UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE ENERGIA

MARCOS ANTÔNIO DE SORDI

OTIMIZAÇÃO DA EFICIÊNCIA ENERGÉTICA EM REDES OPERANDO COM TSCH: AVALIAÇÃO ANALÍTICA DE UMA IMPLEMENTAÇÃO PRÁTICA

DISSERTAÇÃO

CURITIBA
2019

MARCOS ANTÔNIO DE SORDI

OTIMIZAÇÃO DA EFICIÊNCIA ENERGÉTICA EM REDES OPERANDO COM TSCH: AVALIAÇÃO ANALÍTICA DE UMA IMPLEMENTAÇÃO PRÁTICA

Dissertação apresentada ao Programa de Pós-graduação em Sistemas de Energia da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Mestre em Engenharia Elétrica" – Área de Concentração: Automação e Sistemas de Energia.

Orientador: Prof. Dr. Ohara Kerusauskas Rayel

Coorientador: Prof. Dr. Guilherme Luiz Moritz

CURITIBA
2019

Dados Internacionais de Catalogação na Publicação

Sordi, Marcos Antônio de
Otimização da eficiência energética em redes operando com TSCH [recurso eletrônico] : avaliação analítica de uma implementação prática / Marcos Antônio de Sordi.-- 2019.
1 arquivo eletrônico (64 f.) : PDF ; 2,43 MB.

Modo de acesso: World Wide Web.
Texto em português com resumo em inglês.
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Sistemas de Energia. Área de Concentração: Automação e Sistemas de Energia, Curitiba, 2019.
Bibliografia: f. 60-63.

1. Sistemas de energia elétrica - Dissertações. 2. Energia - Consumo. 3. Sincronização. 4. Sistemas de comunicação sem fio. 5. Redes de sensores sem fio. 6. Nós sensores sem fio. 7. Controle de custo. 8. Internet das coisas. 9. Redes de computação - Protocolos. 10. Métodos de simulação. I. Rayel, Ohara Kerusauskas, orient. II. Moritz, Guilherme Luiz, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Sistemas de Energia. IV. Título.

CDD: Ed. 23 -- 621.31

Biblioteca Central do Câmpus Curitiba - UTFPR
Bibliotecária: Luiza Aquemi Matsumoto CRB-9/794

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Pesquisa e Pós-Graduação

TERMO DE APROVAÇÃO DE DISSERTAÇÃO

A Dissertação de Mestrado intitulada "**Otimização da Eficiência Energética em Redes operando com TSCH: avaliação analítica de uma implementação prática**", defendida em sessão pública pelo candidato **Marcos Antônio de Sordi**, no dia 28 de agosto de 2019, foi julgada para a obtenção do título de Mestre em Engenharia Elétrica, área de concentração Automação e Sistemas de Energia, e aprovada em sua forma final, pelo Programa de Pós-Graduação em Sistemas de Energia.

BANCA EXAMINADORA:

Prof. Dr. Ohara Kerusauskas Rayel – Presidente – UTFPR
Prof. Dr. Guilherme de Santi Peron – UTFPR
Prof. Dr. Marcelo Eduardo Pellenz – PUC-PR/UFPR

A via original deste documento encontra-se arquivada na Secretaria do Programa, contendo a assinatura da Coordenação após a entrega da versão corrigida do trabalho.

Curitiba, 28 de agosto de 2019.

Carimbo e Assinatura do(a) Coordenador(a) do Programa

AGRADECIMENTOS

É certo que tudo aquilo que desejamos e pelo qual lutamos arduamente em nossas vidas, não seria possível conquistar sem ajuda. Para a realização deste trabalho não foi diferente. Meus sinceros agradecimentos a todos que de alguma forma, mesmo que distante, contribuíram para sua conclusão, em especial:

à toda minha família, pela compreensão nas incontáveis ausências dominicais;

a Ohara Kerusauskas Rayel, meu orientador, pela dedicação, paciência e determinação. Pelas inúmeras horas de ajuda na pesquisa, tarde à dentro no LabSC, por demonstrar e delimitar o caminho a ser seguido neste trabalho;

a Guilherme Luiz Moritz, meu coorientador, pelo apoio em diversas etapas da pesquisa para que conseguissemos atingir o objetivo final;

aos Professores Marcelo Eduardo Pellenz e Guilherme de Santi Peron pela gentiliza em aceitar o convite para participar da banca examinadora;

ao Programa de Pós-Graduação em Sistemas de Energia, por ter me aceito para participar deste projeto de pesquisa;

ao Departamento de Infra Estrutura do Instituto Federal do Paraná, pela compreensão e liberação para a realização deste trabalho.

Se não podes entender, crê para que entendas.
A fé precede, o intelecto segue.
(Santo Agostinho)

RESUMO

SORDI, M. A. Otimização da Eficiência Energética em Redes operando com TSCH: Avaliação Analítica de uma Implementação Prática. 63 f. Dissertação – Programa de Pós-graduação em Sistemas de Energia, Universidade Tecnológica Federal do Paraná. Curitiba, 2019.

O padrão IEEE 802.15.4-2015 define um número de protocolos da camada de controle de acesso ao meio para comunicações sem fio de baixa potência, o que é desejável em dispositivos com restrições em IoT. Originalmente definido na norma IEEE 802.15.4e, a multiplexação por divisão de tempo e salto em frequência (TSCH) tem atraído a atenção nas recentes pesquisas da comunidade científica, devido à sua contenção reduzida (agendamento de tempo) e robustez (salto de canal). Contudo, o TSCH necessita de um certo nível de sincronização entre seus nós, o que pode levar a um consumo maior de energia. Um mecanismo para o tempo de guarda é implementado para garantir que os nós irão ouvir os quadros mesmo que eles não estejam perfeitamente sincronizados. Neste trabalho, implementa-se a estratégia do Guard Beacon, com o objetivo de reduzir o tempo de guarda além de apresentar um modelo realista de consumo de energia para redes TSCH baseadas no sistema operacional Contiki. Os valores analíticos possuem boa correspondência com os resultados obtidos a partir da ferramenta Powertrace do Contiki simulando uma rede TSCH real e demonstram que o esquema proposto pode reduzir o consumo total de energia em cada nó em 13,5%.

Palavras-chave: Eficiência Energética, Sincronização, Tempo de Guarda

ABSTRACT

SORDI, M. A. TSCH Energy Efficiency Optimization: An Analytical Aproach for a Pratical Implementation. 63 f. Dissertação – Programa de Pós-graduação em Sistemas de Energia, Universidade Tecnológica Federal do Paraná. Curitiba, 2019.

IEEE 802.15.4-2015 standard defines a number of Medium Access Control (MAC) layer protocols for low power wireless communications, which is desirable for constrained Internet of Things (IoT) devices. Originally defined in IEEE 802.15.4e amendment, the Time Slotted Channel Hopping (TSCH) is recently attracting the attention from the research community, due to its reduced contention (time scheduling) and robustness (channel hopping). However, the TSCH needs a certain level of synchronization between the nodes, which can lead to a higher energy consumption. A guard time mechanism is implemented to ensure that the nodes will hear the frames even if they are not perfectly synchronized. In this work, we implement the Guard Beacon strategy, aiming to reduce the guard time and present a realistic energy consumption model for a Contiki OS-based TSCH networks. The analytical values have a good match with the results obtained from the Contiki Powertrace Tool running on a real TSCH network and demonstrate that the proposed scheme can reduce the overall power consumption of each node by 13.5%.

Keywords: Energy Efficiency, Synchronization, Guard Time

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

6LoWPAN	Redes de Baixa Potência em Áreas Pessoais sobre IPv6, do inglês IPv6 over Low Power Wireless Personal Area Networks
AGT	Tempo de Guarda do ACK, do inglês *Ack Guard Time*
API	Interface de Programação de Aplicações, do inglês *Application Programming Interface*
CoAP	Protocolo de Aplicação para Redes de Baixa Potência com Restrições, do inglês *Constrained Application Protocol*
CSMA-CA	Múltiplo Acesso com Verificação de Portadora e Prevenção de Colisão, do inglês *Carrier Sense Multiple Access - Colision Avoidance*
DSME	Extensão Multicanal Determinístico e Síncrono, do inglês *Deterministic and Synchronous Multi-channel Extension*
EBs	Faróis de RF Aprimorados, do inglês *Enhaced Beacons*
IoT	Internet das Coisas, do inglês *Internet of Things*
IP	Protocolo de Internet, do inglês *Internet Protocol*
LLDN	Rede Determinística de Baixa Latência, do inglês *Low Latency Deterministic Network*
Lpm	Modo de Baixo Consumo, do inglês *Low Power Mode*
MAC	Controle de Acesso ao Meio, do inglês *Medium Access Control*
MQTT	Transporte em Fila de Mensagens de Telemetria, do inglês *Message Queue Telemetry Transport*
OS	Sistema Operacional, do inglês *Operating System*
PGT	Tempo de Guarda do Pacote, do inglês *Packet Guard Time*
RDC	Ciclo de Trabalho do Rádio, do inglês *Radio Duty Cycle*
RPL	Protocolo de roteamento para redes com perdas e baixo consumo de energia, do inglês *Routing Protocol for Low-Power and Lossy Networks*
TDM	Multiplexação por Divisão no Tempo, do inglês *Time Division Multiplexing*
TSCH	Multiplexação por Divisão de Tempo e Salto em Frequência, do inglês *Time Slotted Channel Hopping*
WSNs	Redes de Sensores Sem Fio, do inglês *Wireless Sensor Networks*

LISTA DE SÍMBOLOS

T_{mode}	Tempo despendido em cada modo
V	Tensão de entrada
I_{mode}	Corrente consumida em cada modo
T_{sim}	Tempo total de simulação
$\overline{P}_{type}$	Variação da potência consumida
$E_{mode,type}$	Energia consumida para um dado modo para um determinado *timeslot*
T_{slot}	Tempo total de um *timeslot*
$T_{state,mode}$	Tempo gasto para cada modo para um dado estado
Q	Número total de *timeslots*
Δ	Função que retorna o tipo da 2-tupla $(t, type)$

SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais, a Internet além de conectar pessoas ao redor do mundo, está migrando para conectar coisas e objetos. Chamada de Internet das Coisas (IoT, do inglês *Internet of Things*), pode ser definida como um conceito que descreve a ideia de objetos físicos cotidianos serem conectados à Internet e serem capazes de se comunicar com outros dispositivos (HEJAZI et al., 2018).

A infra estrutura necessária para suportar uma rede IoT e possibilitar tal comunicação entre os objetos começa com o emprego de redes de sensores sem fio (WSNs do inglês, *Wireless Sensor Network*), que por meio de nós sensores monitoram o ambiente ao seu redor, e encaminham quaisquer informações e comandos para outros nós sensores ou centrais de monitoramento (AZOIDOU et al., 2018).

Os nós sensores utilizados nessas redes são alimentados principalmente por baterias como fontes independentes de energia. A bateria entrega um fornecimento estável de energia, mas possui um incoveniente, sua capacidade e vida útil são limitadas (KIM et al., 2018). Para atenuar este problema, é necessário minimizar o consumo de energia dos nós sensores, com o objetivo de aumentar sua vida útil e reduzir custos com a substituição prematura. O que em muitos casos, pode se tornar inviável devido às restrições de recursos naturais e econômicos ou ainda, pela natureza hostil dos ambientes onde estas redes possam estar instaladas (YOUSSEF et al., 2016).

Diante deste problema, a eficiência energética se torna um dos maiores desafios na implementação das WSNs. Quanto maior for a rede, maior será a demanda por energia proveniente das baterias. Estimativas de mercado apontam que o número de conexões de IoT feitas na rede celular deve alcançar 3,5 bilhões em 2023 com uma taxa de crescimento anual esperada de 30% (ERICSSON, 2018).

1.1 INTRODUÇÃO AO PROBLEMA

Tipicamente, um nó sensor é um dispositivo que inclui três componentes básicos: um sensor para aquisição de dados a partir do ambiente físico à sua volta, uma CPU para o processamento e armazenamento das informações coletadas, e um sistema de comunicação sem fio para transmissão e recepção de dados, além da fonte de energia, para que o nó sensor execute as tarefas programadas. No entanto, a maior quantidade de energia consumida pelo nó sensor é causada pela CPU e pelo rádio, mesmo quando estes

permanecem em estado inativo (ANASTASI et al., 2009)

Muitos trabalhos na literatura têm abordado a eficiência energética em redes de sensores sem fio com restrição de energia através de redução do ciclo de trabalho do rádio (RDC, do inglês *Radio Duty Cycle*). O RDC agenda os períodos de transmissão e recepção de dados, o período inativo e o período em que o rádio deve ser desligado, em intervalos regulares de tempo. O RDC é parte integrante do protocolo de controle de acesso ao meio (MAC, do inglês *Medium Access Control*). A camada MAC possui um fator significativo na redução do consumo de energia nas redes de sensores sem fio, desta maneira deve estar configurada para atingir a maior eficiência energética possível (OJO et al., 2016).

Protocolos MAC baseados em múltiplo acesso com verificação de portadora e prevenção de colisão (CSMA-CA, do inglês *Carrier Sense Multiple Access - Colision Avoidance*) podem não ser as melhores soluções para WSNs com restrição de energia, pois seu princípio de funcionamento baseado em contenção de pacotes, pode causar um atraso maior na transmissão e recepção dos dados (LI et al., 2015). A consequência deste comportamento é que a bateria do nó sensor pode ter sua vida útil dramaticamente reduzida, se um mecanismo de RDC não estiver associado ao CSMA (DUQUENNOY et al., 2017).

Entretanto, um protocolo MAC está atraindo a atenção da comunidade científica devido a contenção reduzida, pois realiza o agendamento no tempo para o envio de pacotes, e robustez, realizando o salto de canal e empregando diversidade de frequência. Trata-se da multiplexação por divisão de tempo e salto em frequência (TSCH, do inglês *Time Slotted Channel Hopping*).

O aspecto de divisão no tempo do TSCH é uma técnica de multiplexação por divisão no tempo (TDM, do inglês *Time Division Multiplexing*) que requer que todos o nós da rede estejam sincronizados. A divisão no tempo é realizada em intervalos (*timeslots*), que são longos o suficiente para enviar um pacote de dados de tamanho máximo (127 Bytes). Todos os nós da rede seguem um agendamento que indica um intervalo de tempo ativo e uma frequência disponível para transmissão ou recepção. A frequência é calculada a partir de uma função específica dentro de 16 canais disponíveis. A cada intervalo de tempo disponível é calculada uma nova frequência, ocasionando o salto de canais do TSCH (THUBERT et al., 2013).

1.2 MOTIVAÇÃO

O crescimento global do uso de dispositivos móveis para redes de comunicações sem fio tem gerado uma demanda crescente por baterias para este tipo de dispositivo. A eficiência energética contribui para sustentabilidade à medida que o uso racional dos recursos energéticos acaba retardando a substituição e o descarte precoce destas baterias.

Em WSNs esta também é uma questão muito relevante. Assim, é desejável o estudo de protocolos e a elaboração de técncias que contribuam para redução do consumo energético.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Propor um modelo analítico do consumo de energia dos nós sensores e aplicar técnicas disponíveis na literatura a fim de maximizar a eficiência energética.

1.3.2 Objetivos Específicos

- Elaborar um modelo analítico que possibilite prever o consumo energético dos nós sensores sem a necessidade de simulação;
- Investigar ferramentas para avaliação e melhoria no sincronismo da rede no intuito de reduzir o consumo de energia no TSCH;
- Confirmar as impressões sobre o modelo analítico de consumo com os resultados de modelos de simulação;
- Apresentar graficamente os resultados obtidos comparando o comportamento padrão da rede com as otimizações aplicadas ao protocolo.

1.4 ESTRUTURA DO DOCUMENTO

O Capítulo 2 apresenta as principais características do TSCH, suas vantagens e desvantagens.

O Capítulo 3 contempla a modelagem utilizada e as equações que definem o consumo de energia pelo nó sensor em cada tipo de timeslot existente. O tipo de rede, as plataformas testadas e os aprimoramentos feitos no processo de sincronização também são apresentados.

O Capítulo 4 contém as avaliações sobre o modelo de equações analítico. A precisão do modelo proposto, a eficiencia energética e a introdução de *beacons* específicos para sincronização são comparados com a configuração original através de resultados experimentais e de simulação. Também são apresentados os resultados gráficos que comprovam o intervalo de sincronização ótimo que maximiza a eficiência energética do protocolo.

Por fim, o Capítulo 5 apresenta as conclusões e comentários finais desta dissertação.

2 PRELIMINARES

Neste capítulo alguns conceitos sobre o acesso ao canal sem fio serão apresentados. A operação do esquema TSCH é apresentada, incluindo o mecanismo de sincronização. A associação entre precisão da sincronização e o consumo de energia é realizada e serve de base para o desenvolvimento apresentado no Capítulo 3.

