UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE ENGENHARIA ELÉTRICA

Angelo Antônio Langner - GRR20160052
Pedro Henrique Lode Gonçalves - GRR20204948
Ênfase em Eletrônica e Telecomunicações

OTIMIZAÇÃO DO PROTOCOLO RPL EM UM SISTEMA DE MONITORAMENTO DO SOLO

CURITIBA
2024

Angelo Antônio Langner
Pedro Henrique Lode Gonçalves

OTIMIZAÇÃO DO PROTOCOLO RPL EM UM SISTEMA DE MONITORAMENTO DO SOLO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica da Universidade Federal do Paraná, como exigência parcial para obtenção do grau de Bacharel em Engenharia Elétrica com ênfase em Eletrônica e Telecomunicações.

Orientador: Prof. Dr. Carlos Marcelo Pedroso

CURITIBA
2024

Aos nossos pais, pelos aconselhamentos e orientações e por sempre acreditarem em nosso potencial, nos apoiando em todos os momentos, dando-nos assim, um forte incentivo para sempre progredirmos. A vocês a nossa eterna gratidão.

AGRADECIMENTOS

Agradecemos, especialmente, ao Prof. Dr. Carlos Marcelo Pedroso, nosso orientador, pela oportunidade, confiança, dedicação e apoio durante todo o processo de construção deste TCC.

Aos nossos colegas de turma, que com sua cooperação, tornaram o curso muito mais interessante.

A todas as pessoas que de alguma forma fizeram parte dessa jornada, nós agradecemos com um forte abraço.

RESUMO

O cultivo de soja é crucial para a economia global e, especialmente, para a brasileira, dada sua significativa contribuição para a produção de alimentos. A safra 22/23 resultou em 154,6 milhões de toneladas, conforme a Companhia Nacional de Abastecimento (CONAB). O setor do agronegócio, segundo a Confederação da Agricultura e Pecuária do Brasil (CNA), representou 24,1% do Produto Interno Bruto (PIB) brasileiro em 2023, com um valor estimado em R$ 2,62 trilhões. No entanto, os produtores enfrentam desafios com a crescente demanda, variabilidade climática e a competitividade global, necessitando de tecnologias inovadoras para manter a eficiência e a sustentabilidade. Este trabalho visa implementar um sistema de rede de sensores para monitoramento do cultivo da soja, utilizando o sistema operacional Contiki, o emulador Cooja, rádios conforme o padrão IEEE 802.15.4 e um coletor de energia eletromagnética. O objetivo é avaliar se os sensores podem operar exclusivamente com energia captada de ondas eletromagnéticas no ambiente, reduzindo a dependência de baterias e infraestrutura elétrica. O projeto envolve a definição de requisitos para monitoramento da qualidade do solo, desenvolvimento da rede de sensores e configuração do protocolo RPL. A simulação do sistema no Cooja visa coletar dados de consumo de energia, atraso, perda e *jitter* na comunicação, proporcionando uma análise completa da viabilidade e eficiência do sistema proposto.

Palavras-chave: Agricultura de precisão; Redes de sensores; Monitoramento do solo; Contiki; Cooja; RPL.

ABSTRACT

Soybean cultivation is vital for the global economy and particularly for Brazil, given its substantial contribution to food production. The 22/23 crop yielded 154.6 million tons according to the National Supply Company (CONAB). In 2023, the agribusiness sector, as reported by the Confederation of Agriculture and Livestock of Brazil (CNA), accounted for 24.1% of Brazil's GDP, estimated at R$ 2.62 trillion. However, producers face challenges such as increasing demand, climatic variability, and global competitiveness, necessitating innovative technologies to maintain efficiency and sustainability. This work aims to implement a sensor network system for soybean monitoring, utilizing the Contiki operating system, Cooja emulator, IEEE 802.15.4 radios, and an electromagnetic energy harvester. The goal is to assess whether the sensors can operate solely with energy harvested from ambient electromagnetic waves, reducing reliance on batteries and electrical infrastructure. The project involves defining requirements for soil quality monitoring, developing the sensor network, and configuring the RPL protocol. System simulation in Cooja will gather data on energy consumption, delay, loss, and jitter in communication, providing a comprehensive analysis of the proposed system's viability and efficiency.

