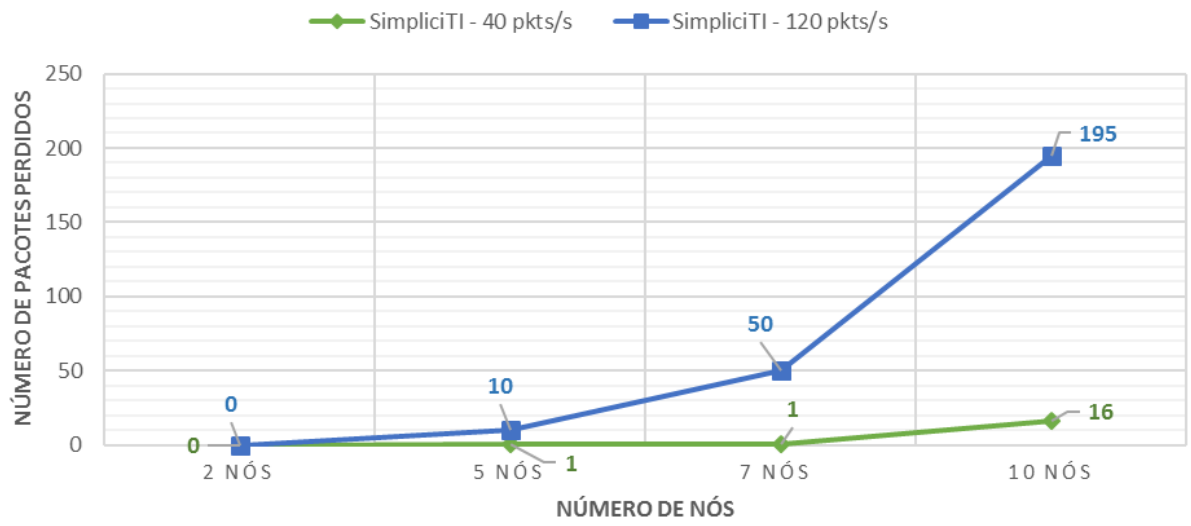

Figura 4.10: Comparação da perda de pacotes com o procedimento de backoff do protocolo SimpliciTI.

antes de descartar o pacote a ser enviado. Esse conjunto de fatores já indicava que para uma situação de rede congestionada, onde o meio estaria ocupado por mais tempo, a transmissão de pacotes poderia ser prejudicada.

Para tentar melhorar este panorama, buscou-se na literatura estratégias que pudessem tornar o procedimento de *backoff* mais eficiente e que, ao mesmo tempo, pudesse ser implementado no hardware do sistema da *Texas Instruments*. Assim, chegou-se ao procedimento de *backoff* do protocolo de controle de acesso ao meio do protocolo IEEE 802.15.4, que flexibiliza a janela de contenção, aumentando as possibilidades de alocação aleatória do envio de pacotes após a percepção do meio congestionado.

Com os experimentos realizados, foram levantados dados sobre o número de *backoffs* realizados pelas duas implementações, além da informação referente à perda de pacotes. Em sistemas com poucos nós ou operando com baixas taxas de transmissão de dados, as duas implementações apresentaram comportamento muito semelhante, o que indica que, para aplicações que não necessitem de muitos nós ou que a frequência de transmissão de dados não seja alta, os dois protocolos são opções viáveis para a implementação da rede. No entanto, ao aumentarmos o número de nós e a frequência do envio de dados, observamos uma grande diferença no comportamento das duas implementações. Como era esperado, a janela fixa do procedimento de *backoff* da implementação nativa SimpliciTI prejudica muito o funcionamento da rede, elevando o número de *backoffs* realizados e implicando em perda de dados, como era esperado. Dos resultados obtidos nos experimentos, pode-se observar uma correlação entre a taxa de envio de pacotes, o número de nós na rede e o número de *backoffs* realizados pela implementação SimpliciTI; quanto maior for a rede ou quanto maior for o nível de contenção no meio, maior será o número de *backoffs* realizados pelo sistema e consequentemente maior será a perda de dados.

A implementação do procedimento de *backoff* do protocolo IEEE 802.15.4, por sua vez, mostrou-