2.1 O PADRÃO IEEE 802.15.4E

O padrão IEEE 802.15.4 publicado inicialmente em 2003, definiu os modos de operação da camada física e da camada de acesso ao meio para redes de sensores de baixa potência em dois modos: com farol de rf (*Beacons*) habilitados e não habilitados (802.15.4, 2006). Em 2012, a revisão 802.15.4e introduziu novos modelos de camada MAC (802.15.4E, 2012):

- Extensão Multicanal Determinístico e Síncrono (DSME, do inglês, *Deterministic and Synchronous Multi-channel Extension*): Modelo de camada MAC apropriado para utilização em redes do tipo malha com requisitos de escalabilidade e latência determinística. Suas principais características são a sincronização no tempo, o envio de *Beacons* em massa, o salto de canal e adaptação do canal (JEONG; LEE, 2012);
- TSCH: Modelo de camada MAC utilizado em redes que executam múltiplos saltos, requerem altas taxas de tranferência, latência limitada e alta confiabilidade. Suas principais características são a sincronização no tempo, o envio de *Beacons* sob demanda e o salto de canal (JEONG; LEE, 2012);
- Rede Determinística de Baixa Latência (LLDN, do inglês, *Low Latency Deterministic Network*): Modelo de camada MAC utilizado em redes do tipo estrela que exigem alta confiabilidade e baixa latência. Suas principais características são a sincronização no tempo e o uso de apenas dois *timeslots* para gerenciamento da rede (KURUNATHAN et al., 2017);

Dentre os modelos de camada MAC apresentadas pelo padrão IEEE 802.15.4E, o TSCH é o que mais atrai a atenção, mostrando-se promissor com relação ao consumo energético das WSNs (OJO et al., 2016). As características de não estar limitado a uma topologia de rede específica, operar com envio agendado de *Beacons*, a possibilidade de

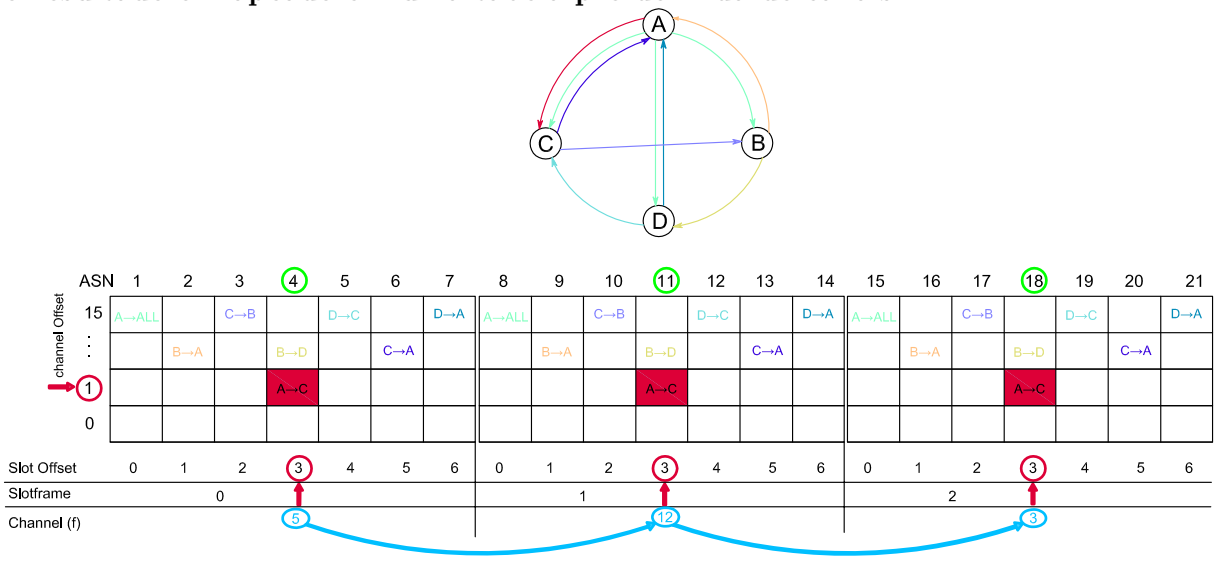
trabalhar com latência determinística e de executar múltiplos saltos de canais empregando sincronização no tempo, tornaram o TSCH objeto de estudo deste tabalho.

Em uma rede TSCH todos os nós são sincronizados. O tempo é dividido em *timeslots* com duração usual de 10 ou 15 ms. Estes *timeslots* são agrupados em intervalos maiores de tempo, formando um quadro (*slotframe*), que se repete continuamente ao longo do tempo (STANISLOWSKI et al., 2014).

O agendamento da comunicação na rede instrui cada nó sobre o que fazer em cada *timeslots*: transmitir, receber, entrar no modo inativo ou dormir. Para que a comunicação ocorra, um *link* é atribuído para a comunicação direcionada entre os dispositivos, contendo um determinado *timeslot* e um determinado identificador de canal. Uma das principais características do TSCH é o esquema de salto de canais. Para a banda de frequências ISM, até dezesseis canais estão disponíveis para comunicação e cada canal recebe um número inteiro de 0 (zero) a 15 (quinze) como identificador (GUGLIELMO et al., 2017).

Um exemplo de rede TSCH é apresentado na Figura 1.

Figura 1 – Exemplo de Operação de uma Rede TSCH. O primeiro *timeslot* é dedicado a pacotes *broadcast*: A→All, enquanto os outros *timeslots* são para comunicação dedicada: B→A, C→B, etc. O *Slotframe* é composto por 7 *timeslots*, e o ciclo se repete por 3 vezes. A frequência utilizada depende da equação (1), onde o resultado é mapeado em uma tabela pré-definida de canais.



Fonte: Autoria própria

A frequência f que será designada para o estabelecimento da conexão é dada por

$$f = F_{\text{mapp}}\left[(ASN + ch_{\text{offset}}) \mod N_{\text{c}}\right], \tag{1}$$

onde ch_{offset} é um parâmetro que permite que diferentes canais sejam usados no mesmo intervalo de tempo para diferentes quadros de slot, o ASN é o Número de Slot Absoluto, que corresponde ao número total de intervalos de tempo que ocorreram desde que a rede foi implementada, $x \mod y$ é o operador que representa o resto da divisão de x por y, N_{c} é o número total dos canais disponíveis e F_{mapp} é uma tabela de consulta que contém uma sequência predefinida de canais.

2.2 AGENDAMENTO DOS TIMESLOTS NO TSCH

Os mecanismos e as regras para configuração dos recursos de comunicação, roteamento e políticas de segurança não são definidos pelo padrão IEEE 802.15.4e (GUGLIELMO et al., 2016b). Contudo, algumas soluções para o agendamento da comunicação e modelo de tráfego são encontradas na literatura. Elas podem ser classificadas em centralizadas e distribuídas. No esquema centralizado, um nó específico da rede, normalmente o nó coordenador, cria, distribui e atualiza o agendamento da comunicação baseado nas informações recebidas dos outros nós da rede, tais como topologia da rede e tráfego gerado. Entretanto o *link* de comunicação entre os nós precisa ser recalculado e redistribuído a cada mudança de operação na rede. Este esquema é utilizado em redes estáticas onde não há mobilidade dos nós. No esquema distribuído, o *link* de comunicação é calculado de forma autônoma por cada nó, baseado em sua localização e informações trocadas com os seus vizinhos. O esquema distribuído é normalmente utilizado em redes onde existe mobilidade dos nós (GUGLIELMO et al., 2016a).

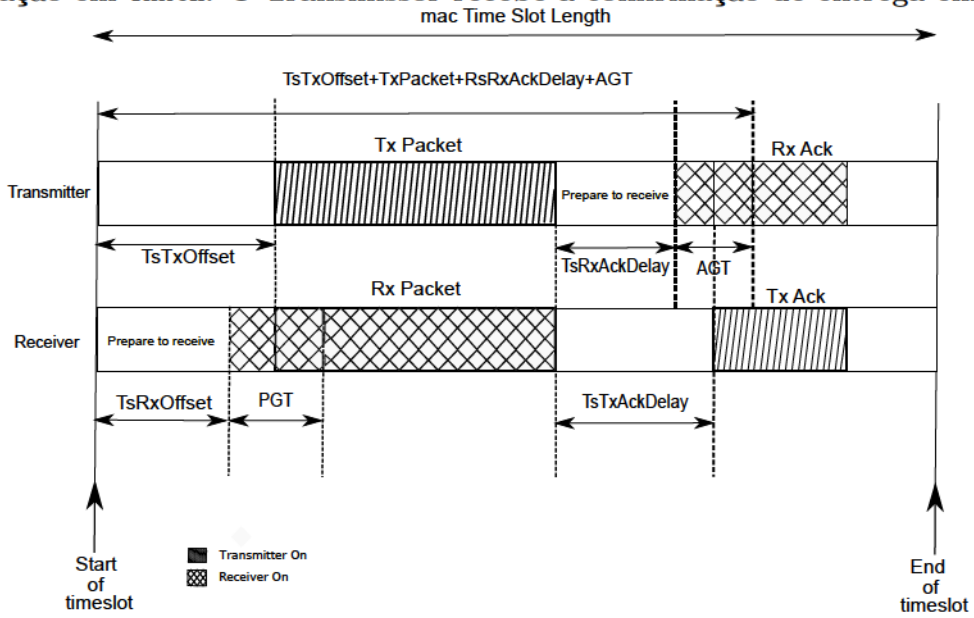
Quando um nó sensor se associa a uma rede TSCH, ele obtém informações sobre a duração de cada *timeslot* e o número de *timeslots* que compõem o *slotframe*. No padrão IEEE 802.15.4e, há seis tipos diferentes de *timeslots* (VILAJOSANA et al., 2014a):

- `TxDataRxAck`: transmissão de dados de um usuário para outro, com confirmação de recebimento (ACK), ou transmissão *unicast*;
- `TxData`: transmissão de dados de um usuário para todos os demais, sem confirmação de recebimento (ACK), ou transmissão *broadcast*;
- `RxDataTxAck`: recepção de dados com confirmação de recebimento (ACK), ou recepção *unicast*;

- `RxData`: recepção de dados sem confirmação de recebimento (ACK), ou recepção *broadcast*;
- `Idle`: intervalo de tempo no qual o nó escuta o canal que está aguardando algum dado, mas não recebe pacote algum;
- `Sleep`: intervalo de tempo em que o rádio permanece desligado.

A Figura 2 ilustra o modelo de *timeslot* definido pelo IEEE 802.15.4e. apresentando a sequência de estados em um *timeslot* do tipo `TxDataRxAck` e em um *timeslot* do tipo `RxDataTxAck`, respectivamente. Os demais tipos de *timeslots* são derivados do mesmo modelo, com a diferença que não possuem o ACK.

Figura 2 – Modelo de *timeslot* para Tx e Rx definido pelo IEEE 802.15.4e: O pacote é enviado `TsTxOffset` e o rádio é desligado. No lado Receptor, o rádio é ligado em `TsRxOffset`, aguarda pelo tempo de guarda (`PGT`) para recepção do pacote e envia confirmação em `TxAck`. O Transmissor recebe a confirmação de entrega em `RxAck`.



Fonte: Autoria própria.

O nó transmissor inicia aguardando por `TsTxOffset`, tempo durante o qual ele prepara os dados para serem enviados e ajusta o rádio de acordo com a frequência calculada, através de (1). O rádio então é ligado e transmite o pacote exatamente `TsTxOffset` depois do início do *timeslot*. Depois que o último byte deixa o rádio, o transmissor aguarda por `TsRxAckDelay` para que o nó receptor se prepare e envie um ACK. Se o ACK é recebido dentro do tempo de guarda do ACK (AGT, do inglês *Ack Guard Time*) a transmisssão é considerada bem sucedida, caso contrário se, nenhum pacote é recebido, o dispositivo desliga o rádio e a transmissão é considerada mal sucedida.

No lado do nó receptor, o dispositivo aguarda por um período de tempo denominado `TsRxOffset` e, liga seu rádio e aguarda a chegada de um pacote. Se após o período de tempo de guarda do pacote (PGT, do inglês *Packet Guard Time*) nenhum pacote é recebido, o dispositivo desliga seu rádio pelo restante do *timeslot*. Se um pacote válido é recebido, o dispositivo aguarda o tempo de `TsTxAckDelay` após a recepção do último Byte, antes de ligar o rádio novamente para enviar o ACK.

2.3 A REDE TSCH

2.3.1 Formação da Rede

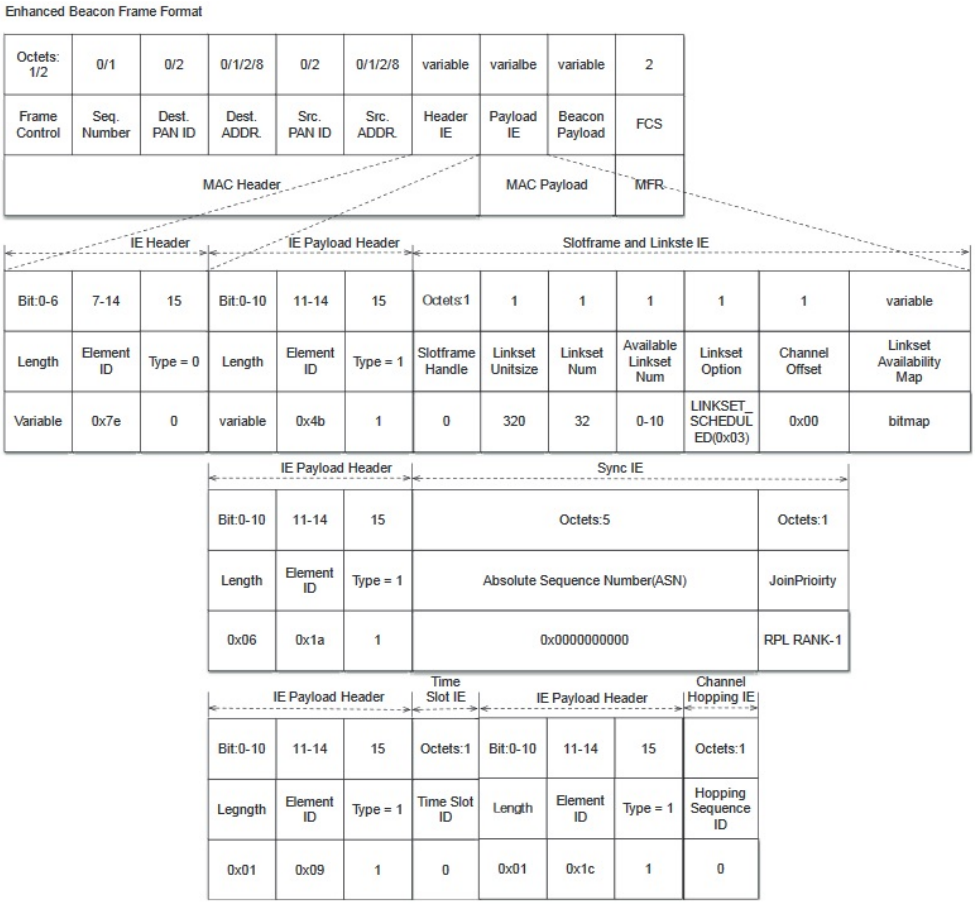
O processo de formação da rede no TSCH começa quando o nó coordenador anuncia sua presença na rede pela transmissão de Faróis de RF Aprimorados (EBs, do inglês *Enhaced Beacons*). EBs são pacotes especiais do TSCH que contém todas as informações necessárias para um nó se associar a rede e iniciar a comunicação com os demais nós (802.15.4E, 2012). Um EB válido contém possui os seguintes elementos de informações:

- Informação de sincronização: permite novos dispositivos sincronizarem seus *clocks* com o dos nós que já estão na rede;
- Informação de salto de canal: permite a novos dispositivos aprenderem a sequência de canais a serem utilizados para comunicação;
- Informação de *timeslot*: usado para especificar a temporização interna do timeslot;
- Informação inicial de *link* e de *slotframe*: permite aos nós que estão se juntando à rede a informação sobre o agendamento inicial a ser seguido.

A Figura 3 mostra o formato de um quadro EB com os respectivos elementos de informações.

Quando um nó deseja ingressar em uma rede TSCH, ele liga seu rádio e começa a aguardar possíveis EBs. Tão logo ele receba um EB válido, são configurados o *slotframe* e o *link* de acordo com as informações recebidas no EB e o dispositivo está apto a se comunicar na rede TSCH (DUY; KIM, 2015). A partir disto, o dispositivo que enviou o EB passa a alocar recursos de comunicação (*link* e *slotframe*) para o nó que está ingressando, bem como incluir uma rotina para autenticar o dispositivo, além de estabelecer chaves de segurança e configurar informações de rota (GUGLIELMO et al., 2014).

Figura 3 – Formato do quadro utilizado para um EB de acordo com o padrão IEEE 802.1.4e. Os elementos de informação que permitem o nó se associar à rede TSHC, sincronizar seu clock e aprender a sequência de salto de canais são mostradas no campo *IE Payload header*.



Fonte: (KIM et al., 2017).

Quando o dispositivo que está ingressando na rede finaliza a sua configuração inicial, ele pode então transmitir EBs, anunciando por sua vez a presença na rede. O envio dos EBs pode ser configurado em intervalos pré determinados de tempo para manter o processo de sincronização entre os dispositivos.

2.3.2 Sincronização dos Nós

Todos os nós na rede TSCH precisam estar sincronizados. Se o transmissor e o receptor estão perfeitamente sincronizados, o rádio possui a informação exata sobre o momento ideal para acionar seu receptor (VILAJOSANA et al., 2014b). No caso ideal, tanto transmissor quanto receptor seriam ligados apenas com duração de tempo equivalente ao tamanho do pacote a ser transmitido. Depois da recepção do pacote, ambos os nós podem desligar seu rádio por alguns poucos milissegundos antes de repetirem o

mesmo processo para envio ou recepção de um ACK.

Este mecanismo pode representar uma solução em termos de consumo de energia, desde que seja minimizado o tempo em que os rádios ficam ligados. Em uma rede TSCH todos os nós transmissores iniciam o envio do pacote em um intervalo de tempo dentro do *timeslot*, instante este chamado de `TsTxOffset` (STANISLOWSKI et al., 2014). Numa rede real, onde não há sincronismo perfeito, os receptores são ligados antes deste instante e continuam ativos por um intervalo de tempo depois deste instante, mecanismo denominado tempo de guarda.

Para manter a sincronização dentro de uma faixa condizente com o tempo de guarda, um pacote periódico de sincronização de relógio é enviado com intervalo de 60 segundos entre os pacotes, de acordo com a configuração padrão. Este é um dos principais métodos para manter a sincronização entre os nós.

O pacote é enviado, no exato momento da transmissão em `TsTxOffset`. O nó receptor por sua vez, sincroniza seu *clock* e calcula o desvio (*drift*) entre o exato momento em que o pacote é recebido e, o momento para qual ele era esperado no início do tempo de guarda, conforme mostrado na Figura 7 da Seção 3.3. A precisão deste processo é determinada pela frequência do cristal. Um cristal oscilando em 32.768 Hz, tipíco em uma WSN, possui uma precisão de aproximadamente 30 μs (CHANG; WANG, 2014).

2.3.3 Consumo de Energia no Processo de Sincronização

Para manter a precisão da sincronização, a rede necessita enviar pacotes com informações de *clock* continuamente. Quanto maior o tempo em que o receptor mantém seu rádio ligado, maior será o consumo de energia. Seja recebendo pacotes para sincronização ou apenas ouvindo o canal, o receptor poderá mantém seu rádio ligado por extensos períodos de tempo sem necessidade (PAPADOPOULOS et al., 2016). O rádio receptor representa um dos maiores consumos de energia em um nó, por este motivo, o controle e a redução do tempo de guarda, bem como técnicas que envolvem aprimoramento no processo de sincronização dos nós mostram-se parâmetros promissores no que diz respeito a eficiência energética da rede.

Diversos trabalhos encontrados na literatura utilizam estas técnicas. Analisar a quantidade de EBs a serem enviados, bem como o período necessário para o envio, foi a técnica utilizada por (CHEN et al., 2014) na estratégia de enviar múltiplos *beacons* com o objetivo de reduzir o consumo de energia no processo de sincronização.

Ao estender o modelo utilizado em (CHEN et al., 2014), é possível reduzir o consumo total de energia analizando a comparação, entre a energia gasta com mais frequentes sincronizações e a economia de energia alcançada pela redução do tempo de escuta, quando o rádio permanece no modo `Idle`, no processo de detecção de alarmes (NADAS et al., 2016).