Keywords: Precision agriculture; Sensor networks; Soil monitoring; Contiki; Cooja; RPL.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ACK	Mensagem Recebida com Sucesso
ANATEL	Agência Nacional de Telecomunicações
BLUETOOTH	Protocolo de Comunicação entre Dispositivos Eletrônicos
CC2420	Transmissor-Receptor de Rádio Frequência
CE	Condutividade Elétrica
CEa	Condutividade Elétrica Aparente
CEPEA	Centro de Estudos Avançados em Economia Aplicada
CNA	Confederação da Agricultura e Pecuária do Brasil
CoAP	Protocolo de Aplicação Restrita
CONAB	Companhia Nacional de Abastecimento
CONTIKI	Sistema Operacional de Código Aberto
Cooja	Sistema Operacional Contiki Java
CPU	Unidade Central de Processamento
CSMA	Acesso Múltiplo com Verificação de Portadora
CSMA-CA	Acesso Múltiplo com Verificação de Portadora e Anulação de Colisão
ERBs	Estação Rádio Base
DAO	Objeto de Anúncio de Destino
DIO	Objeto de Informação
DIS	Solicitação de Informação

DNS	Sistema de Nomes de Domínio
DODAG	Destination-Oriented Directed Acyclic Graph
DSS	Compartilhamento Dinâmico de Espectro
DSSS	Espectro de Espalhamento de Sequência Direta
DTLS	Segurança da Camada de Transporte de Datagrama
EMI	Interferência Eletromagnética
etimer	Temporizador de Eventos
ETX	Contador de Transmissão Esperado
FET	Transistor de Efeito de Campo
FHSS	Espectro de Espalhamento de Salto de Frequência
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
IEEE 802	Protocolos para a Comunicação de Computadores em Redes Locais sem Fio
IoT	Internet das Coisas
IPv6	Protocolo de Internet Versão 6
LC	Tipo de Conector Óptico de Fibra Óptica
LWM2M	Simplificar e Padronizar o Gerenciamento de Dispositivos na IoT
MRHOF	Classificação Mínima com Função Objetivo de Histerese
OF	Função Objetivo
OF0	Função Objetivo 0
PAWC	Capacidade de Água Disponível para as Plantas

PDR	Taxa de Entrega de Pacotes
pH	Potencial Hidrogeniônico
PIB	Produto Interno Bruto
QoS	Qualidade de Serviço
RAM	Memória de Acesso Aleatório
RE	Resistividade Elétrica
RF	Radiofrequência
RFC	Pedido de Comentários
RPL	Protocolo de Roteamento para Redes de Baixa Capacidade e com Perdas
RSSI	Indicação de Intensidade do Sinal Recebido
RX	Taxa de Recepção
SO	Sistema Operacional
TCP	Protocolo de Controle de Transmissão
TSCH	Salto de Canal com Intervalo de Tempo
TX	Taxa de Transmissão
UDP	Protocolo de Datagrama do Usuário
VHF	Frequência Muito Alta
WEBSOCKETS	Protocolo de comunicação Bidirecional
WPAN	Rede Pessoal de Área Local sem Fio