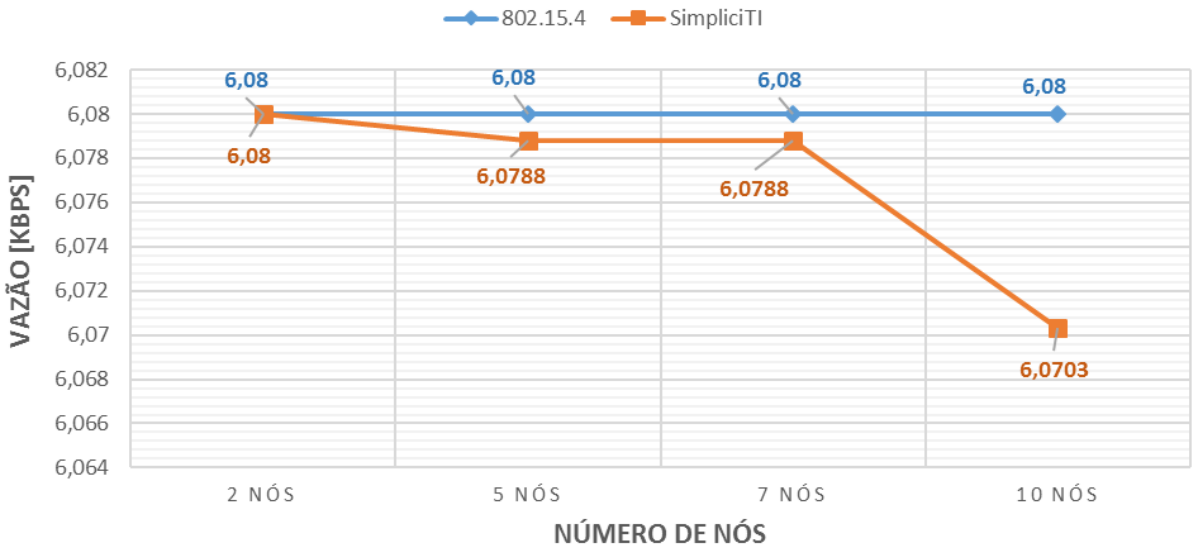

Figura 4.11: Throughput (kbps) para uma taxa de 40 pacotes por segundo.

se robusto perante os experimentos realizados. Assim como na implementação SimpliciTI, foi possível observar uma correlação entre a taxa de envio de pacotes, o número de nós na rede e o número de *backoffs* realizado; porém, esta correlação é muito menor comparada com a correlaçãp da implementação anterior. Além disso, em todos os experimentos realizado com esta implementação, não foi registrada nenhuma perda de pacote. Assim, é possível concluir que a implementação sugerida é mais robusta que a implementação nativa, podendo ser expandida para mais nós sem prejuízo ao fluxo de dados da rede; para aplicações em que a taxa de entrega de dados seja um fator determinante, a implementação do backoff do protocolo MAC IEEE 802.15.4 é a opção mais segura para garantir que não haja perda de dados, pois a rede consegue lidar com facilidade com a contenção no meio, mesmo com condições de tráfego intenso de dados.

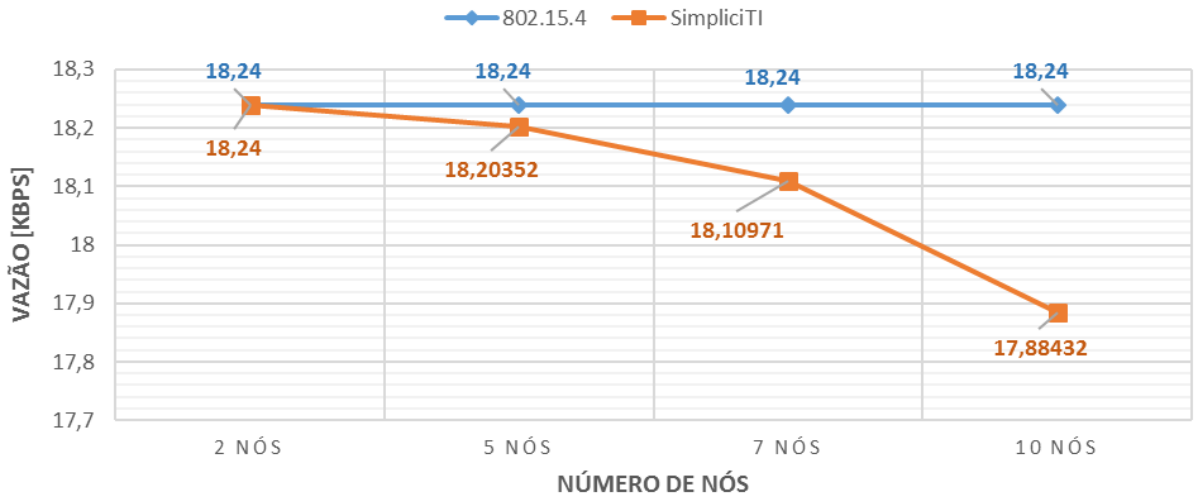

Figura 4.12: Throughput (kbps) para uma taxa de 120 pacotes por segundo.

Capítulo 5

Conclusões

Vivemos um momento onde a tecnologia de comunicação sem fio se encontra em todo lugar. O conceito de Internet das Coisas, que envolve a conectividade, a troca de informação e a interação entre dispositivos sem fio, está cada vez mais difundido; as empresas estão desenvolvendo produtos que cada vez mais apresentam maneiras de se conectar com outros dispositivos, para enriquecer a experiência do usuário. Além disso, diversas áreas passaram a utilizar tecnologia embarcada para melhorar o desempenho de seus processos. Na agricultura, sensores de umidade, temperatura e acidez do solo são espalhados pela lavoura e permitem um controle mais eficaz de processos como colheita, irrigação, e adubagem da terra, por exemplo. Em plantas industriais, onde o controle das variáveis do processo é de extrema importância, o uso de sensores sem fio tem sido cada vez mais recorrente, pois em muitos casos a configuração das plantas dificulta a implementação de uma rede de comunicação via cabos. Para que todos estes processos ocorram sem problemas, é necessário que a comunicação sem fio seja garantida e que não haja perda de informação no processo de comunicação.