Também a partir de um agendamento específico, os EBs são utilizados apenas quando o nó está se associando à rede. O processo de sincronização do nó receptor é feito no momento do recebimento do ACK enviado pelo nó coordenador. Esta foi a técnica proposta por (KIM; KIM, 2018) para evitar colisões e reduzir o consumo de energia otimizando a sincronização.

Diferentemente dos trabalhos anteriores, o objetivo deste trabalho é reduzir o consumo de energia gasto no processo de sincronização com o envio de um número maior de EBs de tamanho reduzido, através da técnica proposta por (CHEN et al., 2014), de modo que o nó consiga receber os pacotes em situações em que a recepção não seria possível utilizando técnicas comuns de sincronização. Devido às topologias aplicadas a este trabalho e à natureza da rede, o modelo de tráfego e agendamento seguem o esquema centralizado, como descrito na Seção 2.2.

3 DESENVOLVIMENTO

3.1 SISTEMA OPERACIONAL CONTIKI

Um sistema operacional para dispositivos IoT deve, em geral, possuir alguns requisitos como: baixa utilização de memória, suporte a multi tarefas e gerenciamento do consumo de energia (DUNKELS et al., 2004).

Muitos sistemas operacionais (OS, do inglês *Operating System*) têm sido desenvolvidos para atender a estes requisitos, além de muitos estudos e discussões sobre qual o melhor OS a ser utilizado (SABRI et al., 2017), dentre eles: FreeRTOS (BARRY, 2003), Contiki (DUNKELS et al., 2004), TinyOS (LEVIS et al., 2005), RIOT (BACCELLI et al., 2013), além do OpenWSN (WATTEYNE et al., 2012) e o projeto Zephyr (PROJECT, 2017).

Contiki é um sistema operacional embarcado de código aberto, escrito em linguagem C, possui uma baixa utilização dos recuros de memória, suporte de *hardware*, possui suporte para um grande número plataformas e é apoiado pela indústria e comunidade acadêmica com mais de 2.300 repositórios e uma alta classificação com mais 3.000 estrelas no seu repositório oficial do Github (DUNKELS et al., 2019). Além disso, possui uma implementação TSCH e 6TiSCH, uma rede IP (do inglês, *Internet Protocol*), com baixo consumo de energia, que suporta os protocolos 6LoWPAN (do inglês, *IPv6 over Low Power Wireless Personal Area Networks*), RPL (do inglês, *Routing Protocol for Low-Power and Lossy Networks*), CoAP (do inglês, *Constrained Application Protocol*) e MQTT (do inglês, *Message Queue Telemetry Transport*). Além da implementação TSCH, apresenta também uma ferramenta útil, o Cooja *Nework Simulator*, um ambiente de simulação que permite simular grandes redes em larga escala com todos os detalhes dos dispositivos de *hardware* emulados.

3.1.1 Powertrace and Energest

O Contiki possui também um conjunto de ferramentas de *software* para a estimativa de energia: *Powertrace e Energest*. O Energest rastreia o tempo que os componentes de *hardware* permanecem ligados, tais como o rádio e a CPU, e o Powertrace reporta estes valores (DUNKELS et al., 2011).

A energia consumida em cada um dos modos pode ser calculada com os

parâmetros físicos de corrente e tensão do *hardware*, e com o tempo total de simulação.

Tabela 1 – Exemplo do Powertrace mostrando as estatísticas de uma simulação com duração de 180 minutos com intervalo de exibição dos parâmtros a cada 60 s.

All CPU	All Lpm	All Tx	All Rx	CPU	Lpm	Tx	Rx
46157	6017166	1045	134830	11875	1954194	42	45024

Fonte: Autoria própria.

A Tabela 1 mostra um trecho de simulação total de 180 segundos obtido através de simulação no Cooja configurado para exibir as estatíticas a cada 60 segundos. O tempo é exibido em uma unidade de tempo chamada *ticks* por segundo, e a duração de cada *tick* depende do oscilador de cada plataforma. Em uma plataforma com um oscilador de 32768 Hz, 1 *tick* equivale a 1/32768. As quatro primeiras colunas mostram os *ticks* gastos nos modos {Rx, Tx, CPU, Lpm} desde o início da simulação, onde Lpm (modo de baixo consumo, do inglês *Low Power Mode*) indica o tempo gasto no estado inativo da CPU. As quatro últimas colunas apenas os *ticks* consumidos no último ciclo de 60 segundos. A partir destes números e das informações de *hardware* da plataforma é possível calcular a potência consumida em um determinado modo e no período de tempo desejado da seguinte maneira:

$$P = \frac{T_{mode} V I_{mode}}{T_{\text{sim}}}, \tag{2}$$

onde T_{mode} é o tempo despendido em cada *mode*, V é a tensão de entrada, I_{mode} é a corrente consumida em cada *mode* e T_{sim} é o tempo total de simulação.

3.2 MODELO DE CONSUMO

Como apresentado na Seção 1.3.2 um dos objetivos deste trabalho é a elaboração de um modelo analítico de consumo energético para os nós sensores. Inicialmente foram utilizados os parâmetros da plataforma Z1, que integra um sistema de comunicação sem fio CC2420 e um microcontrolador MSP430 ambos da *Texas Instruments*, simulada no *Cooja* para analisar as primeiras impressões sobre o consumo dos nós e validar o modelo apresentado. Em seguida, o modelo analítico é aplicado em uma plataforma real de testes, neste caso CC2650 Launchpad da *Texas Instruments*, também para análise do consumo e aplicação das técnicas para redução de energia. Os resultados obtidos para as plataformas são apresentados na Seção 4.

Para estimar o modelo de consumo de energia do nó sensor a partir dos modos {Rx, Tx, CPU, Lpm} apresentados pelo Powertrace é necessário analisar o consumo em

separado da CPU e o consumo do rádio assim como os tempos gastos pela CPU e pelo radio nos estados apresentados na Figura 2, que ocorrem dentro do *timeslot*.

Como os estados para os modos variam durante o *timeslot*, varia também o consumo de energia. As Tabelas 2 e 3 mostram os tempos ativos destes modos em um *timeslot* do tipo `TxDataRxAck` e em um *timeslot* do tipo `RxDataTxAck` respectivamente.

Tabela 2 – Tempos em μs em que os modos {Rx, Tx, CPU, Lpm} permaneceram ativos nos estados considerados dentro de um *timeslot* do tipo `TxDataRxAck`.

Estado	Tempo Total	CPU	Tx	Rx
TxDataOffset	2791	930	2791	0
TxDataPrepare	657	657	657	0
TxDataReady	219	219	219	0
TxDataListen	333	166,5	333	0
TxData	$N \times 32$	$N \times 32$	0	$N \times 32$
RxAckOffset	3395	1697,5	0	3395
RxAckPrepare	70	70	0	70
RxAckReady	35	35	0	35
RxAckListen	0	0	0	0
RxAck	1900	633,33	0	1900
TxProc	550	500	0	0
EndOfSlot	$N \times 32$	0	0	0

Fonte: (WATTEYNE, 2012).

Tabela 3 – Tempos em μs em que os modos {Rx, Tx, CPU, Lpm} permaneceram ativos nos estados considerados dentro de um *timeslot* do tipo `RxDataTxAck`.

Estado	Tempo Total	CPU	Tx	Rx
RxDataOffset	3280	820	0	2791
RxDataPrepare	70	70	0	70
RxDataReady	50	50	0	50
RxDataListen	552	552	0	233
RxData	$N \times 32$	$N \times 32$	0	$N \times 32$
TxAckOffset	3280	820	3280	0
TxAckPrepare	258	258	70	0
TxAckReady	129	129	129	0
TxAckDelay	0	0	0	0
TxAck	1900	166	0	333
RxProc	550	500	0	0
EndOfSlot	$N \times 32$	0	0	0

Fonte: (WATTEYNE, 2012).

A análise da Tabela 2 demonstra que o tempo total gasto pelo estado `TxDataOffset` dentro do *timeslot* de 15 ms corresponde a 2791 μs. Neste período de

tempo de 2791 μs, a CPU se manteve ativa por 930 μs e Tx por 2791 μ. O modo Rx não é ativado durante este estado e Lpm representa a diferença de tempo entre o tempo total gasto no estado e o tempo ativo do modo CPU.

O tempo gasto para o *timeslot* `TxData` e `RxData`, são análogos ao das Tabelas 2 e 3 respectivamente, porém, sem a presença de ACK. Para o *timeslot* `RxIdle`, são considerados apenas os tempos gastos pela CPU e pelo rádio durante o tempo de guarda e, para o *timeslot* `Sleep`, considera-se os componentes em baixo consumo por toda a duração do *timeslot* (15 ms).

3.2.1 Modelo Analítico de Consumo

O modelo de consumo a ser considerado deve então computar os tempos despendidos em cada modo, para cada tipo de *timeslot* e para cada estado que ocorre dentro do *timeslot*. A Tabela 4 mostra a dinâmica do cálculo a ser implementada.

Tabela 4 – Dinâmica para o cálculo do consumo de potência para o modo Rx, para os *timeslots* `TxDataRxAck` e `RxDataTxAck`, nos estados apresentados nas Tabelas 2 e 3.

Mode	Type	State	Mode	Type	State
		TxDataOffset			RxDataOffset
		TxDataPrepare			RxDataPrepare
		TxDataReady			RxDataReady
		TxDataListen			RxDataListen
		TxData			RxData
Rx	TxDataRxAck	RxAckOffset	Rx	RxDataTxAck	TxAckOffset
		RxAckPrepare			TxAckPrepare
		RxAckReady			TxAckReady
		RxAckListen			TxAckDelay
		RxAck			TxAck
		TxProc			TxProc
		EndOfSlot			EndOfSlot

Fonte: Autoria própria.

De acordo com a Tabela 4, é possível concluir que a potência total consumida pelo nó sensor pode ser calculada pela soma das potências consumidas em todos os modos {Rx, Tx, CPU, Idle}, para todos os tipos de *timeslots* {`RxDataTxAck`, `RxData`, `RxIdle`, `TxDataRxAck`, `TxData`, `Sleep`} e por todos estados {`TxDataOffset`, $\cdots$, `EndOfSlot`} presentes no *timeslot*.

A partir das informações contidas nas Tabelas 2, 3 e 4, , é possível encontrar um conjunto de equações que representa o modelo de consumo de energia, como apresenta-se

nas equações (3), (4) e (5).

Para cada *timeslot type* $\in U = \{$`RxDataTxAck`, `RxData`, `RxIdle`, `TxDataRxAck`, `TxData`, `Sleep`$\}$, a variação da potência consumida $\overline{P}_{type}$ é calculada como

$$\overline{P}_{type} = \sum_{mode \in Z} E_{mode,type}/T_{\text{slot}}, \tag{3}$$

onde $Z = \{$Rx, Tx, CPU, Idle$\}$ é o conjunto dos modos de consumo, $E_{mode,type}$ é a energia consumida para um dado *mode* e *timeslot type*, e T_{slot} é o tempo total de um *timeslot* TSCH. $E_{mode,type}$ é calculado como

$$E_{mode,type} = \sum_{state \in S} VI_{mode} T_{state,mode}, \tag{4}$$

onde $S = \{$`TxDataOffset`, $\cdots$, `EndOfSlot`$\}$ é o conjunto de estados em cada *timeslot type* e $T_{state,mode}$ é o tempo gasto para cada *mode* para um dado *state*.

Como cada *timeslot* t pode ser atribuído a um diferente *type*, ele pode ser representado por uma 2-tupla $(t, type)$. Uma tupla é uma seqüência finita também chamada de lista ordenada de objetos. Apesar de muito parecida com uma lista, a principal diferença é que os objetos que compõem a tupla não podem ser alterados. As tuplas trabalham com estrutura ao invés da ordem dos elementos. Então se o primeiro elemento da tupla for um *timeslot* do tipo `TxDataRxAck`, sua posição não poderá ser invertida. Assim, pode-se calcular o consumo médio de energia para cada *mode* como

$$\overline{P}_{mode} = \sum_{t=0}^{Q-1} \frac{E_{mode,\Delta(t,type)}}{QT_{\text{slot}}}, \tag{5}$$

onde Q é o número total de *timeslots*, e Δ é a função que retorna o *type* da 2-tupla $(t, type)$.

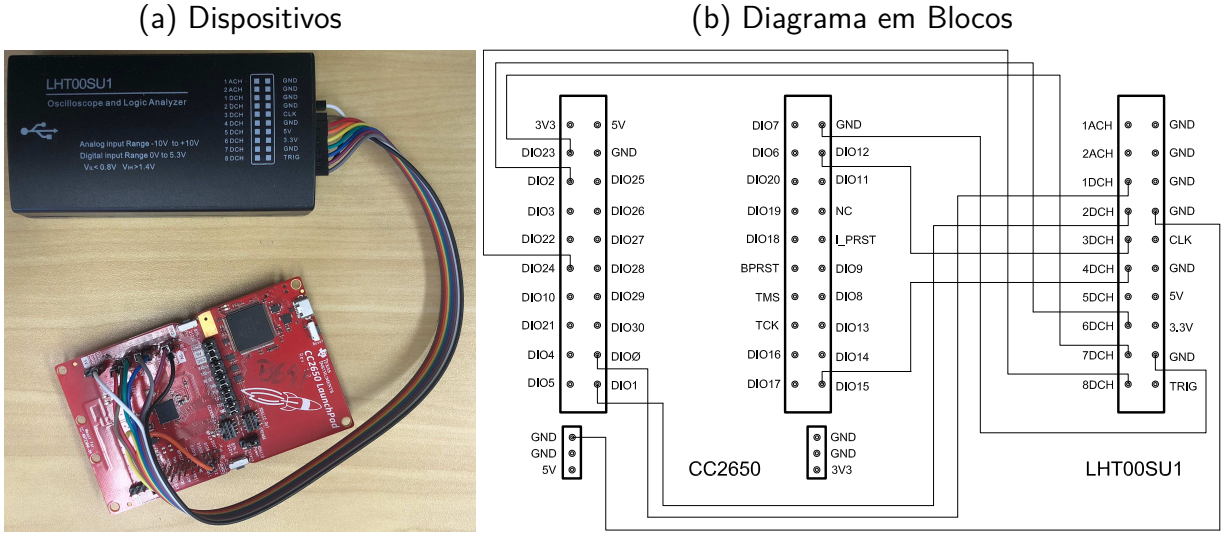
3.2.2 Extensão para Plataforma CC2650 Launchpad

Ao estender o modelo de consumo para a plataforma CC2650 Launchpad da *Texas Instruments*, percebeu-se que os tempos despendidos nos estados dentro dos *timeslots* para o rádio e a CPU eram diferentes da plataforma Z1 simulada no Cooja, já que as simulações realizadas demonstraram diferenças significativas entre o modelo de consumo proposto e a simulação.

Para entender como a energia era gasta pelo rádio e pela CPU em cada intervalo de tempo, foi necessário o uso de ferramentas que fizessem a captura de dados, mostrando

os modos utilizados e os tempos gastos pelos mesmos dentro dos *timeslots*.

Figura 4 – Plataforma CC2650 conectada ao Analisador Lógico para captura dos tempos despendidos em cada modo.



Fonte: Autoria própria.

O cenário é o da Figura 4 e consiste em um analisador lógico, conectado aos pinos digitais da plataforma CC2650 Launchpad, que em conjunto com adaptações realizadas no *firmware* permite que sejam capturadas a duração de cada modo e estado para qualquer *timeslot*.

Para facilitar a compreensão desse procedimento, a Tabela 5 apresenta os estados de depuração inseridos no código do Contiki TSCH, para que os intervalos de tempo em que cada componente de *hardware* permanecesse ativo em um *timeslot* e pudessem ser medidos.

Os intervalos de tempo medidos entre os estados, para um *timeslot* do tipo `RxDataTxAck` são mostrados na Figura 5 e na Tabela 6.

Com o objetivo de apresentar intervalos de tempo mais precisos para os tempos despendidos em cada estado e, ajustar estes tempos em um formato padrão conhecido, foram aplicados os resultados medidos em comparação com o modelo de *timeslot* de 15 ms, obtendo os resultados ilustrados na Figura 6 e na Tabela 6.

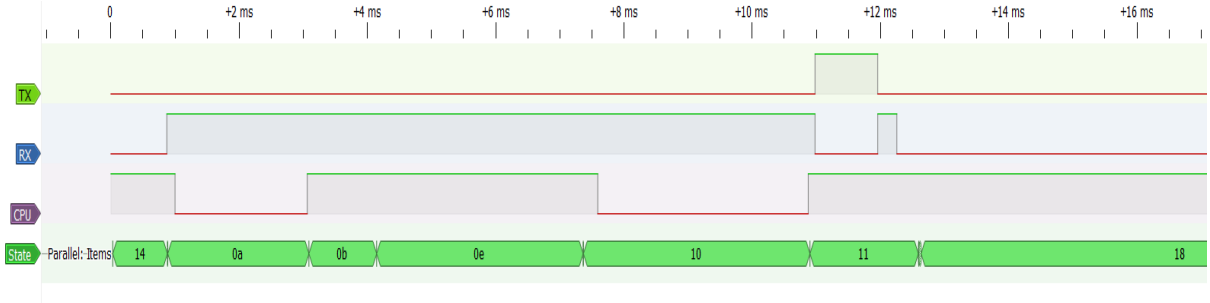
A comparação demonstra que os tempos despendidos nos estados 0x13, 0x14, 0x15 e 0x0A possuem pequenas diferenças em relação a `TsRxOffset` definido para um tempo de guarda $t_g = 1800$ μs, período em que o rádio e a CPU se preparam no início do *timeslot*. O estado 0x0B apresenta um tempo próximo a metade do tempo de guarda

Tabela 5 – Estados de Depuração

Estado	Número	Estado	Número
TX_INIT	0x01	RX_IDLE_RX_OFF	0x0D
TS_TX_OFFSET	0x02	PACKET_DETECTED	0x0E
TS_TX_OFFSET_AFTER_TRANSMIT	0x03	PACKET_RECEIVED	0x0F
TS_RX_ACK_DELAY	0x04	RX_OFF_AFTER_PACKET_RECEIVED	0x10
TS_ACK_WAIT	0x05	RX_ACK_SEND	0x11
ACK_RECEIVED	0x06	RX_END	0x12
RADIO_OFF_AFTER_ACK_RECEIVED	0x07	SLOT_START	0x13
RADIO_OFF_END_TX_SLOT	0x08	SLOT_START_TURN_RADIO_ON	0x14
TX_END	0x09	SLOT_START_RADIO_IS_ON	0x15
RX_INIT	0x0A	SLOT_END	0x16
TS_RX_OFFSET	0x0B	SLOT_SCHEDULE	0x17
RX_IDLE	0x0C	SLOT_OPERATION_END	0x18

Fonte: Autoria própria

Figura 5 – Slot `RxDataTxAck` medido no Analizador Lógico



Fonte: Autoria própria.