SUMÁRIO

1 INTRODUÇÃO

O cultivo da soja é uma atividade de extrema importância para a economia global e ainda de maior importância para a economia brasileira, fornecendo alimentos essenciais. Nesse viés, segundo o levantamento de 02/2024 feito pela Companhia Nacional de Abastecimento (CONAB), foram produzidas 154.609,5 mil toneladas na safra 22/23 [1]. Somado a isso, pesquisadores do Centro de Estudos Avançados em Economia Aplicada (CEPEA) e da Confederação da Agricultura e Pecuária do Brasil (CNA) indicam, em 21/12/2023, que o PIB do setor do agronegócio pode alcançar R$ 2,62 trilhões em 2023, o que corresponderia a 24,1% do PIB do País [8]. Dessa maneira, com a crescente demanda por produtos derivados da soja e as constantes mudanças nas condições climáticas e a competitividade do mercado global, os agricultores brasileiros enfrentam diversas dificuldades para garantir uma produção eficiente, sustentável e minimizar os riscos e perdas nas safras. Nesse contexto, a adoção de novas tecnologias e inovações no plantio de soja são fundamentais para manter-se competitivo e eficaz no mercado do agronegócio. Assim, um desses avanços tecnológicos promissores é a implementação de sistemas de monitoramento autônomo na lavoura, que podem oferecer dados do solo, tais como: temperatura e umidade, entre outros. Permitindo aos agricultores analisar e tomar decisões para garantir a máxima produção e eficiência da safra [2][3].

Contudo, para a implementação desse tipo de sistema, também é necessário uma rede elétrica conjunta para a alimentação energética do sistema, o que implica em um grande investimento em infraestrutura e cabos, com risco de danificar o solo durante a construção da rede. Já o uso de baterias requer grande capacidade de carga para que o sistema tenha autonomia de operação por longo período antes da necessidade de recarga ou substituição das mesmas, implicando em alto investimento e na necessidade frequente de manutenção. Dessa maneira, as tecnologias que serão utilizadas para contornar estas dificuldades são: o sistema operacional Contiki com o emulador Cooja; rádios no padrão IEEE 802.15.4 e um coletor e armazenador de energia das ondas eletromagnéticas no ambiente.

1.1 OBJETIVOS

Com base no que foi mencionado, este trabalho visa desenvolver em simulações a implementação de um sistema de rede de sensores em uma lavoura de soja, realizando modificações na configuração do protocolo RPL para avaliar a qual distância a fonte de emissão eletromagnética deve estar para que a energia consumida pelos dispositivos seja exclusivamente carregada pelas ondas eletromagnéticas disponíveis no ambiente.

1.1.2 OBJETIVOS ESPECÍFICOS

Dos objetivos específicos visa-se:

- Apresentar os requisitos necessários para um sistema de monitoramento de qualidade do solo;
- Desenvolver o projeto da estrutura da rede de monitoramento, incluindo protocolos de comunicação e o estudo dos dispositivos eletrônicos dos módulos utilizados para simulação;
- Desenvolver a aplicação de monitoramento no Contiki, com o servidor para a coleta de dados;
- Minimizar o gasto de energia no sistema de monitoramento, através de modificações na configuração do RPL;
- Realizar a simulação do sistema no ambiente Cooja e coletar os dados de consumo de energia, além do atraso, perda e jitter na comunicação.
- Calcular a distância máxima possível para a fonte de emissão eletromagnética de maneira que o sistema ainda possua alimentação suficiente.

1.2 MATERIAIS E MÉTODOS

1.2.1 MATERIAIS

Os materiais utilizados para o desenvolvimento do sistema de monitoramento são:

- Computador com sistema operacional Linux;
- Linguagem de programação C;
- Compilador da linguagem C;
- Sistema Operacional Contiki;
- Simulador Cooja.

É importante montar pelo menos um módulo da rede para que seja possível medir o gasto de energia durante sua operação e transmissão de mensagens. A partir disso, será possível mensurar a quantidade de energia necessária a ser coletada para manter o balanço de energia neutro e comparar com as simulações.

1.2.2 CONJUNTO DE DADOS

Para o estudo, serão necessários os seguintes levantamentos de dados:

- Dados de temperatura e umidade crítica para a saúde da soja em cultivo;
- Dados da quantidade adequada de nitrogênio, potássio e fósforo no solo para o crescimento da soja e de quão frequente deve ser o monitoramento e a distância entre os sensores;
- Dados da potência média recebida de transmissoras de televisão digital, 4G e 5G na região do campo;
- Dados da atenuação de ondas transmitidas nos campos de soja.