Os dois grandes desafios relacionados a RSSF são a autonomia energética dos sensores, assim como a garantia de transmissão dos dados pela rede. Em um nó, o componente com maior gasto energético é o rádio transmissor, quando operando para envio ou recebimento de pacotes. O problema relacionado à transmissão sem fio é a susceptibilidade da mensagem enviada à problemas relacionados à propagação de ondas eletromagnéticas, tais como interferência, atenuação e dispersão. Existem estratégias na literatura que apresentam resultados satisfatórios em reduzir estes efeitos indesejados de RSSF. O controle do *duty cycle* do transceptor, por exemplo, é uma das técnicas mais utilizadas para a redução do consumo de energia do sensor. Já os problemas relacionados à transmissão dos dados possuem diferentes abordagens, de acordo com o efeito a ser contornado. Atenuação e dispersão do sinal estão diretamente ligados à topologia da rede e pelo caminho que o sinal realiza para chegar no seu destino final; desta forma, redes maiores normalmente utilizam topologias em malha, com transmissão de dados *multi-hop*, reduzindo a distância efetiva de transmissão dos dados e garantindo que a mensagem chegue ao destinatário. O problema de interferência e disputa pelo meio de transmissão é abordado pela literatura através de modificações no protocolo de controle de acesso ao meio, modificando a maneira como a rede coordena a transmissão dos dados de cada sensor, de maneira a evitar a colisão ou perda de pacotes.

Neste trabalho foi utilizada uma plataforma de desenvolvimento oferecida pela *Texas Instruments*, que integra microcontrolador, sensor e rádio transmissor, para a implementação e análise de desempenho de uma RSSF. Ao estudar o funcionamento do protocolo de comunicação nativo da Texas, o SimpliciTI, foi constatado que o mecanismo de *backoff* implementado para tentativas e reenvio de dados quando o meio se encontra ocupado não era eficiente, pois apresentava uma janela fixa de intervalos de tempo aleatórios que poderiam ser designados para o nó esperar até tentar reenviar o pacote.

De acordo com os dados obtidos nos testes, a implementação do procedimento de *backoff* do protocolo IEEE 802.15.4 foi realizada com sucesso. Além disso, foi constatado que este procedimento é muito mais robusto que o adotado pelo protocolo SimpliciTI, havendo nenhuma perda de pacote nos testes realizados. Vale notar que a implementação deste procedimento de *backoff*, por ser mais robusta, demanda um esforço computacional maior, tendo como consequência um maior consumo de energia.

Para trabalhos futuros, a comparação do consumo de energia do procedimento implementado com o procedimento original se faz extremamente necessário, visto que o maior consumo de energia e a robustez da rede oferecem um *tradeoff* crítico para a especificidade da aplicação da RSSF, bem como para a vida útil dos seus nós. É viável também a comparação do procedimento implementado com o procedimento de *backoff* de outros protocoloos, como por exemplo o 802.11, de redes *wi-fi*.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] FREY, T. *http://www.futuristspeaker.com/2011/10/tapping-into-the-secret-language-of-plants/ - 02/06/2015.*

[2] COOLEY, D. *http://rtcmagazine.com/articles/view/102871 - 02/06/2015.*

[3] YE, W.; HEIDEMANN, J.; ESTRIN, D. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *Networking, IEEE/ACM Transactions on*, IEEE, v. 12, n. 3, p. 493–506, 2004.

[4] HTTPS://EN.WIKIPEDIA.ORG/WIKI/CARRIER_SENSE_MULTIPLE_ACCESS_WITH_COLLISION_AVOIDANCE. July, 2015.

[5] SILICON LABORATORIES. *http://www.ti.com/.* Julho, 2015.

[6] TEXAS INSTRUMENTS. *Low-pwer RF protocol from Texas Instruments.* [S.l.].

[7] TEXAS INSTRUMENTS. *MSP430x2xx Family User's Guide.* [S.l.], 2004. SLAU144J.

[8] IEEE. *IEEE Standard for Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs).* 2006. 22–23 p.

[9] GÉRALD, S. The internet of things: Between the revolution of the internet and the metamorphosis of objects. In: *Forum American Bar Association.* [S.l.: s.n.], 2010.

[10] LOUREIRO, A. A. et al. Redes de sensores sem fio. In: *Simpósio Brasileiro de Redes de Computadores (SBRC).* [S.l.: s.n.], 2003. p. 179–226.

[11] MARCONDES, A. *Técnicas de taxa de transmissão adaptativa para redes de sensores sem fio.* Tese (Doutorado) — Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Metrologia Científica e Industrial, Florianópolis, 2010.

[12] OLIVEIRA, S. S. de; MOTOYAMA, S. Um protocolo mac para redes de sensores sem fio voltado para aplicações específicas utilizando o método de acesso tdma. *XXVI Simpósio Brasileiro de Telecomunicações-SBrT*, 2008.

[13] DAM, T. V.; LANGENDOEN, K. An adaptive energy-efficient mac protocol for wireless sensor networks. In: ACM. *Proceedings of the 1st international conference on Embedded networked sensor systems.* [S.l.], 2003. p. 171–180.

[14] POLASTRE, J.; HILL, J.; CULLER, D. Versatile low power media access for wireless sensor networks. In: ACM. *Proceedings of the 2nd international conference on Embedded networked sensor systems*. [S.l.], 2004. p. 95–107.