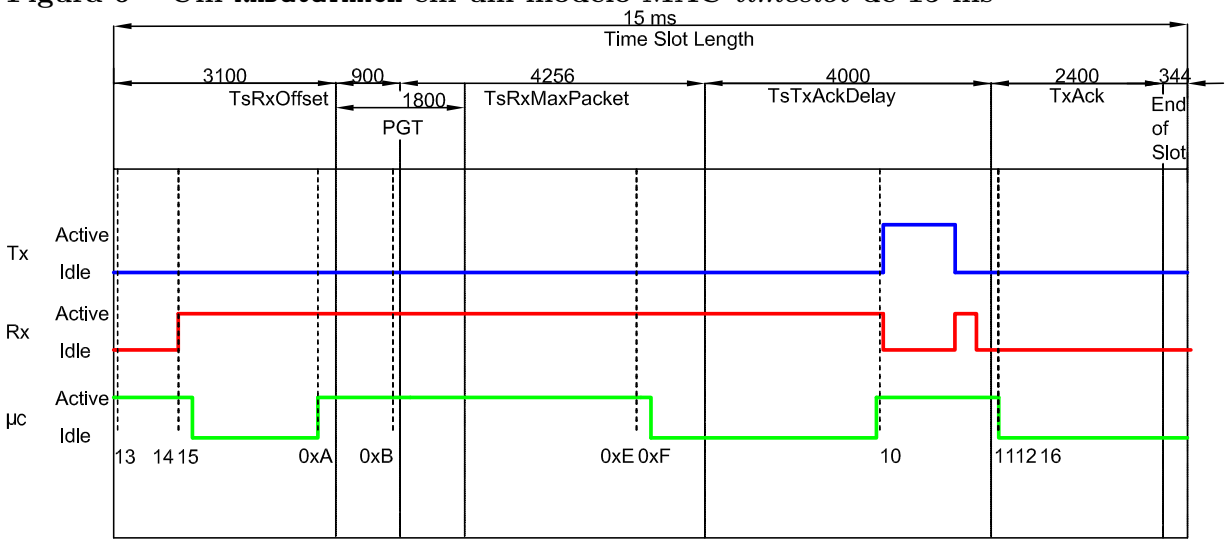
Tabela 6 – Tempos em μs em que os modos permaneceram ativos para um *timeslot* do tipo `RxDataTxAck`

Estado	Tempo Total	CPU	Tx	Rx
0x13	29,12	29,12	0	0
0x14	851,62	851,62	0	7,37
0x15	6,12	6,12	0	6,12
0x0A	1957,75	139,37	0	1957,75
0x0B	1056,25	1056,25	0	1056,25
0x0E	$N \times 34,8592$	$N \times 34,8592$	0	$N \times 34,8592$
0x0F	4,87	4,87	0	4,87
0x10	3532,87	250,37	0	3532,87
0x11	1690	1690	376,87	978,25
0x12	5,37	5,37	0	0
0x16	5,5	5,5	0	0

Fonte: Autoria própria.

(PGT/2), período o qual o rádio liga seu receptor a procura de pacotes, contudo, não é exato.

Figura 6 – Um `RxDataTxAck` em um modelo MAC *timeslot* de 15 ms



Fonte: Autoria própria.

O estados 0xE e 0xF são proporcionais ao tamanho do pacote recebido em `TsRxMaxPacket`. Entretanto, o período para o qual o rádio deve aguardar para enviar um ACK, configurado em 0x10, possui um período de tempo menor que o especificado em `TsTxAckDelay` e, o envio do ACK, ajustado em 0x11, acontece adiantado do tempo previsto em `TxAck`. Além de que é possível notar um comportamento do receptor que é ativado após o envio do ACK, período o qual deveria permanecer inativo.

As diferenças encontradas no *timeslot* `RxDataTxAck` e o modelo de *timeslot* definido em IEEE 802.15.4e sugeriram que o comportamento dos demais *timeslots* poderiam apresentar diferenças também entre si, para os mesmos períodos de tempos medidos, impossibilitando a deriavação utilizada anteriormente na Seção 3.2.1 e, tornado obrigatório a medida de tempo nos outros *timeslots* utilizados.

Seguindo este mesmo procedimento, as medidas de tempo despendidos em um *timeslot* do tipo `RxData` é mostrado na Tabela 7. A Tabela 8 mostra o tempo em um *timeslot* do tipo `RxIdle`, as Tabelas 9 e 10 mostram as medidas de tempo em *timeslots* do tipo `TxDataRxAck` e `TxData` respectivamente. Por fim, a Tabela 11 mostra o tempo despendido em um *timeslot* do tipo `Sleep`.

3.3 A ESTRATÉGIA DO *GUARD BEACON*

Uma vez obtidos os tempos despendidos pelo rádio e pela CPU em cada tipo de *timeslot*, tornou-se possível calcular a energia consumida utilizando o modelo de consumo

Tabela 7 – Tempos em μs em que os modos permaneceram ativos para um *timeslot* do tipo `RxData`

State	Total Time	CPU	Tx	Rx
0x13	32,25	32,25	0	0
0x14	849,37	849,37	0	7,25
0x15	6	6	0	6
0x0A	1958,37	132,12	0	1958,37
0x0B	1459,25	1459,25	0	1459,25
0x0E	$N \times 34,8592$	$N \times 34,8592$	0	$N \times 34,8592$
0x0F	4,87	4,87	0	4,87
0x10	632,75	632,75	0	294,37
0x12	5,37	5,37	0	0
0x16	6,25	6,25	0	0

Fonte: Autoria própria.

Tabela 8 – Tempos em μs em que os modos permaneceram ativos para um *timeslot* do tipo `RxIdle`

State	Total Time	CPU	Tx	Rx
0x13	29,12	29,12	0	0
0x14	846,75	846,75	0	7,5
0x15	6	6	0	6
0x0A	1958,37	129,12	0	1958,37
0x0B	2202,12	2202,12	0	2202,12
0x0C	495,87	495,87	0	166,37
0x0D	4,75	4,75	0	0
0x12	5,37	5,37	0	0
0x16	5,37	5,37	0	0

Fonte: Autoria própria.

proposto. Contudo, como apresentado na Seção 2.3.2, para manter o sincronismo entre si os nós precisam enviar e receber EBs periodicamente para manter o erro de sincronismo dentro de uma faixa em que a comunicação continue possível.

Pela configuração padrão do Contiki, o EB utilizado no início de formação da rede para associação dos nós e para troca de informações de sincronismo possui tamanho de 37 Bytes, com um tempo de guarda padrão de $t_g = 2200$ μs com um período de envio de *Beacon*, ou intervalo de sincronização IS = 60s.

Quanto maior o tempo de guarda e maior o EB usado para sincronização, maior será o tempo em que o rádio irá permanecer com o receptor ligado, causando consumo desnecessário de energia para receber e processar dados que não são informações úteis.

Tabela 9 – Tempos em μs em que os modos permaneceram ativos para um *timeslot* do tipo `TxDataRxAck`

State	Total Time	CPU	Tx	Rx
0x13	32	32	0	0
0x14	874,12	871,12	0	7,5
0x15	5,87	5,87	0	5,87
0x01	$(N+13,57)\times 32$+2836.62	$(N+13,57)\times 32$+224	$(N+13,57)\times 32$	2836,62
0x03	3270,62	132,87	0	3270,62
0x04	812,5	812,5	0	812,5
0x05	617,75	617,75	0	617,75
0x06	5,37	5,37	0	5,37
0x07	682,25	682,25	0	348,37
0x08	3822,37	3822,37	0	0
0x09	5,62	5,62	0	0
0x16	6,12	6,12	0	0

Fonte: Autoria própria.

Tabela 10 – Tempos em μs em que os modos permaneceram ativos para um *timeslot* do tipo `TxData`

State	Total Time	CPU	Tx	Rx
0x13	26	26	0	0
0x14	855,62	855,62	0	7,25
0x15	5,75	5,75	0	5,75
0x01	$(N+13,57)\times 32$+2836,62	$(N+13,57)\times 32$+224	$(N+13,57)\times 32$	2836,62
0x03	602,12	602,12	0	285,12
0x08	3847,75	3847,75	0	0
0x09	5,5	5,5	0	0
0x16	6,25	6,25	0	0

Fonte: Autoria própria.

Tabela 11 – Tempos em μs em que os modos permaneceram ativos para um *timeslot* do tipo `Sleep`

State	Total Time	CPU	Tx	Rx
0x18	15000	0	0	0

Fonte: Autoria própria.

Apenas reduzir o tempo de guarda para diminuir o consumo energético pode aumentar a probabilidade de perda de pacotes, principalmente EBs utilizados para o sincronismo e, dependendo do intervalo de sincronização e do *clock drift*, ocorrerá um tempo de guarda limite, onde qualquer configuração de tempo de guarda abaixo deste limite, impossibilitará a comunicação e o sincronismo entre os nós.

Diante disto, tornou-se evidente que a implementação de uma estratégia para

reduzir o comprimento dos *Beacons* utilizados no processo de sincronização aplicada a uma técnica para garantir a comunicação entre os nós em tempos de guarda reduzidos seriam parâmetros promissores para a redução do consumo de energia.

Com o objetivo de reduzir o tempo de guarda sem comprometer a sincronização, um novo esquema chamado *Guard Beacon* foi proposto por (CHEN et al., 2014). A estratégia utilizada analisa a quantidade de EBs a serem enviados, bem como o período necessário para o envio. Enviando múltiplos beacons, a estratégia tem como objetivo a redução do consumo de energia no processo de sincronização. O número ótimo de *Beacons* a serem enviados é dado por (CHEN et al., 2014)

$$N_{opt} = \left\lfloor \sqrt{\frac{t_a P_l}{T_b P_s}} \right\rceil, \tag{6}$$

Onde $\lfloor \rceil$ são os operadores matemáticos para delimitar a expressão, $t_a = K\sqrt{T_s^2\sigma_f^2 + \sigma_\tau^2 + \sigma_\theta^2}$, T_s é o intervalo de *Beacon*, σ_f é a taxa estimada do *clock drift*, σ_τ é o desvio estimado do atraso na entrega de mensagens, σ_θ é o desvio estimado do *clock*, P_l é a potência consumida no modo `Idle`, T_b é a duração do *Beacon* e P_s é o consumo de potência para a transmissão de dados.

O tempo de envio dos *Beacons* $x'_1, x'_2, \cdots, x'_{N_{opt}}$ é dado por

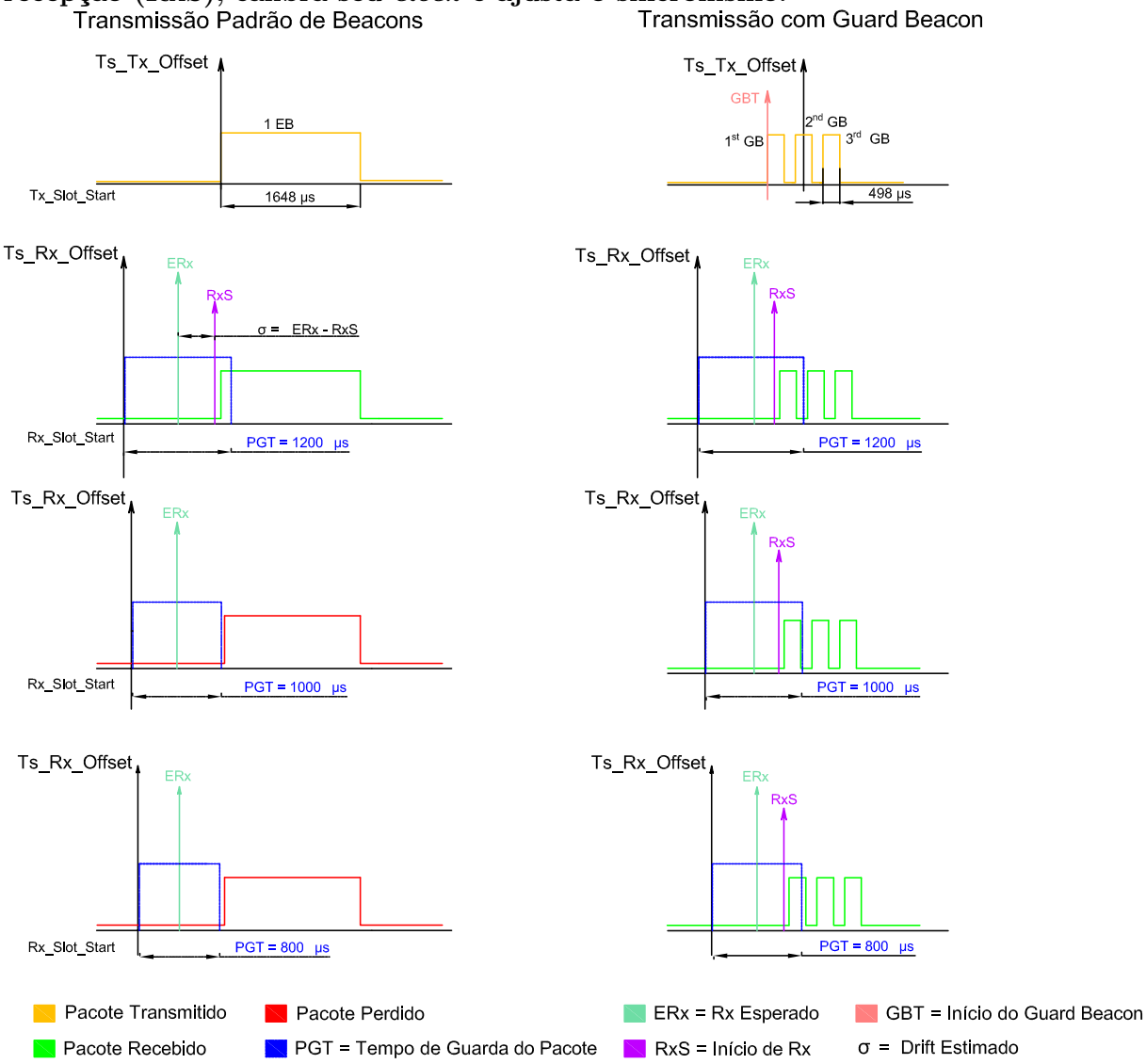
$$x'_i = \sqrt{2}\sigma_e erf^{-1}\left(\frac{2i}{N_{opt}} - 1\right), \tag{7}$$

onde $erf^{-1}(\cdot)$ representa a função erro inversa.

O esquema utilizado por (CHEN et al., 2014) demonstra que a estratégia do *Guard Beacon* pode reduzir o consumo de energia no processo de sincronização em cerca de 40%. Planejando reduzir o consumo de energia na rede TSCH, a estratégia do *Guard Beacon* foi implementada no mecanismo de envio de EBs do TSCH e seu esquema de funcionamento é mostrado na Figura 7.

Nesta técnica, chamada de GB deste ponto em diante, os *Beacons* enviados no processo de sincronização não possuem nenhuma mensagem útil, apenas 2 *Bytes* são enviados, um indicando se tratar de um *frame* do tipo *Guard Beacon Frame* e outro para identificação do número do *Guard Beacon* enviado. Desta maneira os nós podem manter a sincronização e economizar energia reduzindo o tempo em que a CPU e rádio se mantém ativos para recepção e processamento do pacote, em comparação com um EB de 37 Bytes de comprimento.

Figura 7 – Estratégia do Guard Beacon: Os *Guard Beacons* são enviados avançados de `TsTxOffset` em GBT. O receptor aumenta a probabilidade de receber um dos *Guard Beacons* com tempos de guarda (PGT) menores. O *Drift* estimado é calculado através da diferença no tempo da receção esperada (ERx) e o início da recepção (RxS), calibra seu *clock* e ajusta o sincronismo.



Fonte: Autoria própria

No entanto, houve um contratempo tentando enviar os *Beacons* extras nos tempos ótimos utilizados em (7). Embora o tempo descrito na documentação do TSCH para envio de 1 Byte seja de 32 μs, a API (Interface de Programação de Aplicativos) somente retorna de uma transmissão após aproximadamente 434 μs, cerca 13,57 Bytes, conforme pode ser observado nos tempos medidos nas Tabelas 9 e 10 apresentadas na Seção 3.2.2.

O tempo de guarda utilizado para coletar os dados em um *timeslot* de 15 ms foi de 1800 μs, então não seria possível enviar mais de 3 GBs, uma vez que o comprimento

de 3 Bytes somados com o tempo de atraso da API resulta num tempo total de 1618 μs. Também não seria possível usar o tempo obtido a partir da equação (7), pois uma vez enviados o 3 GBs subsequentemente, quase todo o tempo de guarda disponível iria ser usado.

A abordagem a ser adotada foi o envio dos 3 GBs subsequentemente de qualquer maneira, antecipados do momento exato da transmissão após o início do *timeslot*. Na Seção 4.2.3 é mostrado que, embora a implementação do mecanismo do *Guard Beacon* não seja estrita, a redução no consumo de energia em comparação ao envio de apenas um *Beacon* é significativa.

Além disso, em uma rede com um maior número de nós o *clock drift* pode ocorrer em direções opostas (ou seja, + 10ppm para um nó e -12ppm para outro nó em relação à fonte de *clock* da rede), o primeiro quadro GB é enviado antecipadamente de `TsTxOffset`, aumentando a probabilidade de que todos os nós da rede recebam pelo menos um dos três GBs, ainda conforme mostra a Figura 7.

3.3.1 Alterações de Firmware no Contiki TSCH

A estratégia do *Guard Beacon* adotada deve então, implementar o mecanismo de envio de 3 GBs subsequentemente, cada GB com comprimento de 1 Byte. O envio deve acontecer antecipadamente ao momento esperado para a transmissão, de forma que nós recebam pelo menos um dos GBs. O código com as alterações realizadas para possibilitar a estratégia está disponível em (RAYEL; SORDI, 2019).

A primeira e mais importante é a que trata da criação do Guard Beacon propriamente dito. Ela ocorre no código principal do TSCH, que é responsável pelo gerenciamento de processos tais como associação do nó à rede, envio periódico de *Beacons* e manter o nó ativo na rede. Além do processo de negociação para pacotes pendentes que entram ou saem do nó. As alterações ocorridas neste *firmware* foram:

- Criação de uma *Thread* exclusiva para o envio periódico de *Guard Beacons*;
- Definição de um período para o envio de *Guard Beacons* chamado `TSCH_GB_PERIOD`;
- Criação de um pacote de comprimento de 1 Byte chamado `GUARD_BEACON_FRAME`;
- Definição para o envio de 3 *Guard Beacons* apenas se nó for o coordenador, conforme o `TSCH_GB_PERIOD` configurado;

- Estabelecer a regra para que quando desabilitada a estratégia do *Guard Beacon*, o envio de EBs obedecerá a configuração padrão do Contiki TSCH.