1.2.3 RECURSOS COMPUTACIONAIS

Essencialmente serão usados três recursos computacionais, um compilador em C como o GCC, um ambiente de desenvolvimento como o Visual Studio para o desenvolvimento de protocolos para o SO Contiki e o emulador do Contiki, Cooja que permitirá simulações de diversas situações e casos que podem ocorrer no campo.

1.2.4 MÉTODOS

A metodologia de desenvolvimento do projeto se baseará nos dados obtidos na literatura e comentados como necessários na seção 6.2 Conjunto de Dados. A partir desses dados define-se com qual frequência serão enviados os dados acerca da qualidade do solo, no quesito de adubação e para quais níveis de temperatura e umidade devem ser enviadas as mensagens de alerta.

Portanto, junto com as determinações das especificações eletrônicas já pensadas para o menor gasto energético, se chegará à conclusão do quanto de energia é necessário para fazer a medição dos dados e transmiti-los para cada módulo da rede e o quão frequente deve ser esse gasto energético.

Essas informações, somadas às especificações do circuito receptor e armazenador de energia, como energia captada e eficiência, obtidas da literatura, adicionadas aos dados de potência recebida de ondas eletromagnéticas no campo, permitirão uma análise inicial do balanço energético do sistema.

Após essa análise, será desenvolvida a aplicação em linguagem C no Contiki de modo que seja minimizado o gasto energético, garantindo que as informações essenciais sejam transmitidas. O consumo de energia será estimado usando o simulador Cooja com os parâmetros de simulação obtidos da literatura e das especificações dos componentes eletrônicos utilizados.

A primeira simulação a que os protocolos serão submetidos considerará as condições ideais de uma cultura de soja. Caso o protocolo desenvolvido não seja suficiente para garantir um balanço energético positivo, ele será imediatamente rejeitado e ajustado para atender a esse requisito.

Se o protocolo atender às condições ideais, será então testado em parâmetros mais realistas, passando por um ciclo contínuo de simulações até que se chegue à conclusão de que o balanço energético positivo é inviável nas condições rurais ou que ele seja alcançado dentro das diferentes variáveis do ambiente rural.

Contudo, caso o protocolo satisfaça esse caso ideal, ele será submetido novamente, mas com parâmetros mais realistas, continuando este processo até que se conclua que não é viável conseguir o balanço energético positivo nas condições rurais ou que consiga-se esse objetivo dentro das diversas condições que podem ocorrer dentro do ambiente rural.

Finalmente, mesmo que não seja viável obter um balanço energético positivo, espera-se desenvolver protocolos e escolher equipamentos eletrônicos que minimizem o gasto energético para que o período entre a necessidade de recarga de baterias seja maximizado.

A proposta deste trabalho é a implementação de uma rede de monitoramento da qualidade do solo para o cultivo da soja, definindo os fatores necessários para o desenvolvimento e viabilidade do projeto de rede, protocolos de comunicação e o projeto eletrônico dos módulos usados na rede, de modo que seja minimizado o gasto de energia no sistema. A implementação será realizada no ambiente do Cooja. Desse modo, utilizando a simulação, serão realizados ajustes no protocolo RPL para reduzir o consumo de energia e objetivando o uso de captura de energia (*energy harvesting*) para os dispositivos.

Em seguida, serão avaliados os sensores, microcontroladores e rádios necessários para obter, interpretar e transmitir essas informações e as características eletrônicas e físicas relevantes para o projeto, como consumo de energia. Na sequência, será analisado o sistema operacional Contiki e as modificações que podem ser feitas em suas configurações e protocolos para minimizar o gasto energético do módulo da rede. Nesse viés, será analisado o ambiente dos campos de soja para analisar as ondas eletromagnéticas que podem estar sendo transmitidas em suas áreas e estimado a quantidade de energia que pode ser retirada de sistemas 4G/5G ou TV Digital em diversas distâncias da antena transmissora.

Essas modificações serão testadas no emulador Cooja, possibilitando realizar diversas modelagens das possíveis situações nos campos de soja, analisando se os gastos energéticos são menores do que o coletado das ondas eletromagnéticas. Isso será feito utilizando cenários de testes, variando os parâmetros do protocolo RPL e do Contiki.