[15] BUETTNER, M. et al. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In: ACM. *Proceedings of the 4th international conference on Embedded networked sensor systems*. [S.l.], 2006. p. 307–320.

[16] EL-HOIYDI, A. et al. Wisemac, an ultra low power mac protocol for the wisenet wireless sensor network. In: ACM. *Proceedings of the 1st international conference on Embedded networked sensor systems*. [S.l.], 2003. p. 302–303.

[17] TEXAS INSTRUMENTS. *MSP Low-Power Microcontrollers*. [S.l.], 2015. Slab034ab.

[18] TEXAS INSTRUMENTS. *MSP430F2274 - Mixed Signal Microcontroller*. [S.l.], 2006. SLAS504G.

[19] TEXAS INSTRUMENTS. *CC2500 - Low-Cost Low-Power 2.4 GHz RF Transceiver*. [S.l.], 2015. SWRS040C.

[20] FRIEDMAN, L. *SimpliciTI: Simple Modular RF Network Developers Notes*. [S.l.], 2007–2009.

[21] MORALES, Z. S. M. *Wireless Sensor Monitor Using the eZ430-RF2500*. [S.l.], 2007–2009.

[22] WESTLUND, L. *Random Number Generation Using the MSP430*. [S.l.], 2006. SLAA338.

ANEXOS

I. TABELAS

Abaixo estão listadas as tabelas criadas a partir dos resultados dos testes realizados como cada procedimento de *backoff*.

I.1 Resultado dos testes do procedimento de *backoff* do protocolo SimpliciTI

Tabela I.1: 2 nós - 2 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2
CR = 1	4	4
CR = 2	0	0
CR = 3	0	0
CR = 4	0	0
Total de CCA Retries	8	

Tabela I.2: 5 nós - 2 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
CR = 1	4	6	4	7	6
CR = 2	0	0	0	0	0
CR = 3	0	0	0	0	0
CR = 4	0	0	0	0	0
Total de CCA Retries	27				

Tabela I.3: 7 nós - 2 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
CR = 1	13	6	12	12	9	20	8
CR = 2	2	0	0	0	0	0	0
CR = 3	0	0	0	0	0	0	0
CR = 4	0	0	0	0	0	0	0
Total de CCA Retries	84						

Tabela I.4: 10 nós - 2 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
CR = 1	17	13	24	12	11	13	17	15	13	9
CR = 2	0	0	0	1	0	0	0	0	0	0
CR = 3	0	0	0	0	0	0	0	0	0	0
CR = 4	0	0	0	0	0	0	0	0	0	0
Total de CCA Retries	146									

Tabela I.5: 2 nós - 5 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2
CR = 1	3	2
CR = 2	0	0
CR = 3	0	0
CR = 4	0	0
Total de CCA Retries	5	

Tabela I.6: 5 nós - 5 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
CR = 1	20	21	15	23	24
CR = 2	0	0	0	0	0
CR = 3	0	0	0	0	0
CR = 4	0	0	0	0	0
Total de CCA Retries	103				

Tabela I.7: 7 nós - 5 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
CR = 1	30	22	20	25	21	31	26
CR = 2	0	0	1	0	1	0	0
CR = 3	0	0	0	0	0	0	0
CR = 4	0	0	0	0	0	0	0
Total de CCA Retries	179						

Tabela I.8: 10 nós - 5 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
CR = 1	38	31	46	36	34	40	37	37	37	31
CR = 2	2	0	0	0	1	2	1	1	0	2
CR = 3	0	0	0	0	0	0	0	0	0	0
CR = 4	0	0	0	0	0	0	0	0	0	0
Total de CCA Retries	385									

Tabela I.9: 2 nós - 10 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2
CR = 1	7	8
CR = 2	0	0
CR = 3	0	0
CR = 4	0	0
Total de CCA Retries	15	

Tabela I.10: 5 nós - 10 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
CR = 1	31	32	37	35	44
CR = 2	1	0	1	1	1
CR = 3	0	0	0	0	0
CR = 4	0	0	0	0	0
Total de CCA Retries	187				

Tabela I.11: 7 nós - 10 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
CR = 1	47	50	52	47	57	52	37
CR = 2	1	2	1	1	2	2	0
CR = 3	0	0	1	0	0	0	0
CR = 4	0	0	0	0	0	0	0
Total de CCA Retries	363						

Tabela I.12: 10 nós - 10 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
CR = 1	80	57	86	80	67	71	82	80	81	66
CR = 2	6	3	4	6	6	5	11	9	4	4
CR = 3	0	0	1	0	0	1	2	0	0	0
CR = 4	0	0	0	0	0	0	0	0	0	0
Total de CCA Retries	878									

Tabela I.13: 2 nós - 40 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2
CR = 1	21	34
CR = 2	0	3
CR = 3	0	0
CR = 4	0	0
Total de CCA Retries	61	

Tabela I.14: 5 nós - 40 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
CR = 1	92	88	103	117	83
CR = 2	12	10	20	15	10
CR = 3	1	4	1	5	0
CR = 4	0	1	0	1	1
Total de CCA Retries	662				

Tabela I.15: 7 nós - 40 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
CR = 1	115	162	120	150	156	155	155
CR = 2	33	27	20	29	37	38	28
CR = 3	3	7	6	3	6	2	6
CR = 4	1	1	0	2	1	1	3
Total de CCA Retries	1572						