Outras alterações ocorreram no código do *project-conf.h*. Nele estão contidas as informações do projeto que podem ser definidas de acordo com a configuração desejada, ou pré-definidas pelo TSCH, tais como configuração do tempo de guarda, comprimento do *slotframe*, período do EB, sequência de salto de canal e etc. As principais definições consideradas com relação à estratégia do *Guard Beacon* foram:

- Habilitar ou Desabilitar a Estratégia do Guard Beacon na rede - `#define GUARD BEACON_(1)`;
- Definir o tamanho em Bytes do quadro do *Guard Beacon* - `GUARD_BEACON_FRAME`;
- Definição do intervalo de envio dos *Guard Beacons* - `TSCH_GB_PERIOD`;
- Definição do instante de tempo exato para o início da transmissão dos *Guard Beacons* - `GUARD_BEACON_TIME`;

Também foram realizadas alterações no processo de transmissão e recepção dos pacotes.

- Alterações na *Thread* de transmissão do TSCH:
  - Realiza o teste lógico para verificar se o *Guard Beacon* está habilitado ou não;
  - Quando não está habilitado, envia EBs de acordo com a configuração padrão do Contiki TSCH;
  - Quando habilitado, realiza o teste lógico para verificar se o pacote a ser enviado é de *broadcast*;
  - Se o pacote é de *broadcast*, envia o pacote apenas uma vez;
  - Caso contrário, é um EB. Então é verificado o tamanho do pacote, se maior que 1 Byte, o pacote é enviado apenas uma vez. Se o pacote é menor que 1 Byte, e o nó for cordenador:
    - Identifica o primeiro beacon como `beacon_id = 0x11`. Então cria o agendamento da transmissão para:

$$t = \texttt{TsTxOffset - GUARD\_BEACON\_TIME} \tag{8}$$

Prepara o pacote, envia o primeiro EB e limpa a fila de transmissão;

  * Atualiza o pacote para o segundo EB e o identifica como `beacon_id = 0x22`. Cria o agendamento de envio para:

$$t = \texttt{TsTxOffset} \tag{9}$$

    Prepara o pacote, envia o segundo EB e limpa a fila de transmissão;
  * Atualiza o pacote para o terceiro EB e o identifica como `beacon_id = 0x33`. Cria o agendamento de envio para

$$t = \texttt{TsTxOffset + GUARD\_BEACON\_TIME} \tag{10}$$

    Prepara o pacote, envia o terceiro EB e limpa a fila de transmissão;

- Alterações na *Thread* de recepção do TSCH:
  - Após o pacote ser detectado, realiza o teste lógico para verficar se o pacote é um *Guard Beacon*;
  - Se for um *Guard Beacon*, o pacote é forçado para ser reconhecido como um pacote *broadcast*, pois o `GUARD_BEACON_FRAME` não é um *frame* válido, e a análise do pacote (*Parse*) não funciona e o pacote seria descartado. O pacote é recebido e identificado por seu `beacon_id`;
  - Se não é um *Guard Beacon*, o pacote é recebido e o *frame* é analisado (*Parse*);
  - Define a correção do *drift* apenas para o nó que não é coordenador;
  - Define o *drift* estimado como a diferença entre o tempo esperado e o tempo do início recepção do *Guard Beacon*, com base na sua identificação:
    * Se o *Guard Beacon* recebido for o `beacon_id = 0x11`, então

$$\textit{drift estimado} -= \texttt{GUARD\_BEACON\_TIME} \tag{11}$$

    * Se `beacon_id = 0x22`,

$$\textit{drift estimado} = \texttt{GUARD\_BEACON\_TIME} \tag{12}$$

    * E se for o terceiro *Guard Beacon* recebido, `beacon_id = 0x33`,

$$\textit{drift estimado} += \texttt{GUARD\_BEACON\_TIME} \tag{13}$$

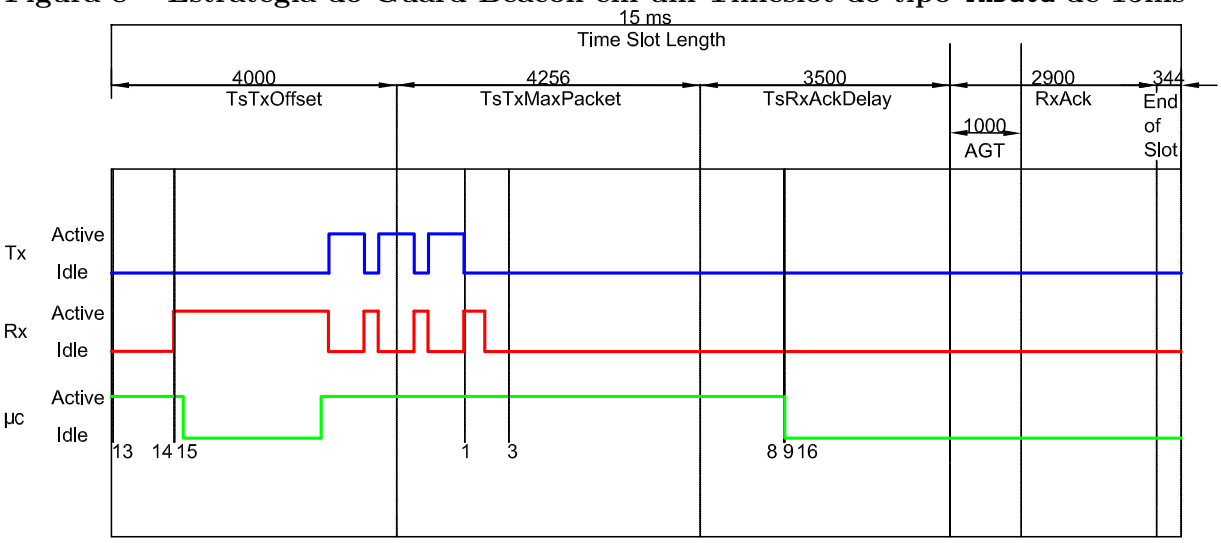
  - Altera a comparação de diferença do ASN para que suporte a defasagem de até 3 *slotframes* sem sincronização.

A estratégia do *Guard Beacon* apresenta a solução para reduzir a energia gasta no processo de sincronização. O envio de 3 GBs subsequentes, de tamanho reduzido, e avançados de `TsTxOffset` mantém os nós sincronizados e operando no menor tempo de guarda possível reduzindo o consumo de energia, conforme descrito na Seção 2.3.2.

3.3.2 Extensão do Guard Beacon para o Modelo Simulado

Com as implementações feitas no código Contiki TSCH para a estratégia do *Guard Beacon*, foi necessária uma nova simulação para obter os tempos gastos em cada etapa, analisar seu comportamento e verificar dados cruciais que diretamente implicaram na modelagem final do consumo de energia. A simulação com o tempo de guarda padrão t_g = 1800 μs do *Guard Beacon* aplicado a um *timeslot* do tipo `TxGB` é mostrada na Figura 8, e de um *timeslot* do tipo `RxGB` é mostrada na Figura 9. Os tempos despendidos em cada modo são mostrados nas Tabelas 12 e 13 respectivamente.

Figura 8 – Estratégia do Guard Beacon em um Timeslot do tipo `TxData` de 15ms

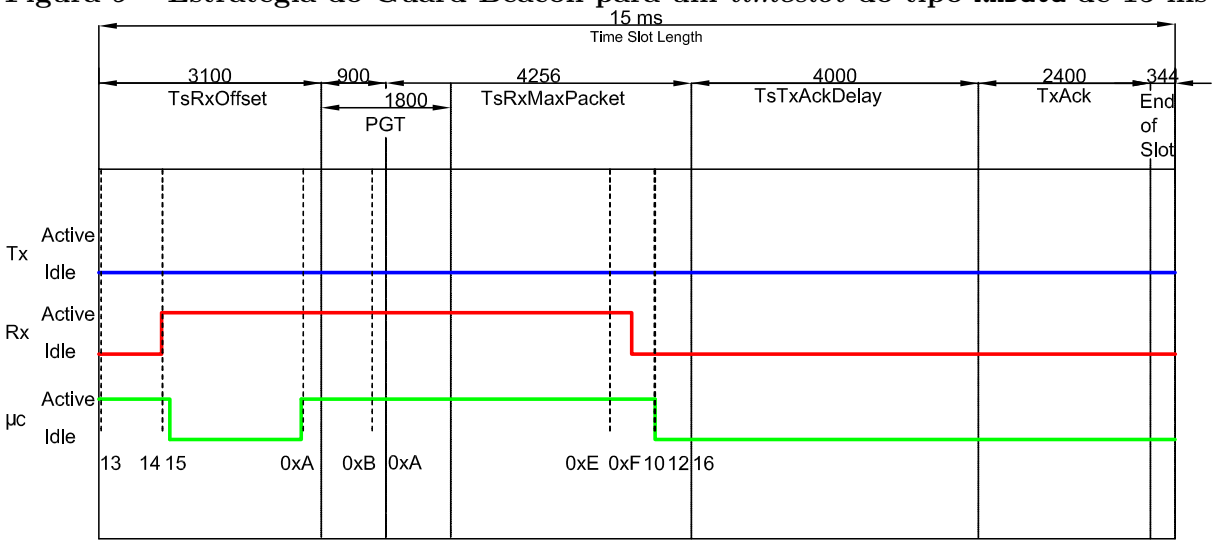


Fonte: Autoria própria

Após as investigações realizadas a respeito do comportamento dos nós em cada *timeslot*, se tornou necessária a consideração de algumas características inerentes às modificações realizadas:

- *Atraso de Tempo do Pacote*: Foi detectado que, por algum motivo, a API do rádio somente retorna de uma transmissão, após 434 μs, quando deveria entrar no modo `Idle`.

Figura 9 – Estratégia do Guard Beacon para um *timeslot* do tipo `RxData` de 15 ms



Fonte: Autoria própria

Tabela 12 – Variação dos tempos ativos em μs para um *timeslot* do tipo `TxGB`.

State	Total Time	CPU	Tx	Rx
0x13	26	26	0	0
0x14	855,62	855,62	0	7,37
0x15	5,75	5,75	0	5,75
0x01	4068,62	2133,87	1494,62	2571,87
0x03	618,25	618,25	0	279,87
0x08	3856,87	3856,87	0	0
0x09	5,5	5,5	0	0
0x16	6,25	6,25	0	0

Fonte: Autoria própria

Tabela 13 – Variação dos tempos ativos em μs para um *timeslot* do tipo `RxGB`.

State	Total Time	CPU	Tx	Rx
0x13	29	29	0	0
0x14	852,62	852,62	0	7,25
0x15	6	6	0	6
0x0A	1958,37	125,5	0	1958,37
0x0B	963,62	963,62	0	963,62
0x0E	3310,25	3310,25	0	3310,25
0x0F	4,87	4,87	0	4,87
0x10	616,37	616,37	0	298,5
0x12	5,37	5,37	0	0
0x16	5,37	5,37	0	0

Fonte: Autoria própria

- *Rx Ligada em `TxData`*: Em *timeslots* do tipo `TxData`, onde não há envio de ACK,

o rádio mantém o Rx *On* por aproximadamente 298 μs.

- *Rx Ligada Após o Pacote Recebido*: Nos *timeslots* do tipo `RxData`, independente do tamanho do pacote recebido, o rádio mantém o Rx *On* por aproximadamente 275 μs.
- *CPU Ligada Após o SLOT END*: Ao enviar e receber pacote *unicast*, a CPU permanece ativa, mesmo após o período de tempo do *timeslot* expirar. Cerca de 11,7 ms para um *timeslot* do tipo `TxDataRxAck` e 9,76 ms para um *timeslot* do tipo `RxDataTxAck`.
- *CPU Ligada Durante os timeslots* `Sleep`: Devido as operações executadas pelo Contiki OS, nos intervalos de tempo dos *timeslots* `Sleep` dentro do *slotframe*, a *CPU* é aleatoriamente ativada para instantes de tempo de aproximadamente 208 μs.

Essas considerações se mostraram de grande importância, pois a partir delas, foi possível calcular os tempos de atrasos para compor as equações de consumo do modelo estendido para a estratégia do *Guard Beacon*. Com os tempos em cada *mode* e *state* para os *timeslots type* $\in W = \{\texttt{TxGB}, \texttt{RxGB}\}$ medidos, a variação de potência consumida da rede TSCH pode ser calculada com as equações 3, 4 e 5 apresentadas na Seção 3.2.

4 RESULTADOS

Esta seção apresenta os resultados obtidos em dois cenários diferentes. No primeiro, a análise do consumo de energia de uma rede usando nós tipo Z1 através da plataforma de rede Cooja do Contiki. No segundo, uma rede usando os nós CC2650 Launchpad da *Texas Instruments*, operando na prática. Os dados da simulação, bem como os resultados simulados e calculados pelo modelo analítico, são apresentados a seguir.

4.1 PLATAFORMA Z1

Os parâmetros utilizados para simulação no Cooja e para o cálculo de acordo com o modelo analítico de consumo são apresentados nas Tabelas 14 e 15. Para melhor entendimento, os resultados obtidos com a plataforma são apresentados em um resumo descrito na Tabela 16. São apresentados os resultados numéricos de comparação de potência consumida entre os modos, no modelo analítico e simulado, bem como a redução no consumo e a vida útil da bateria.

Tabela 14 – Parâmetros de Simulação de Rede

Parâmetros de Topologia	**Valor**
Topologia	Linha, Estrela
Numero de Nós	2, 9
Espaçamento entre os Nós	20 m
Parâmetros de Simulação	**Valor**
Duração	60 min
Número de Saltos	1
Parâmetros do TSCH	**Valor**
Frequência do EB	60 s
Comprimento do Slotframe	7
Comprimento do Timeslot	15 ms
Tempo de Guarda	1200 μs - 3200 μs

Fonte: Autoria própria.

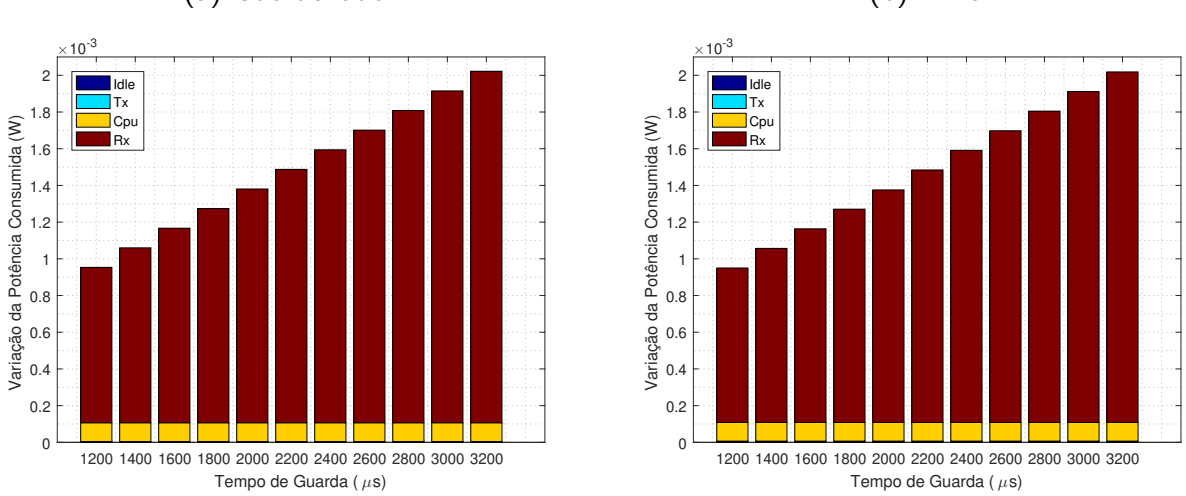
As Figuras 10 e 11 mostram a variação do consumo de potência para dois nós em linha, calculada a partir do modelo analítico e com os dados de simulação, respectivamente. É possível observar que o consumo do modelo analítico é ligeiramente maior, e que dentre os modos analisados $Z = \{\text{CPU, Idle, Tx, Rx}\}$, os maiores vilões no consumo de energia são a CPU e Rx.

Tabela 15 – Parâmetros de Simulação de Hardware

Parâmetros do *Hardware*	Valor
Consumo em Tx	17,4 mA
Consumo em Rx	18,8 mA
Consumo em Sleep	0,1 μA
Consumo da CPU Ativa	4 mA
Consumo da CPU em *Low-Power*	5 μA

Fonte: (TEXAS INSTRUMENTS, 2013).

Figura 10 – Consumo de Potência entre os Modos - Analítico - 2 Nós em Linha.



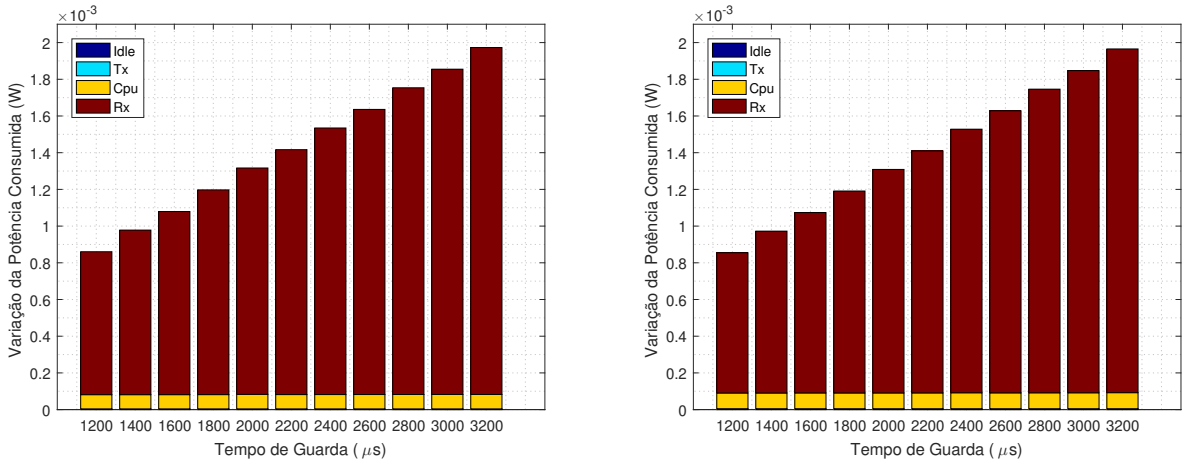
Fonte: Autoria própria.