2 REVISÃO DE LITERATURA

2.1 INDICADORES DA SAÚDE DA SOJA

É crucial para os agricultores minimizar perdas e maximizar a colheita de seu plantio, buscando eficiência, para garantir não apenas a viabilidade econômica de suas operações, mas também a sustentabilidade ambiental e a segurança alimentar da sociedade. Nesse contexto econômico, reduzir perdas e aumentar a produção significa maiores lucros, preços mais competitivos e alimentos mais baratos para o consumidor, tornando essencial a otimização da produção para aproveitar ao máximo os recursos investidos e limitados, como terras férteis. Com isso em mente e considerando a competitividade do mercado nacional e internacional de soja, torna-se muito interessante para o agricultor possuir formas de verificar a qualidade dos grãos e evitar situações que possam acarretar danos nas plantas e consequentemente em perdas no cultivo.

2.1.1 TEMPERATURA

O sucesso no cultivo da soja está diretamente relacionado a condições climáticas favoráveis, especialmente à temperatura. De acordo com FARIA et al. [10], a faixa ideal para o desenvolvimento saudável da planta está entre 20°C e 30°C, sendo que 30°C é considerada a temperatura ótima para o crescimento da cultura. Durante a semeadura, a temperatura do solo também desempenha um papel fundamental, com valores entre 20°C e 30°C promovendo uma germinação uniforme e rápida. Dentro dessa faixa, 25°C se destaca como especialmente benéfica. Por outro lado, temperaturas abaixo de 10°C podem limitar ou até mesmo impedir o crescimento vegetativo, enquanto valores superiores a 40°C podem prejudicar o desenvolvimento, interferindo na floração e reduzindo a retenção de vagens.

2.1.2 DISPONIBILIDADE DE ÁGUA

Segundo FARIA et al. [10], a disponibilidade de água no solo é um fator essencial para o desenvolvimento das plantas, sendo responsável por cerca de 90% do peso da planta e influenciando diversos processos fisiológicos e bioquímicos. Na soja, a relevância da água é particularmente evidente em dois momentos críticos: germinação/emergência e floração/enchimento de grãos. Durante a fase inicial, tanto o excesso quanto a falta de água podem comprometer a cultura. Para uma germinação eficiente, as sementes precisam absorver pelo menos 50% do seu peso em água, sendo ideal manter o teor de água no solo entre 50% e 85% da capacidade total disponível.

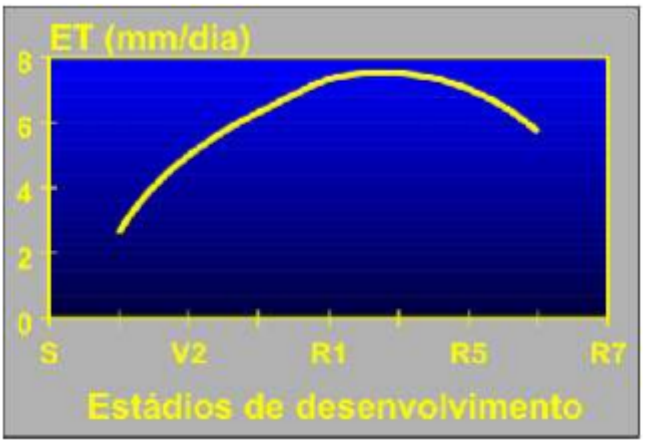

Figura 1 - Evapotranspiração (ET) diária da cultura de soja em seus estádios de desenvolvimento [10].

Com o avanço do ciclo, a necessidade hídrica da planta aumenta, atingindo seu ponto máximo na fase de floração/enchimento de grãos, com uma demanda diária média de 7 a 8 mm, como ilustrado na Figura 1. Após essa etapa, a demanda por água começa a diminuir gradativamente. Entretanto, déficits hídricos significativos durante esse período podem causar alterações fisiológicas na planta, como fechamento estomático, enrolamento das folhas, queda precoce de folhas e flores, além do aborto de vagens, o que impacta negativamente o rendimento dos grãos. Estima-se que a necessidade total de água para alcançar o máximo potencial produtivo da soja varie entre 450 e 800 mm ao longo do ciclo.