Tabela I.16: 10 nós - 40 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
CR = 1	185	200	190	190	207	188	165	204	177	190
CR = 2	57	50	38	55	54	56	55	60	45	46
CR = 3	16	9	12	16	13	14	8	14	9	16
CR = 4	6	6	5	1	8	8	5	8	4	3
Total de CCA Retries	3525									

Tabela I.17: 2 nós - 120 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2
CR = 1	70	29
CR = 2	15	4
CR = 3	0	0
CR = 4	0	0
Total de CCA Retries	137	

Tabela I.18: 5 nós - 120 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
CR = 1	147	180	147	157	161
CR = 2	34	53	47	48	54
CR = 3	10	14	14	13	17
CR = 4	1	8	8	7	6
Total de CCA Retries	1588				

Tabela I.19: 7 nós - 120 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
CR = 1	185	189	225	172	209	241	211
CR = 2	60	77	57	65	79	87	70
CR = 3	21	20	27	26	19	27	15
CR = 4	11	15	15	21	16	15	15
Total de CCA Retries	3319						

Tabela I.20: 10 nós - 120 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
CR = 1	219	246	217	232	208	230	205	219	221	244
CR = 2	90	85	83	94	87	104	100	98	91	111
CR = 3	31	39	50	35	39	38	38	39	43	43
CR = 4	30	43	31	39	35	38	28	23	35	31
Total de CCA Retries	6644									

Tabela I.21: Taxa de envio / número de falhas por nó para 5 nós

	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
40 pacotes/segundo	0	0	0	0	1
Total	1				
120 pacotes/segundo	1	1	2	3	3
Total	10				

Tabela I.22: Taxa de envio / número de falhas por nó para 7 nós

	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
40 pacotes/segundo	0	0	0	1	0	0	0
Total	1						
120 pacotes/segundo	3	5	8	10	9	7	8
Total	50						

Tabela I.23: Taxa de envio / número de falhas por nó para 7 nós

	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
40 pacotes/segundo	2	3	2	1	2	1	0	4	1	0
Total	16									
120 pacotes/segundo	16	24	25	24	15	25	17	15	18	16
Total	195									

I.2 Resultado dos testes do procedimento de *backoff* do protocolo 802.15.4

Tabela I.24: 2 nós - 2 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2
NB = 1	2	1
NB = 2	0	0
NB = 3	0	0
NB = 4	0	0
NB = 5	0	0
Total de backoffs	3	

Tabela I.25: 5 nós - 2 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
NB = 1	3	9	8	5	7
NB = 2	0	0	0	0	0
NB = 3	0	0	0	0	0
NB = 4	0	0	0	0	0
NB = 5	0	0	0	0	0
Total de backoffs	32				

Tabela I.26: 7 nós - 2 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
NB = 1	9	7	11	10	9	13	13
NB = 2	0	0	0	0	0	0	0
NB = 3	0	0	0	0	0	0	0
NB = 4	0	0	0	0	0	0	0
NB = 5	0	0	0	0	0	0	0
Total de backoffs	72						

Tabela I.27: 10 nós - 2 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
NB = 1	17	12	17	12	10	18	9	21	13	23
NB = 2	0	1	2	0	0	0	1	0	0	0
NB = 3	0	0	0	0	0	0	0	0	0	0
NB = 4	0	0	0	0	0	0	0	0	0	0
NB = 5	0	0	0	0	0	0	0	0	0	0
Total de backoffs	362									

Tabela I.28: 2 nós - 5 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2
NB = 1	7	4
NB = 2	0	0
NB = 3	0	0
NB = 4	0	0
NB = 5	0	0
Total de backoffs	11	

Tabela I.29: 5 nós - 5 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
NB = 1	21	22	13	22	8
NB = 2	0	0	0	0	0
NB = 3	0	0	0	0	0
NB = 4	0	0	0	0	0
NB = 5	0	0	0	0	0
Total de backoffs	86				

Tabela I.30: 7 nós - 5 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
NB = 1	33	20	39	27	24	31	9
NB = 2	2	0	2	1	0	1	0
NB = 3	0	0	0	0	0	0	0
NB = 4	0	0	0	0	0	0	0
NB = 5	0	0	0	0	0	0	0
Total de backoffs	195						

Tabela I.31: 10 nós - 5 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
NB = 1	43	48	51	38	41	34	32	48	40	45
NB = 2	4	3	2	5	1	3	5	1	4	2
NB = 3	0	0	0	0	0	0	0	0	0	0
NB = 4	0	0	0	0	0	0	0	0	0	0
NB = 5	0	0	0	0	0	0	0	0	0	0
Total de backoffs	480									

Tabela I.32: 2 nós - 10 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2
NB = 1	7	4
NB = 2	0	0
NB = 3	0	0
NB = 4	0	0
NB = 5	0	0
Total de backoffs	11	

Tabela I.33: 5 nós - 10 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
NB = 1	21	22	13	22	8
NB = 2	0	0	0	0	0
NB = 3	0	0	0	0	0
NB = 4	0	0	0	0	0
NB = 5	0	0	0	0	0
Total de backoffs	86				