As comparações entre o modelo analítico e o simulado são mostradas na Figura 12. Dentre os modos com o maior consumo, o modelo analítico apresenta um consumo de 8,01 % maior que o modelo simulado em Rx, e de cerca de 31,7 % para a CPU, o que representa em números um consumo de cerca de 62,43 μW e 24,55 μW respectivamente. O aumento no consumo dos modos Idle e Tx do modelo analítico sobre o simulado não apresentam diferenças siginificativas, na ordem de 2 μW tanto para o nó coordenador quanto para o nó filho.

A Figura 13 demonstra o erro percentual do modelo analítico sobre o modelo simulado. O parâmetro utilizado é a potência total do nó, calculada pela soma dos modos existentes {Rx, Tx, CPU, Idle}. A análise demonstra que no pior caso, o modelo analítico apresenta uma potência total do nó 10 % maior que no modelo simulado para um tempo de guarda $t_g = 1200$ μs, o que representa um consumo de 89,55 μW maior.

Figura 11 – Consumo de Potência entre os Modos - Simulado - 2 Nós em Linha.

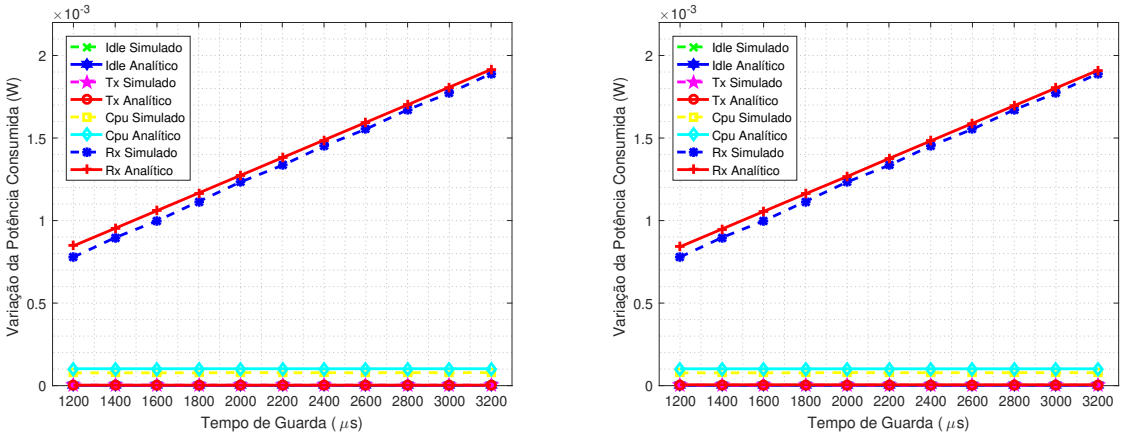
(a) Coordenador (b) Filho



Fonte: Autoria própria.

Figura 12 – Comparação Entre o Modelo Analítico e Simulado - 2 Nós em Linha.

(a) Coordenador (b) Filho

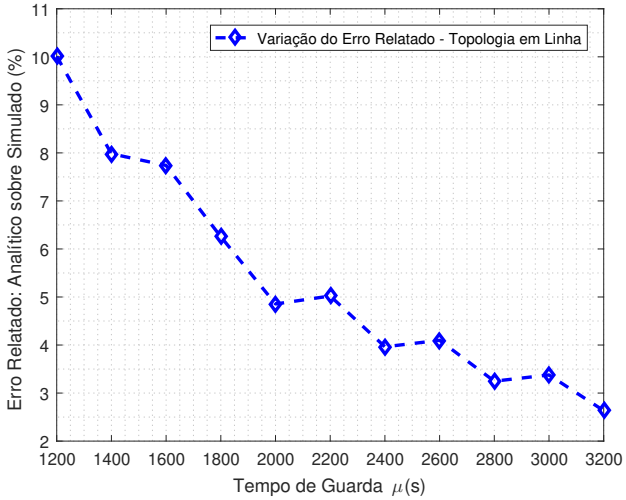


Fonte: Autoria própria.

Deste ponto em diante, as comparações serão feitas somente sobre a análise dos modos para o nó filho, tendo em vista que o nó coordenador apresenta um consumo semelhante. Os modos Idle e Tx apresentam valores muito pequenos, da ordem μW, o que torna impossível a análise visual gráfica destes modos. Além disto, os modos Tx e Idle não serão exibidos, sendo apenas computados seus consumos na análise de potência total dos nós.

A Figura 14 mostra a variação do consumo entre os modos pelo modelo analítico e simulado para os 8 nós filhos, na topologia em estrela. A análise das Figuras mostra o

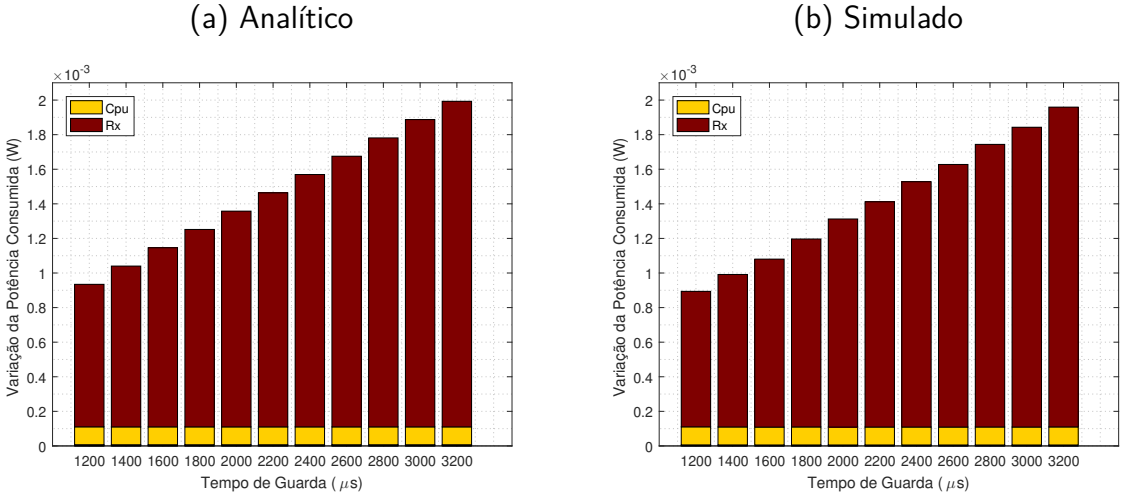
Figura 13 – Erro Percentual Relatado do Modelo Analítico sobre o Modelo Simulado - Modelo Analítico - Parâmetro Utilizado: Potência Total do Nó



Fonte: Autoria própria

modelo analítico com um consumo maior nos modos Rx e CPU com relação ao modelo simulado, comportamento análogo à topologia em linha.

Figura 14 – Consumo de Potência entre os Modos - Analítico e Simulado - 8 Nós Filhos para a Topologia em Estrela.



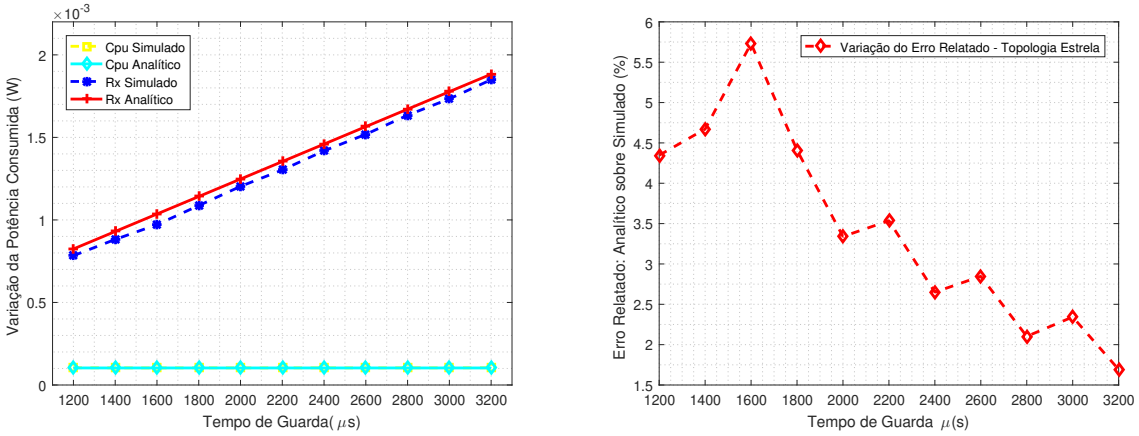
Fonte: Autoria própria.

A comparação entre a potência nos modos e o erro percentual do modelo analítico em relação ao modelo simulado são mostrados na Figura 15.

O modelo analítico apresenta um consumo de cerca de 6,54 % maior representando 63,6 μW a mais no consumo do modo Rx. A CPU apresenta um percentual de 1,16 % maior no modelo simulado, ou cerca 1,2 μW a mais no consumo. Os modos Idle e Tx não

Figura 15 – Comparação da Potência Consumida nos Modos - Modelo Analítico e Simulado. Erro Percentual entre os Modelos - Parâmetro Utilizado: Potência Total dos Nós.

(a) Variação da Potência Consumida (b) Erro Percentual



Fonte: Autoria própria.

apresentam aumento significativo do modelo analítico sobre o simulado, cerca de 1,2 μW. Em relação a potência total consumida pelo nó, o modelo analítico apresenta potência 5,73 % maior sobre o modelo analítico no pior caso para um tempo de guarda $t_g = 1600$ μs, o que representa um consumo de 65,95 μW maior.

4.1.1 Desempenho do Modelo Analítico de Consumo

Através dos resultados apresentados nas Figuras 12, 13 e 15, é possível perceber que o modelo proposto apresenta boa precisão quando comparado aos resultados práticos. O consumo calculado é cerca de 10 % maior que o obtido via simulação no Cooja (através da ferramentas Powertrace e Energest) na topologia em linha e 5,73 % maior na topologia em estrela, considerando sempre o pior caso e analisando a potência total do nó, quando somados todos os consumos nos modos. Como o modelo de consumo proposto sempre apresenta uma estimativa pessimista, isto significa que o tempo de vida do nó em geral é ligeiramente maior que o calculado.

4.1.2 Redução no Tempo de Guarda e no Consumo de Energia

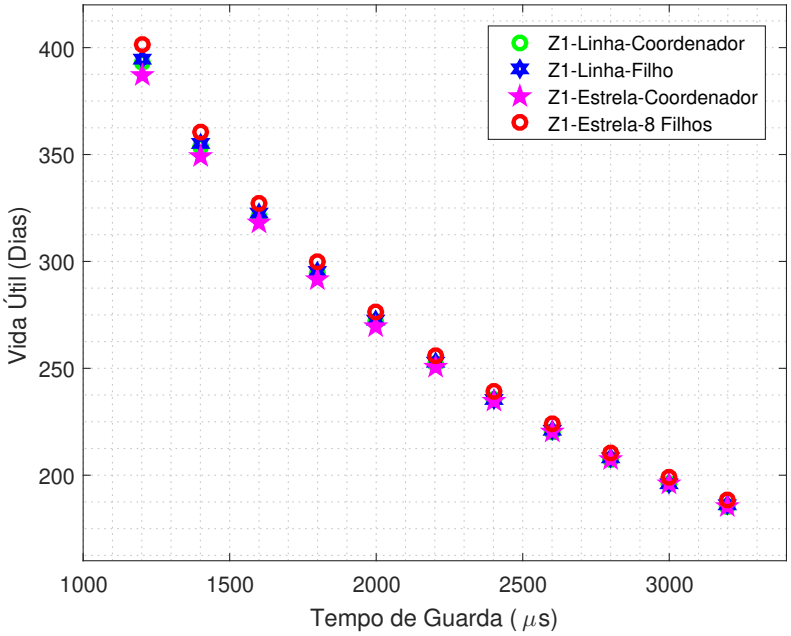
Tanto no modelo analítico como nos resultados simulados, percebe-se que a duração do tempo de guarda tem um impacto crítico no consumo de energia. Os resultados demonstraram que quanto menor o tempo de guarda, menor o consumo de energia dos

nós, devido ao fato de que o rádio permanece menos tempo ligado, à espera de pacotes.

A partir da configuração padrão do tempo de guarda no Contiki ($t_g = 2200$ μs), foram realizadas reduções com intervalos de 200 μs até um tempo de guarda ótimo de 1200 μs, já que este é o valor limite para que os nós não percam pacotes de dados e se mantenham sincronizados num intervalo de *Beacon* de 60 s. Esta redução pode representar uma economina no consumo de energia em mais de 50 % em ambas as topologias, linha e estrela.

Considerando-se que os nós estão equipados com uma bateria de 3000 mAh, esta redução também representa um aumento na vida útil de aproximadamente 44,8 % na topologia em linha, e 54,14 % na topologia em estrela quando é reduzido o tempo de guarda padrão ($t_g = 2200$ μs) para 1200 μs. Em termos quantitativos, o aumento é de 122 dias para a topologia linha e de 135 dias para a topologia estrela, conforme pode-se observar na Figura 16.

Figura 16 – Tempo de Vida da Bateria para os Nós Sensores - Plataforma Z1



Fonte: Autoria própria

4.2 PLATAFORMA CC2650 LAUNCHPAD

Nesta seção são apresentados os resultados calculados pelo modelo analítico proposto e os resultados da execução em uma plataforma real, a CC2650 Launchpad.

Tabela 16 – Resultados Obtidos com a Plataforma Z1 - Topologia Linha e Estrela - Comparação Numérica dos Modos entre o Modelo Analítico e Simulado, Redução no Consumo e Vida Útil Estimada da Bateria.

Topologia em Linha	**Modelo**	**Percentual** (%)	**Potência** (μW)
Modo Rx	Analítico	8,01 ↑	62,43
Modo CPU	Analítico	31,72 ↑	24,55
Potência Total do Nó	Analítico	10 ↑	89,5
Redução no t_g (μs)	**Alcance (μs)**	**Percentual** (%)	**Potência** (μW)
2200 ∼ 1200	1000	53 ↓	453,1
Bateria	**Vida Útil (dias)**	**Percentual** (%)	**Ganho (dias)**
3000 mAh	395	44,8 ↑	122
Topologia em Estrela	**Modelo**	**Percentual** (%)	**Potência** (μW)
Modo Rx	Analítico	6,54 ↑	63,6
Modo CPU	Simulado	1,16 ↑	1,2
Potência Total do Nó	Simulado	5,73 ↑	65,95
Redução no t_g (μs)	**Alcance (μs)**	**Percentual** (%)	**Potência** (μW)
2200 ∼ 1200	1000	56 ↓	500,5
Bateria	**Vida Útil (dias)**	**Percentual** (%)	**Ganho (dias)**
3000 mAh	401	54,14 ↑	135

Fonte: Autoria própria.

Todos os dados utilizados foram obtidos a partir dos *logs* gerados pelo Contiki. Também são apresentados os resultados quando a estratégia de Guard Beacon é inserida no processo de sincronização dos nós, diminuindo o tempo de guarda e reduzindo o consumo de energia. Os parâmetros utilizados são apresentados nas Tabelas 17 e 18.

Tabela 17 – Parâmetros de Simulação de Rede - Plataforma CC2650 Launchpad

Parâmetros da Topologia	**Valor**
Topologia	Linha
Número de Nós	2
Parâmetros de Simulação	**Valor**
Duração	10 min
Número de Saltos	1
Parâmetros do TSCH	**Valor**
Frequência de envio de EB	1 / min
Comprimento do Slotframe	7
Comprimento do Timeslot	15 ms
Tempo de Guarda	200 μs - 3200 μs
Intervalo de EBs	60, 80, 100 e 120 s

Fonte: Autoria própria.

Tabela 18 – Parâmetros de Hardware

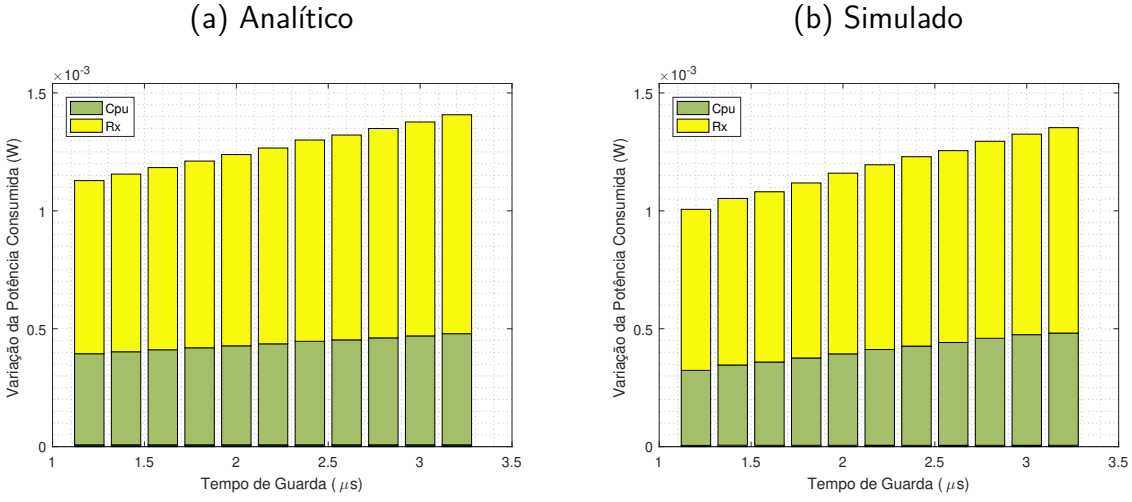
Parâmetros do *Hardware*	Value
Consumo em Tx	9,1 mA
Consumo em Rx	6,1 mA
Consumo em Sleep	1 μA
Consumo da CPU Ativa	2,93 mA
Consumo da CPU em *Low-Power*	1 μA

Fonte: (INSTRUMENTS, 2015).

Os resultados pertinentes são devidamente explicados no decorrer desta Seção, contudo, a Tabela 19 mostra um resumo dos resultados obtidos utilizando o modelo analítico e comparando com o modelo simulado. A análise dos modos bem como as reduções no consumo de energia e a vida útil da bateria utilizada também são mostradas.

A Figura 17 demonstra a diferença entre os valores reais e analíticos para um nó coordenador, obtida através dos valores utilizados para o modelo de consumo da platataforma Z1, para a plataforma CC2650 Launchpad. Entretanto, como descrito na Seção 3.2 a duração de cada estado é significativamente diferente para a plataforma CC2650 Launchpad. Isto prova que apesar do modelo de consumo ser o mesmo, os tempos para cada estado devem ser obtidos para cada plataforma de *hardware*, assim partiu-se para a investigação visando encontrar os tempos corretos para a plataforma CC2650.

Figura 17 – Resultado da Utilização dos Valores Aplicados no Modelo de Consumo da Plataforma Z1 para a Plataforma CC2650 - Nó Coordenador.