2.1.3 FOTOPERÍODO

De acordo com FARIA et al. [10], o desenvolvimento da soja também é afetado pelo comprimento do dia, devido à sua sensibilidade ao fotoperíodo, que varia conforme a latitude da área de cultivo. Sendo uma planta de dias curtos, a soja inicia a floração quando o período diário de luz atinge um limite crítico específico para a espécie. Contudo, pesquisas realizadas no Brasil possibilitaram a introdução de genes em diferentes variedades de soja, resultando no prolongamento do período juvenil da planta.

Esse fenômeno, conhecido como período juvenil longo, permitiu o surgimento de cultivares adaptadas às condições das regiões tropicais, com elevado potencial produtivo e qualidade de grãos, além de características agronômicas vantajosas. Tais avanços têm desempenhado um papel fundamental na expansão e otimização do cultivo de soja, especialmente em áreas com climas tropicais.

2.1.4 POTENCIAL HIDROGENIÔNICO

Adicionalmente, a disponibilidade dos diversos nutrientes que contribuem para o crescimento das plantas, incluindo a soja, é influenciada por vários fatores, como o pH do solo [11]. A Figura 2 demonstra a relação entre a disponibilidade dos elementos químicos utilizados pelas plantas em função do pH do solo [11]. Nela é possível ver que os principais nutrientes para o crescimento da soja (Potássio, Nitrogênio, Fósforo, entre outros) possuem valores próximos de seus respectivos máximos de disponibilidade para valores de pH entre 6,5 e 7, ou seja, para que estejam disponíveis para o crescimento da soja saudável, o pH ideal do solo é próximo de 6,5 e o agricultor deve fazer o possível para mantê-lo nesse alcance de valores.

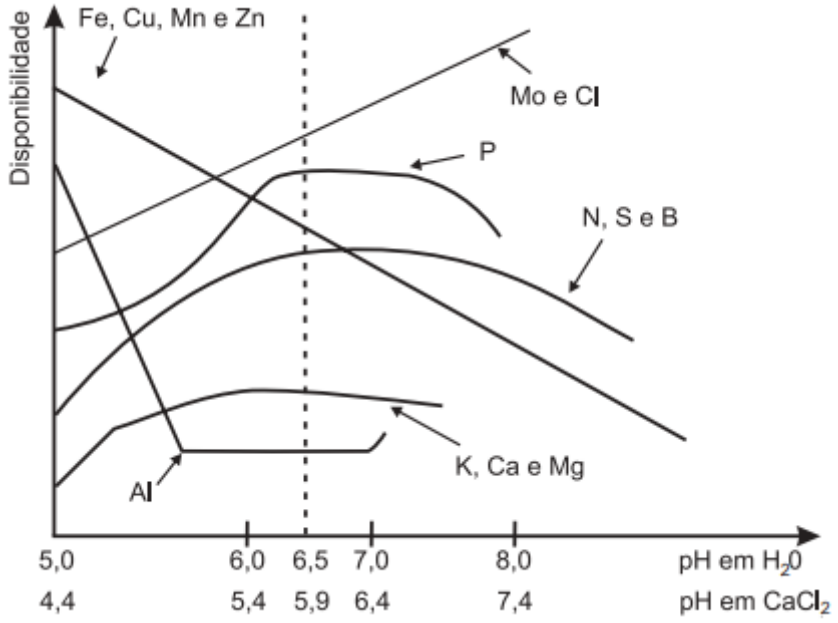

Figura 2 - Relação entre o pH e a disponibilidade de nutrientes do solo [11].

Dessa forma, o nível de pH do solo torna-se interessante para que o agricultor possa saber se deve fazer a calagem ou adubação, devido a relação de Nitrogênio, Fósforo, Potássio e Cálcio com o pH.