Tabela I.34: 7 nós - 10 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
NB = 1	33	20	39	27	24	31	9
NB = 2	2	0	2	1	0	1	0
NB = 3	0	0	0	0	0	0	0
NB = 4	0	0	0	0	0	0	0
NB = 5	0	0	0	0	0	0	0
Total de backoffs	195						

Tabela I.35: 10 nós - 10 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
NB = 1	43	48	51	38	41	34	32	48	40	45
NB = 2	4	3	2	5	1	3	5	1	4	2
NB = 3	0	0	0	0	0	0	0	0	0	0
NB = 4	0	0	0	0	0	0	0	0	0	0
NB = 5	0	0	0	0	0	0	0	0	0	0
Total de backoffs	480									

Tabela I.36: 2 nós - 40 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2
NB = 1	11	16
NB = 2	0	0
NB = 3	0	0
NB = 4	0	0
NB = 5	0	0
Total de backoffs	27	

Tabela I.37: 5 nós - 40 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
NB = 1	56	58	55	58	44
NB = 2	2	3	4	3	3
NB = 3	0	0	0	1	0
NB = 4	0	0	1	0	0
NB = 5	0	0	0	0	0
Total de backoffs	308				

Tabela I.38: 7 nós - 40 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
NB = 1	88	82	68	79	90	90	73
NB = 2	6	5	3	5	4	1	3
NB = 3	1	0	0	0	2	1	0
NB = 4	0	0	0	0	0	1	0
NB = 5	0	0	0	0	0	0	0
Total de backoffs	640						

Tabela I.39: 10 nós - 40 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
NB = 1	127	108	128	107	102	126	115	119	99	111
NB = 2	10	11	13	17	15	13	13	12	16	14
NB = 3	1	0	2	3	0	3	1	4	1	1
NB = 4	0	0	0	0	1	0	0	1	1	1
NB = 5	0	0	0	0	0	0	0	0	0	0
Total de backoffs	1474									

Tabela I.40: 2 nós - 120 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2
NB = 1	24	18
NB = 2	0	0
NB = 3	0	0
NB = 4	0	0
NB = 5	0	0
Total de backoffs	42	

Tabela I.41: 5 nós - 120 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5
NB = 1	80	84	82	64	65
NB = 2	7	10	4	4	7
NB = 3	0	0	0	0	0
NB = 4	0	0	0	0	1
NB = 5	0	0	0	0	0
Total de backoffs	443				

Tabela I.42: 7 nós - 120 pacotes por nó por segundo

Qtde. de backoffs	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7
NB = 1	99	93	99	94	90	88	110
NB = 2	12	16	16	14	20	15	7
NB = 3	0	1	3	2	0	1	2
NB = 4	0	0	0	0	0	1	1
NB = 5	0	0	0	1	1	0	0
Total de backoffs	918						

Tabela I.43: 10 nós - 120 pacotes por nó por segundo

Qtde. de CCA Retries	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6	Nó 7	Nó 8	Nó 9	Nó 10
NB = 1	117	122	111	124	144	136	115	130	97	124
NB = 2	20	28	23	29	25	23	14	11	16	20
NB = 3	6	3	1	1	9	4	1	4	1	5
NB = 4	0	1	0	0	1	2	1	0	0	0
NB = 5	0	0	0	0	0	0	0	0	0	0
Total de CCA Retries	1763									