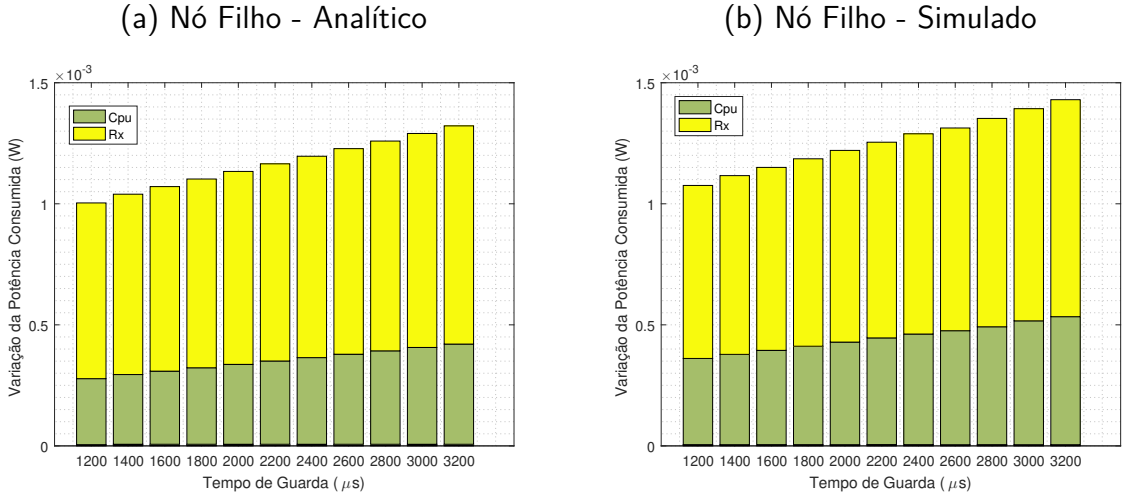


Fonte: Autoria própria.

4.2.1 Desempenho do Modelo Analítico

Os resultados obtidos demonstram que o modelo analítico de consumo se mostra eficaz também para predizer o comportamento energético dos nós da plataforma CC2650 Launchpad. Entretanto, conforme descrito na Seção 3.3.2, a CPU é ativada pela execução das demais *threads* do Contiki nos timeslots para os quais deveria estar em baixo consumo, o que causa um aumento na potência consumida da CPU em média de 27,72 % no modelo simulado em relação ao modelo analítico, contudo, no modo Rx, o modelo analítico apresenta um aumento de 0,55 %. Os valores percentuais apresentados representam em números cerca de 114,8 μW e 5 μW, conforme pode-se observar na Figura 18.

Figura 18 – Consumo de Potência entre os Modos - CC2650 Launchpad - Modelo Analítico e Simulado.



Fonte: Autoria própria.

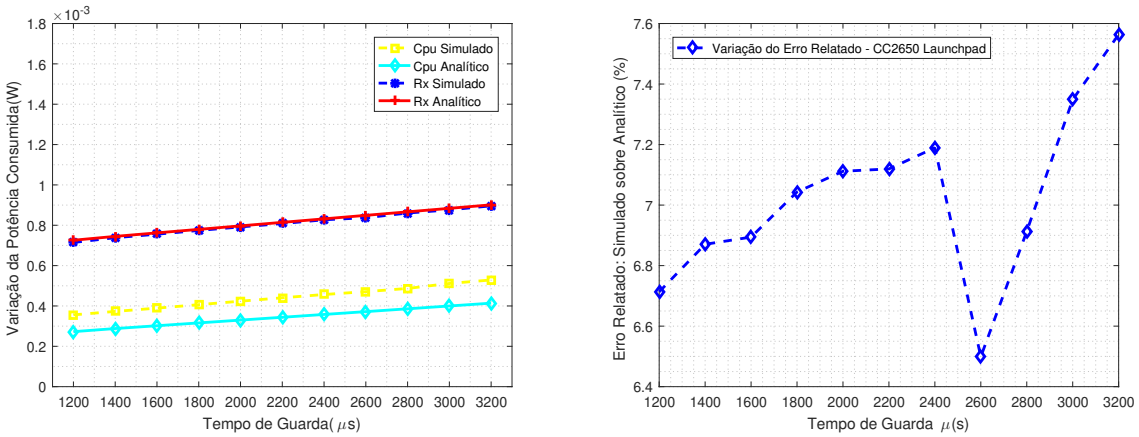
Esta constatação gera uma diferença na potência total do nó entre os modelos, de cerca de 7,56 % maior no modelo simulado. Por fim, a Figura 19 mostra a comparação entre os modos e o erro percentual entre o modelo analítico e simulado, através da potência total do nó.

4.2.2 Redução no Tempo de Guarda e no Consumo de Energia

Seguindo a mesma abordagem feita na análise do consumo para a plataforma Z1, novamente foi possível verificar o impacto do tempo de guarda na redução do consumo energético do nó. Reduzindo o tempo de guarda em intervalos de 200 μs até o tempo de guarda ótimo de 1200 μs, pode-se observar uma economia de energia de 18,18 % em relação à configuração padrão do Contiki com tempo de guarda de 2200 μs.

Figura 19 – Comparação da Potência Consumida nos Modos - Modelo Analítico e Simulado. Erro Percentual entre os Modelos - Parâmetro Utilizado: Potência Total dos Nós - CC2650 Launchpad.

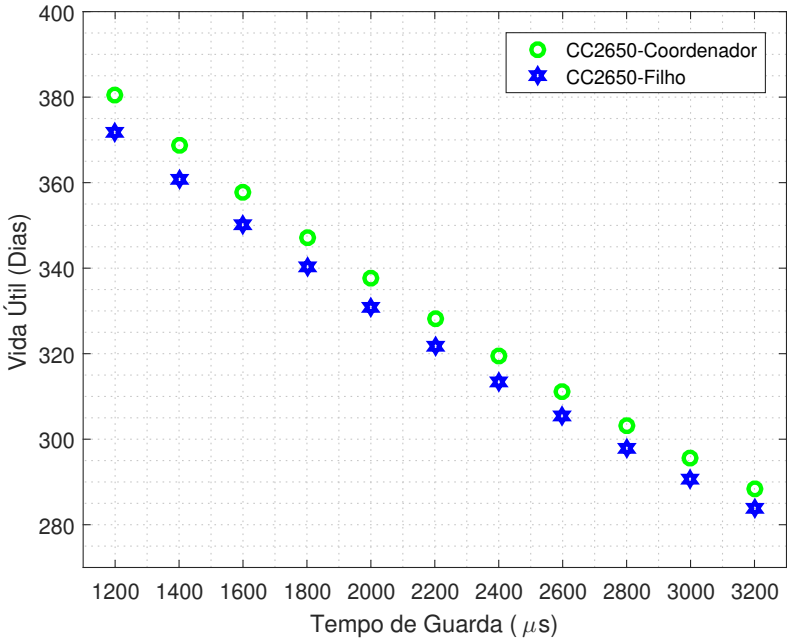
(a) Variação da Potência Consumida (b) Erro Percentual



Fonte: Autoria própria.

Referente à vida útil do nó, considerando-se uma bateria de 3000 mAh e esta redução no tempo de guarda, verificou-se um aumento de aproximadamente 15,51 %. Em termos quantitativos, o aumento da vida útil é de cerca de 50 dias, como demonstrado na Figura 20.

Figura 20 – Tempo de Vida da Bateria do Nó Sensor - CC2650 Launchpad



Fonte: Autoria própria

Tabela 19 – Resultados Obtidos na Plataforma CC2650 Launchpad - Comparação Numérica dos Modos entre o Modelo Analítico e Simulado, Redução no Consumo e Vida Útil Estimada da Bateria.

Configuração Padrão	**Modelo**	**Percentual** (%)	**Potência** (μW)
Modo Rx	Analítico	0,55 ↑	5
Modo CPU	Simulado	27,72 ↑	114,8
Potência Total do Nó	Simulado	7,56 ↑	98,2
Redução no t_g (μs)	**Alcance (μs)**	**Percentual** (%)	**Potência** (μW)
2200 ∼ 1200	1000	18,18 ↓	195,6
Bateria	**Vida Útil (dias)**	**Percentual** (%)	**Ganho (dias)**
3000 mAh	373	15,51 ↑	50

Fonte: Autoria própria.

4.2.3 Análise do Consumo Utilizando a Estratégia do Guard Beacon

Conforme descrito na Seção 4.2, também são apresentados os principais resultados obtidos na utilização da estratégia do *Guard Beacon*, de acordo com a Tabela 20. Os dados obtidos com a configuração padrão são replicados para facilitar a comparação.

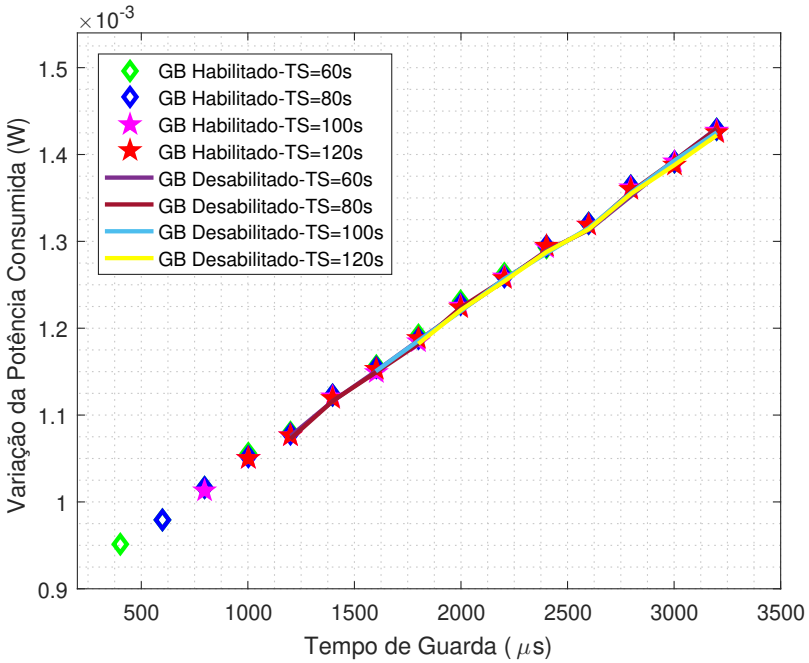
Com o objetivo de avaliar a eficiência da estratégia do *Guard Beacon*, foram simulados dois cenários, primeiro com a configuração padrão do Contiki TSCH, variando o período de EB de 60 segundos até alcançar 120 segundos, com variação de 20 segundos. Ou seja, usando 1 EB de 37 bytes de para sincronização. Segundo, utilizando os mesmos períodos de sincronização, com a estratégia do *Guard Beacon* habilitada. Enviando então 3 GBs de 1 byte de comprimento cada. Os resultados são mostrados na Figura 21.

É possível perceber como a técnica do *Guard Beacon* pode maximizar a eficiência energética sem prejudicar a sincronização dos nós. Para diferentes intervalos de sincronização (*IS*) existem valores mínimos do tempo de guarda, e o menor consumo é obtido através da técnica do *Guard Beacon*, ao se considerar um intervalo de sincronização de 60 s e tempo de guarda de apenas 400 μs.

A Figura 22 apresenta a economia de energia alcançada para o intervalo de sincronização IS = 120 s, que pode chegar a 13,1 % quando o tempo de guarda é reduzido de 1800 μs para 1000 μs, ambos os tempos de guarda mínimos possíveis para este intervalo de sincronização e com *Guard Beacon* desabilitado e habilitado, respectivamente.

Contudo, o cenário mais promissor foi alcançado com IS = 60 s conforme pode ser observado na Figura 23, onde pode se alcançar uma economia de energia percentual de 13,5%, quando o tempo de guarda é reduzido de 1200 μs para 400 μs, ambos os tempos

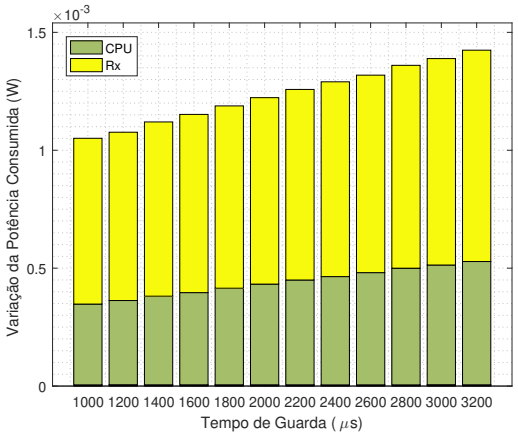
Figura 21 – Variação da Potência Consumida Aplicando a Estratégia do Guard Beacon.



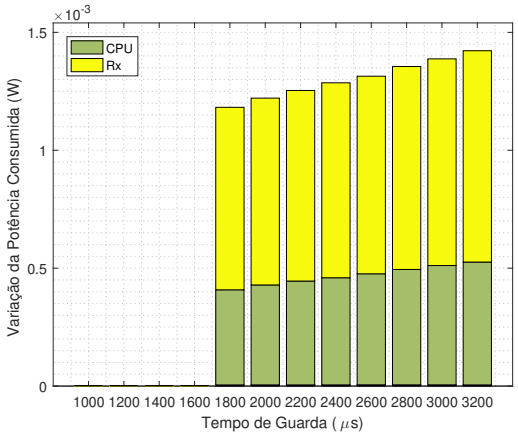
Fonte: Autoria própria.

Figura 22 – Variação da Potência Consumida Aplicando a Estratégia do Guard Beacon - GB Habilitado e GB Desabilitado - com Intervalo de Sincronização de 120s.

(a) GB Habilitado - Nó Filho



(b) GB Desabilitado - Nó Filho



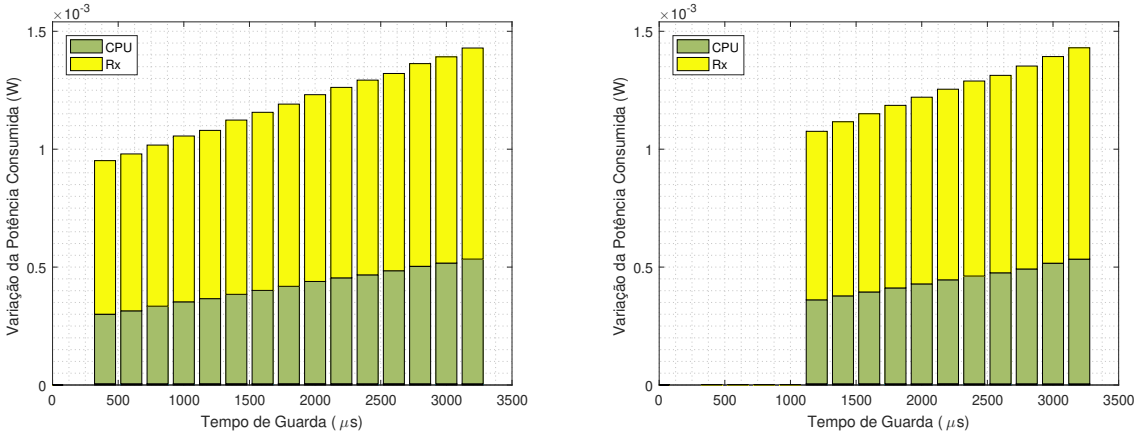
Fonte: Autoria própria.

de guarda mínimos possíveis para este intervalo de sincronização e com Guard Beacon desabilitado e habilitado, respectivamente.

Para comprovar novamente a precisão do modelo analítico, a estratégia do *Guard Beacon* foi habilitada com um intervalo de sincronização de 60 s, e os resultados de

Figura 23 – Variação da Potência Consumida Aplicando a Estratégia do Guard Beacon - GB Habilitado e GB Desabilitado - com Intervalo de Sincronização de 60s.

(a) GB Habilitado - Nó Filho (b) GB Desabilitado - Nó Filho

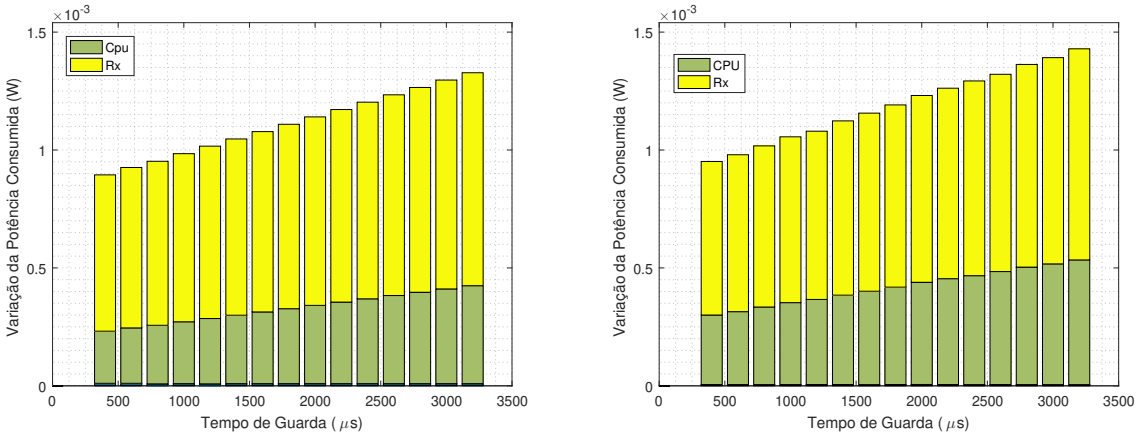


Fonte: Autoria própria.

execução na prática foram comparados com o os obtidos analiticamente. A Figura 24 apresenta a variação de potência para os modelos e a Figura 25 mostra o erro percentual entre os modelos.

Figura 24 – Potência Consumida entre os Modos - Guard Beacon Habilitado - Comparação entre Modelo Analítico e Simulado.

(a) Nó Filho - Analítico (b) Nó Filho - Simulado

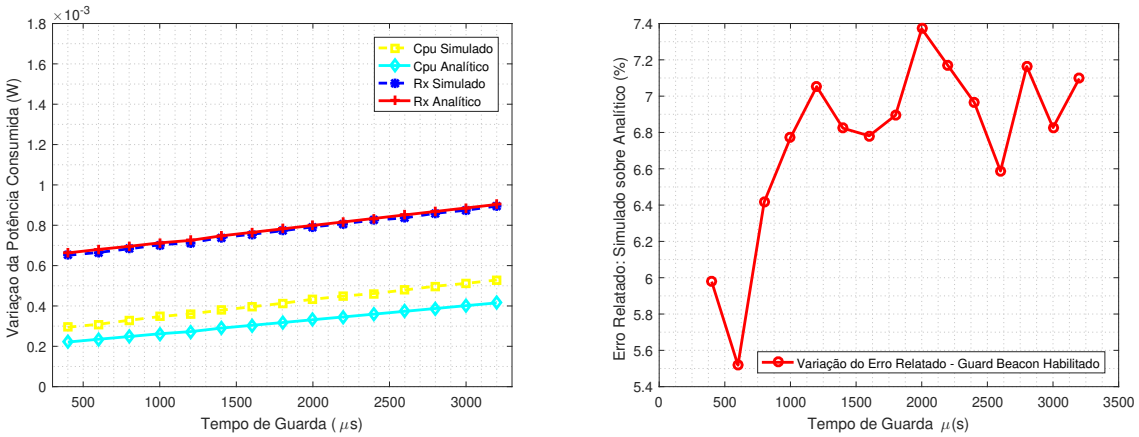


Fonte: Autoria própria.