II. CÓDIGO EM MATLAB PARA AQUISIÇÃO DE DADOS

Foram gerados vários códigos para serem utilizados no *software* MATLAB. Contudo, para fins de simplicidade e respeito ao meio ambiente, será listado somente o código de aquisição de dados pela porta Serial de uma RSSF com dez nós.

```
%% Início do programa

% Conjunto de comandos clear, para limpar variáveis, gráficos e a tela do
% matlab
close all;
clear all;
clc;
fclose('all');
delete(instrfindall);

% ********************************************************************
%Número de informações a serem gravadas, qnts vezes cada sensor deve ser
%lido
% ********************************************************************
No_de_dados = 1200;


% ********************************************************************
% Código utilizado para os testes
% ********************************************************************
% code = '802.15.4';
code = 'SimpliciTI';


% ********************************************************************
%Número de nós testados! (NECESSÁRIO SETAR MANUALMENTE)
% ********************************************************************
nodes = 10;


% ********************************************************************
% Depende da taxa do clock ajustada no código em segundos
% ********************************************************************
```

```
clk_time = 6000;
taxa_pkt = 2; %pkt\seg

% clk_time = 2400;
% taxa_pkt = 5; %pkt\seg

% clk_time = 1200;
% taxa_pkt = 10; %pkt\seg

% clk_time = 300;
% taxa_pkt = 40; %pkt\seg

% clk_time = 100;
% taxa_pkt = 120; %pkt\seg



% ********************************************************************
% Criando objeto da porta serial e abrindo a conexão (ATENTAR PARA A PORTA
SERIAL QUE O AP ESTÁ UTILIZANDO)
% ********************************************************************
sensor = serial('COM6', 'BaudRate', 9600);
fopen(sensor);

% Criando o arquivo txt
nossosDados = fopen('DataLog - Protocolo_.txt','w');
leitura_Sensor01 = fopen('Sensor 01_.txt', 'w');
leitura_Sensor02 = fopen('Sensor 02_.txt', 'w');
leitura_Sensor03 = fopen('Sensor 03_.txt', 'w');
leitura_Sensor04 = fopen('Sensor 04_.txt', 'w');
leitura_Sensor05 = fopen('Sensor 05_.txt', 'w');
leitura_Sensor06 = fopen('Sensor 06_.txt', 'w');
leitura_Sensor07 = fopen('Sensor 07_.txt', 'w');
leitura_Sensor08 = fopen('Sensor 08_.txt', 'w');
leitura_Sensor09 = fopen('Sensor 09_.txt', 'w');
leitura_Sensor10 = fopen('Sensor 10_.txt', 'w');

%Header do arquivo txt
fprintf(nossosDados,'%s\r\n','******* AQUISIÇÃO DE DADOS MSP430 *******');
fprintf(leitura_Sensor01,'******* Nž de backoffs realizados pelo sensor 01 ao
```

```
enviar 1000 pacotes a uma taxa de %i pkts por s (rede %s com %i sensores) em
%s *******\r\n', taxa_pkt, code, nodes, date);
fprintf(leitura_Sensor02,'******* Nž de backoffs realizados pelo sensor 02 ao
enviar 1000 pacotes a uma taxa de %i pkts por s (rede %s com %i sensores) em
%s *******\r\n', taxa_pkt, code, nodes, date);
fprintf(leitura_Sensor03,'******* Nž de backoffs realizados pelo sensor 03 ao
enviar 1000 pacotes a uma taxa de %i pkts por s (rede %s com %i sensores) em
%s *******\r\n', taxa_pkt, code, nodes, date);
fprintf(leitura_Sensor04,'******* Nž de backoffs realizados pelo sensor 04 ao
enviar 1000 pacotes a uma taxa de %i pkts por s (rede %s com %i sensores) em
%s *******\r\n', taxa_pkt, code, nodes, date);
fprintf(leitura_Sensor05,'******* Nž de backoffs realizados pelo sensor 05 ao
enviar 1000 pacotes a uma taxa de %i pkts por s (rede %s com %i sensores) em
%s *******\r\n', taxa_pkt, code, nodes, date);
fprintf(leitura_Sensor06,'******* Nž de backoffs realizados pelo sensor 06 ao
enviar 1000 pacotes a uma taxa de %i pkts por s (rede %s com %i sensores) em
%s *******\r\n', taxa_pkt, code, nodes, date);
fprintf(leitura_Sensor07,'******* Nž de backoffs realizados pelo sensor 07 ao
enviar 1000 pacotes a uma taxa de %i pkts por s (rede %s com %i sensores) em
%s *******\r\n', taxa_pkt, code, nodes, date);
fprintf(leitura_Sensor08,'******* Nž de backoffs realizados pelo sensor 08 ao
enviar 1000 pacotes a uma taxa de %i pkts por s (rede %s com %i sensores) em
%s *******\r\n', taxa_pkt, code, nodes, date);
fprintf(leitura_Sensor09,'******* Nž de backoffs realizados pelo sensor 09 ao
enviar 1000 pacotes a uma taxa de %i pkts por s (rede %s com %i sensores) em
%s *******\r\n', taxa_pkt, code, nodes, date);
fprintf(leitura_Sensor10,'******* Nž de backoffs realizados pelo sensor 10 ao
enviar 1000 pacotes a uma taxa de %i pkts por s (rede %s com %i sensores) em
%s *******\r\n', taxa_pkt, code, nodes, date);

% Variável data recebe o valor atual lido do sensor
data = fgets(sensor);

%Vetores que guardam os valores medidos por cada sensor
dados_no1  = zeros(No_de_dados,1);
dados_no2  = zeros(No_de_dados,1);
dados_no3  = zeros(No_de_dados,1);
dados_no4  = zeros(No_de_dados,1);
dados_no5  = zeros(No_de_dados,1);
dados_no6  = zeros(No_de_dados,1);
dados_no7  = zeros(No_de_dados,1);
dados_no8  = zeros(No_de_dados,1);
```

```
dados_no9  = zeros(No_de_dados,1);
dados_no10 = zeros(No_de_dados,1);


%Inicialização dos índices dos vetores que vão acumular os valores das
%leituras dos sensores
a = 1;
b = 1;
c = 1;
d = 1;
e = 1;
f = 1;
g = 1;
h = 1;
i = 1;
j = 1;