Na Figura 25 é realizada uma comparação entre o modelo analítico e os resultados práticos. Da análise, é possível perceber que o modelo analítico proposto apresenta uma potência total consumida pelo nó sensor 7,4 % menor em relação aos dados reais, o que representa em números 84,4 μW, observado o pior caso, com $t_g = 2000$ μs. O modo Rx

Figura 25 – Comparação da Potência Consumida nos Modos - Modelo Analítico e Simulado. Erro Percentual entre os Modelos - Parâmetro Utilizado: Potência Total dos Nós Utilizando a Estratégia do Guard Beacon.

(a) Variação da Potência Consumida (b) Erro Percentual



Fonte: Autoria própria.

apresenta uma potência 0,88 % maior no modelo analítico, contudo, a CPU do modelo simulado apresenta um aumento de 30,3 %, devido aos mesmos fatores apresentados na Seção 3.3.2 e na Seção 4.2.1. Estes valores representam em números cerca de 7 μW para Rx e 100 μW para a CPU.

Com a estratégia do *Guard Beacon* ativada, ao considerarmos uma bateria de 3000 mAh, o aumento na vida útil é de 12,71 %, representando um aumento de 47 dias na vida útil da bateria, conforme apresenta a Figura 26.

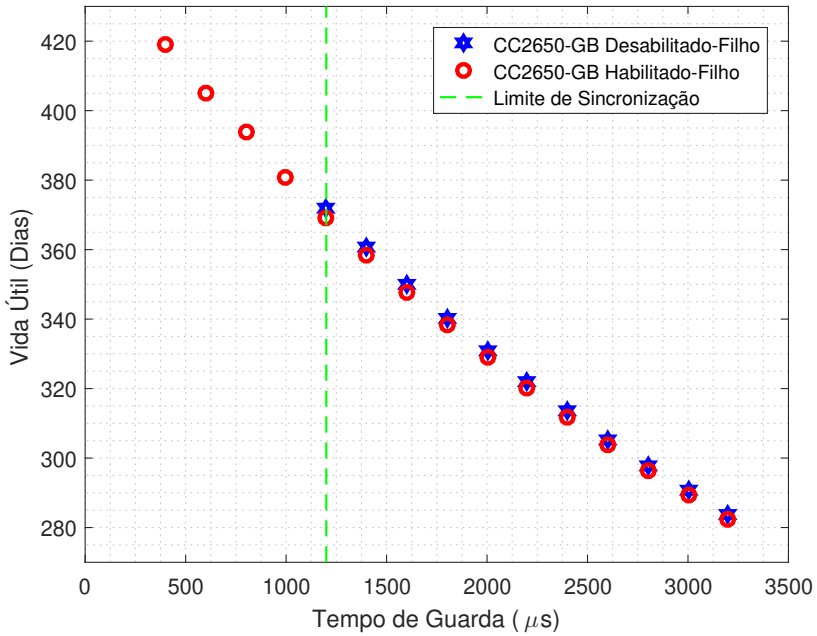
4.2.4 Impacto das Alterações de Firmware na Potência Consumida

A Figura 27 apresenta a comparação entre a estratégia do Guard Beacon e a configuração padrão do Contiki TSCH, utilizando o modelo analítico e simulado.

Os resultados demonstraram que o impacto das alterações feitas no *firmware* do Contiki TSCH foram insignificantes no aumento da potência total consumida quando comparadas a economia de energia proporcionada na redução do tempo de guarda.

A análise dos modos demonstra que o consumo é muito parecido quando o *Guard Beacon* está habilitado e quando desabilitado, com variação de 1 a 3 μW de potência entre os modos. A grande vantagem no uso da técnica do *Guard Beacon* pode ser observada na redução do tempo de guarda, com intervalo de sincronização em IS = 60 s, a redução de 2200 μs para 400 μs pode gerar uma redução no consumo de energia do nó em cerca de

Figura 26 – Tempo de Vida da Bateria - Plataforma CC2650



Fonte: Autoria própria

Tabela 20 – Resultados Obtidos com a Configuração Padrão e com a Estratégia do *Guard Beacon* Habilitada - CC2650 Launchpad - Comparação Numérica dos Modos entre o Modelo Analítico e Simulado, Redução no Consumo e Vida Útil Estimada da Bateria.

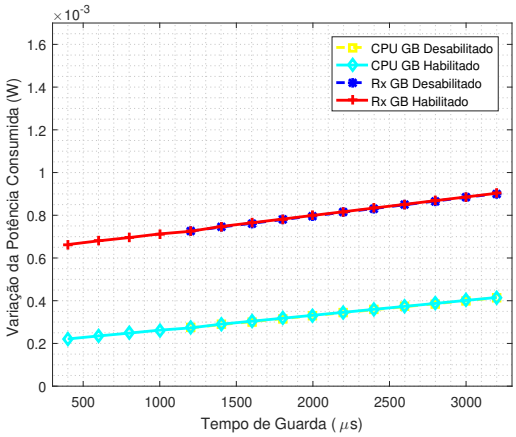
Configuração Padrão	**Modelo**	**Percentual** (%)	**Potência** (μW)
Modo Rx	Analítico	0,55 ↑	5
Modo CPU	Simulado	27,72 ↑	114,8
Potência Total do Nó	Simulado	7,56 ↑	98,2
Redução no t_g (μs)	**Alcance (μs)**	**Percentual** (%)	**Potência** (μW)
2200 ∼ 1200	1000	18,18 ↓	195,6
Bateria	**Vida Útil (dias)**	**Percentual** (%)	**Ganho (dias)**
3000 mAh	373	15,51 ↑	50
***Guard Beacon* Habilitado**	**Modelo**	**Percentual** (%)	**Potência** (μW)
Modo Rx	Analítico	0,88 ↑	7
Modo CPU	Simulado	30,3 ↑	100
Potência Total do Nó	Simulado	7,4 ↑	84,4
Redução no t_g (μs)	**Alcance (μs)**	**Percentual** (%)	**Potência** (μW)
1200 ∼ 400	800	13,5 ↓	128,2
Bateria	**Vida Útil (dias)**	**Percentual** (%)	**Ganho (dias)**
3000 mAh	419	12,71 ↑	47

Fonte: Autoria própria.

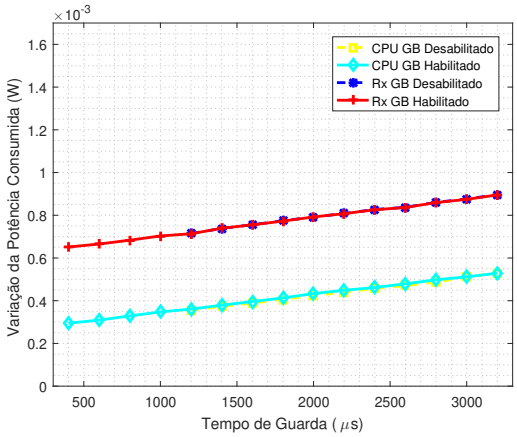
31,68 %, se somadas a reduções apresentadas na Seção 4.2.2 e na Seção 4.2.3.

Figura 27 – Comparação entre a Estratégia do Guard Beacon e a Configuração Padrão.

(a) Nó Filho - Analítico



(b) Nó Filho - Simulado



Fonte: Autoria própria.

5 CONCLUSÃO E COMENTÁRIOS FINAIS

Neste trabalho apresentou-se um modelo analítico de consumo de energia para redes de sensores sem fio operando com sistema operacional Contiki e utilizando TSCH. O modelo analítico foi avaliado através da comparação do consumo calculado com os resultados de simulação e de execução prática, simultaneamente com a implementação da Estratégia do *Guard Beacon*, visando a diminuição do tempo de guarda, e por consequência do consumo de energia.

Os resultados mostraram que a Estratégia do *Guard Beacon* é mais eficiente energeticamente do que o envio de um único *Beacon* para diferentes intervalos de sincronização. Em ambos os casos, os resultados mostram que o comprimento do tempo de guarda está fortemente correlacionado com o consumo de energia. A utilização de parâmetros ótimos maximiza a eficiência energética possibilitando a redução do tempo de guarda sem prejuízo à sincronização dos nós.

Com relação a trabalhos futuros, considera-se aplicar a estratégia do *Guard Beacon* no processo de *rendezvous*, com o objetivo de aprimorar o mecanismo pelo qual os nós se associam ao coordenador de uma rede TSCH, reduzindo o tempo em que os nós precisam manter seus receptores ligados até ouvir o pacote de associação, o que permitirá diminuir o desperdício de energia ocorrido neste processo.

REFERÊNCIAS

802.15.4, I. Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (wpans). *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, p. 1–320, Sep. 2006.

802.15.4E, I. Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans) amendment 1: Mac sublayer. *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, p. 1–225, April 2012.

ANASTASI, G.; CONTI, M.; FRANCESCO, M. D.; PASSARELLA, A. Energy conservation in wireless sensor networks: A survey. **Ad Hoc Networks**, v. 7, p. 537–568, 2009.

AZOIDOU, E.; PANG, Z.; LIU, Y.; LAN, D.; BAG, G.; GONG, S. Battery lifetime modeling and validation of wireless building automation devices in thread. **IEEE Transactions on Industrial Informatics**, v. 14, n. 7, p. 2869–2880, July 2018. ISSN 1551-3203.

BACCELLI, E.; HAHM, O.; GUNES, M.; WAHLISCH, M.; SCHMIDT, T. C. RIOT OS: Towards an OS for the Internet of Things. In: **2013 IEEE Conf. on Comput. Commun. Workshops (INFOCOM WKSHPS)**. 2013. p. 79–80.

BARRY, R. **The *FreeRTOS*^{TM} Kernel**. 2003. Disponível em: <http://www.freertos.org/>.

CHANG, T.; WANG, Q. Adaptive compensation for time-slotted synchronization in wireless sensor network. **International Journal of Distributed Sensor Networks**, v. 10, n. 4, p. 540397, 2014. Disponível em: <https://doi.org/10.1155/2014/540397>.

CHEN, Y.; QIN, F.; YI, W. Guard beacon: An energy-efficient beacon strategy for time synchronization in wireless sensor networks. **IEEE Communications Letters**, v. 18, n. 6, p. 987–990, June 2014. ISSN 1089-7798.

DUNKELS, A.; ERIKSSON, J.; FINNE, N.; TSIFTES, N. **Powertrace: Network-level Power Profiling for Low-power Wireless Networks**. Kista, Sweden, 2011.

DUNKELS, A.; GRONVALL, B.; VOIGT, T. Contiki - a lightweight and flexible operating system for tiny networked sensors. In: . 2004. p. 455–462. ISSN 0742-1303.

DUNKELS, A. et al. **Contiki, the open source OS for the Internet of Things**. Jul. 2019. Disponível em: <https://github.com/contiki-os/contiki>.

DUQUENNOY, S.; ELSTS, A.; NAHAS, B. A.; OIKONOMOU, G. Tsch and 6tisch for contiki: Challenges, design and evaluation. In: **2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)**. 2017. p. 11–18. ISSN 2325-2944.

DUY, T. P.; KIM, Y. An efficient joining scheme in ieee 802.15.4e. In: **2015 International Conference on Information and Communication Technology Convergence (ICTC)**. 2015. p. 226–229.

ERICSSON. **Ericsson Mobility Report**. jun. 2018. Disponível em: <https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf>.

GUGLIELMO, D. D.; BRIENZA, S.; ANASTASI, G. Ieee 802.15.4e: A survey. **Computer Communications**, v. 88, p. 1–24, 2016.

GUGLIELMO, D. D.; BRIENZA, S.; ANASTASI, G. A model-based beacon scheduling algorithm for ieee 802.15.4e tsch networks. In: **2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)**. 2016. p. 1–9.

GUGLIELMO, D. D.; NAHAS, B. A.; DUQUENNOY, S.; VOIGT, T.; ANASTASI, G. Analysis and experimental evaluation of ieee 802.15.4e tsch csma-ca algorithm. **IEEE Transactions on Vehicular Technology**, v. 66, n. 2, p. 1573–1588, Feb 2017. ISSN 0018-9545.

GUGLIELMO, D. D.; SEGHETTI, A.; ANASTASI, G.; CONTI, M. A performance analysis of the network formation process in ieee 802.15.4e tsch wireless sensor/actuator networks. In: **2014 IEEE Symposium on Computers and Communications (ISCC)**. 2014. p. 1–6. ISSN 1530-1346.

HEJAZI, H.; RAJAB, H.; CINKLER, T.; LENGYEL, L. Survey of platforms for massive iot. In: **2018 IEEE International Conference on Future IoT Technologies (Future IoT)**. 2018. p. 1–8.

INSTRUMENTS, T. **CC2650 *SimpleLink*^{TM} Multistandard Wireless MCU**. fev. 2015. Disponível em: <http://www.ti.com/lit/ds/symlink/cc2650.pdf>.

JEONG, W.; LEE, J. Performance evaluation of ieee 802.15.4e dsme mac protocol for wireless sensor networks. In: **2012 The First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT)**. 2012. p. 7–12.

KIM, K. T.; KIM, H.; PARK, H.; KIM, S. An industrial iot mac protocol based on ieee 802.15.4e tsch for a large-scale network. In: **2017 19th International Conference on Advanced Communication Technology (ICACT)**. 2017. p. 721–724.

KIM, K. T.; KIM, J. An energy efficient real-time mac protocol. In: **2018 International Conference on Information and Communication Technology Convergence (ICTC)**. 2018. p. 1180–1184. ISSN 2162-1233.

KIM, M.; LEE, J.; KIM, Y.; SONG, Y. H. An analysis of energy consumption under various memory mappings for fram-based iot devices. In: **2018 IEEE 4th World Forum on Internet of Things (WF-IoT)**. 2018. p. 574–579.

KURUNATHAN, H.; SEVERINO, R.; KOUBAA, A.; TOVAR, E. Worst-case bound analysis for the time-critical mac behaviors of ieee 802.15.4e. In: **2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)**. 2017. p. 1–9.

LEVIS, P. et al. TinyOS: An Operating System for Sensor Networks. In: WEBER, W.; RABAEY, J. M.; AARTS, E. (Ed.). **Ambient Intelligence**. Berlin, Heidelberg: Springer, 2005. p. 115–148. ISBN 978-3-540-27139-0.

LI, P.; VERMEULEN, T.; LIY, H.; POLLIN, S. An adaptive channel selection scheme for reliable tsch-based communication. In: **2015 International Symposium on Wireless Communication Systems (ISWCS)**. 2015. p. 511–515. ISSN 2154-0225.

NADAS, J. P. B.; SOUZA, R. D.; PELLENZ, M. E.; BRANTE, G.; BRAGA, S. M. Energy efficient beacon based synchronization for alarm driven wireless sensor networks. **IEEE Signal Processing Letters**, v. 23, n. 3, p. 336–340, March 2016. ISSN 1070-9908.

OJO, M.; ADAMI, D.; GIORDANO, S. Performance evaluation of energy saving mac protocols in wsn operating systems. In: **2016 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)**. 2016. p. 1–7.

PAPADOPOULOS, G. Z.; MAVROMATIS, A.; FAFOUTIS, X.; MONTAVONT, N.; PIECHOCKI, R.; TRYFONAS, T.; OIKONOMOU, G. Guard time optimisation and adaptation for energy efficient multi-hop tsch networks. In: **2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)**. 2016. p. 301–306.

PROJECT, T. Z. **An RTOS for IoT**. 2017. Disponível em: <https://www.zephyrproject.org/>.

RAYEL, O. K.; SORDI, M. A. de. **Contiki's TSCH Guard Beacon Implementation**. Jul. 2019. Disponível em: <https://doi.org/10.5281/zenodo.3268707>.

SABRI, C.; KRIAA, L.; AZZOUZ, S. L. Comparison of iot constrained devices operating systems: A survey. In: **2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)**. 2017. p. 369–375. ISSN 2161-5330.

STANISLOWSKI, D.; VILAJOSANA, X.; WANG, Q.; WATTEYNE, T.; PISTER, K. S. J. Adaptive synchronization in ieee802.15.4e networks. **IEEE Transactions on Industrial Informatics**, v. 10, n. 1, p. 795–802, Feb 2014. ISSN 1551-3203.

TEXAS INSTRUMENTS. **CC2420 (NRND) Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee^{TM} Ready RF Transceiver**. mar. 2013. Disponível em: <http://www.ti.com/product/CC2420/technicaldocuments>.

THUBERT, P.; WATTEYNE, T.; PALATTELLA, M. R.; VILAJOSANA, X.; WANG, Q. Ietf 6tsch: Combining ipv6 connectivity with industrial performance. In: **2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing**. 2013. p. 541–546.

VILAJOSANA, X.; PEIRO, P. T.; GALEGO, F. V.; ZARATE, J. A.; ALONSO, L. Standardized low-power wireless communication technologies for distributed sensing applications. **Sensors**, v. 14, n. 2, p. 2663–2682, 2014.

VILAJOSANA, X.; WANG, Q.; CHRAIM, F.; WATTEYNE, T.; CHANG, T.; PISTER, K. S. J. A realistic energy consumption model for tsch networks. **IEEE Sensors Journal**, v. 14, n. 2, p. 482–489, Feb 2014. ISSN 1530-437X.

WATTEYNE, T. **State Machine**. 2012. Access date: 1 jul. 2017. Disponível em: <https://openwsn.atlassian.net/wiki/spaces/OW/pages/688251/State+Machine>.

WATTEYNE, T.; VILAJOSANA, X.; KERKEZ, B.; CHRAIM, F.; WEEKLY, K.; WANG, Q.; GLASER, S.; PISTER, K. OpenWSN: a standards-based low-power wireless development environment. **Trans. Emerging Tel. Tech.**, v. 23, n. 5, p. 480–493, Aug. 2012.

YOUSSEF, M. F.; ELSAYED, K. M. F.; ZAHRAN, A. H. Contiki-AMAC – the enhanced adaptive radio duty cycling protocol: Proposal and analysis. In: **2016 Int. Conference on Select. Topics in Mobile Wireless Netw. (MoWNeT)**. 2016. p. 1–6.