%Inicialização das variáveis que contam o número de falhas em cada nó
fail_no1 = 0;
fail_no2 = 0;
fail_no3 = 0;
fail_no4 = 0;
fail_no5 = 0;
fail_no6 = 0;
fail_no7 = 0;
fail_no8 = 0;
fail_no9 = 0;
fail_no10 = 0;


%Cada iteração k é uma linha obtida da porta serial, sendo os dados do AP e
%os dados dos EDs, por isso 'nodes + 1'; multiplicamos pelo LIMITE_DE_DADOS
%para que todos os nós disponíveis sejam lidos LIMITE_DE_DADOS vezes
for k = 1:(nodes)*No_de_dados

    %disp(data)
    data = fgets(sensor);
    fprintf(nossosDados,'%s\r\n',data);
    header = strcat(data(1), data(2), data(3), data(4));

    %Aloca as leituras de cada sensor nos respectivos vetores
    if strcmp(header, '$000') && strcmp(data(5), '1')
```

```
%         dados_no1(a) = str2num(strcat(data(8),data(9),data(10),data(11)));
          dados_no1(a) = str2num(data(25));
          a=a+1;

          %Contagem de falhas no nó
          if strcmp(data(30), '1')
              fail_no1 = fail_no1 + 1;
          end
      end

      if strcmp(header, '$000') && strcmp(data(5), '2')
%         dados_no2(b) = str2num(strcat(data(8),data(9),data(10),data(11)));
          dados_no2(b) = str2num(data(25));
          b=b+1;

          %Contagem de falhas no nó
          if strcmp(data(30), '1')
              fail_no2 = fail_no2 + 1;
          end
      end

      if strcmp(header, '$000') && strcmp(data(5), '3')
%           dados_no3(c) = str2num(strcat(data(8),data(9),data(10),data(11)));
          dados_no3(c) = str2num(data(25));
          c=c+1;

          %Contagem de falhas no nó
          if strcmp(data(30), '1')
              fail_no3 = fail_no3 + 1;
          end
      end

      if strcmp(header, '$000') && strcmp(data(5), '4')
%           dados_no4(d) = str2num(strcat(data(8),data(9),data(10),data(11)));
          dados_no4(d) = str2num(data(25));
          d=d+1;

          %Contagem de falhas no nó
          if strcmp(data(30), '1')
              fail_no4 = fail_no4 + 1;
          end
      end
```

```
    if strcmp(header, '$000') && strcmp(data(5), '5')
%         dados_no5(e) = str2num(strcat(data(8),data(9),data(10),data(11)));
        dados_no5(e) = str2num(data(25));
        e=e+1;

        %Contagem de falhas no nó
         if strcmp(data(30), '1')
             fail_no5 = fail_no5 + 1;
         end
    end

    if strcmp(header, '$000') && strcmp(data(5), '6')
%         dados_no6(f) = str2num(strcat(data(8),data(9),data(10),data(11)));
        dados_no6(f) = str2num(data(25));
        f=f+1;

        %Contagem de falhas no nó
         if strcmp(data(30), '1')
             fail_no6 = fail_no6 + 1;
         end
    end

    if strcmp(header, '$000') && strcmp(data(5), '7')
%         dados_no7(g) = str2num(strcat(data(8),data(9),data(10),data(11)));
        dados_no7(g) = str2num(data(25));
        g=g+1;

        %Contagem de falhas no nó
         if strcmp(data(30), '1')
             fail_no7 = fail_no7 + 1;
         end
    end

    if strcmp(header, '$000') && strcmp(data(5), '8')
%         dados_no8(h) = str2num(strcat(data(8),data(9),data(10),data(11)));
        dados_no8(h) = str2num(data(25));
        h=h+1;

        %Contagem de falhas no nó
         if strcmp(data(30), '1')
             fail_no8 = fail_no8 + 1;
```

```
        end
    end

    if strcmp(header, '$000') && strcmp(data(5), '9')
%         dados_no9(i) = str2num(strcat(data(8),data(9),data(10),data(11)));
        dados_no9(i) = str2num(data(25));
        i=i+1;

        %Contagem de falhas no nó
         if strcmp(data(30), '1')
             fail_no9 = fail_no9 + 1;
         end
    end

    if strcmp(header, '$001') && strcmp(data(5), '0')
%         dados_no10(j) = str2num(strcat(data(8),data(9),data(10),data(11)));
        dados_no10(j) = str2num(data(25));
        j=j+1;

        %Contagem de falhas no nó
         if strcmp(data(30), '1')
             fail_no10 = fail_no10 + 1;
         end
    end

end


% Salva os dados lidos em arquivos .txt
 fprintf(leitura_Sensor01,'%i\r\n',dados_no1);
 fprintf(leitura_Sensor02,'%i\r\n',dados_no2);
 fprintf(leitura_Sensor03,'%i\r\n',dados_no3);
 fprintf(leitura_Sensor04,'%i\r\n',dados_no4);
 fprintf(leitura_Sensor05,'%i\r\n',dados_no5);
 fprintf(leitura_Sensor06,'%i\r\n',dados_no6);
 fprintf(leitura_Sensor07,'%i\r\n',dados_no7);
 fprintf(leitura_Sensor08,'%i\r\n',dados_no8);
 fprintf(leitura_Sensor09,'%i\r\n',dados_no9);
 fprintf(leitura_Sensor10,'%i\r\n',dados_no10);

% Salva as variáveis do workspace
save('bkp_workspace');
```

```
% Fecha o arquivo txt e a conexão com a porta serial
close(nossosDados);
close(leitura_Sensor01);
close(leitura_Sensor02);
close(leitura_Sensor03);
close(leitura_Sensor04);
close(leitura_Sensor05);
close(leitura_Sensor06);
close(leitura_Sensor07);
close(leitura_Sensor08);
close(leitura_Sensor09);
close(leitura_Sensor10);
fclose(data);
```

III. DESCRIÇÃO DO CONTEÚDO DO CD

Estão contidos no CD todos os arquivos utilizados neste trabalho, tanto os códigos fonte utilizados na programação dos nós e do ponto de acesso quanto os códigos fonte da aquisição de dados no ambiente MATLAB. Estão presentes também os arquivos de *datalog* dos testes realizados. Além disso, estão presentes os slides da apresentação deste trabalho.