2.1.5 ADUBAÇÃO

Segundo HUNGRIA et. al. [12], a adubação também é relevante para o máximo rendimento da cultura da soja, pois desempenha um papel fundamental na garantia da nutrição balanceada da planta. Dessa forma, o elemento nitrogênio (N) é de extrema importância para o cultivo da soja, visto que os grãos são notavelmente ricos em proteínas, contendo cerca de 6,5% de nitrogênio em média. Isso implica que, para produzir 1000 kg de grãos de soja, são necessários aproximadamente 65 kg de nitrogênio. Além disso, para atender às demandas das folhas, caules e raízes, é recomendável adicionar pelo menos mais 15 kg de nitrogênio, totalizando uma necessidade de 80 kg de nitrogênio. Logo, para alcançar rendimentos de 3000 kg de grãos por hectare, são requeridos 240 kg de nitrogênio, dos quais 195 kg são removidos da plantação pelos grãos. A determinação da quantidade adequada de nutrientes, especialmente no que diz respeito à adubação corretiva, é estabelecida a

partir das informações obtidas na análise do solo. Entre os elementos que podem ser ajustados estão o Nitrogênio (N), o Fósforo (P), o Potássio (K) e o Enxofre (S) e existem fórmulas no mercado que possuem os quatro elementos juntos [11].

2.2 SENSORIAMENTO

O sensoriamento do solo na agricultura de soja é uma prática que permite fornecer informações detalhadas e em tempo real sobre as condições do plantio. Por meio de uma variedade de técnicas e tecnologias, os agricultores podem coletar dados sobre a fertilidade, umidade, textura e outros aspectos do solo que são essenciais para o crescimento saudável da soja e de outras plantas [13]. Nesse sentido, uma das principais vantagens do sensoriamento do solo é a sua capacidade de otimizar o manejo agrícola. Por exemplo, ao mensurar as variações na disponibilidade de nutrientes e água no solo dentro de uma área de cultivo, os agricultores podem aplicar fertilizantes de forma precisa ou irrigar apenas uma área específica, economizando recursos e reduzindo custos [13]. Isso não apenas aumenta a eficiência do uso de insumos, mas também reduz o impacto ambiental associado ao excesso de aplicação de produtos químicos e desperdício de água potável na agricultura [13]. Além disso, o sensoriamento do solo permite uma abordagem proativa para o manejo de problemas potenciais, como a compactação do solo ou infertilidade do solo. Ao identificar áreas perdendo sua fertilidade, os agricultores podem implementar práticas de manejo, como a utilização de técnicas de cultivo mínimo ou adubação do solo, para melhorar as condições de crescimento das plantas de soja e maximizar a absorção de nutrientes. Para tal, existem diversos sensores que medem grandezas diferentes, como a umidade do solo, a temperatura do solo, pH do solo, quantidades de íons de N, P, K, Mg etc, entre outros sensores, cada um com seu próprio princípio de funcionamento e muitas vezes existindo mais de uma maneira de medir a mesma grandeza [14]. Visto que cada tecnologia de sensoriamento de solo tem suas vantagens e desvantagens, e que nenhum sensor isolado pode abranger todas as propriedades do solo, é crucial escolher um conjunto complementar de sensores para medir as propriedades necessárias do solo [15]. A integração de vários sensores de solo próximos em uma única plataforma multi-sensor pode trazer uma série de benefícios operacionais em comparação com

sistemas de sensor único [15]. Isso inclui um desempenho operacional mais robusto, uma vez que medidas independentes são realizadas no mesmo solo, uma cobertura de atributos mais ampla, e uma maior dimensionalidade do espaço de medição, com diferentes sensores capturando várias partes do espectro eletromagnético [15]. Assim, serão detalhados os sensores que serão utilizados, a necessidade deles e seus princípios de funcionamento.

2.2.1 NECESSIDADE DE SENSORES

Como mencionado, há vantagens significativas em possuir informações de temperatura, umidade e pH do solo, além de ser interessante saber diretamente as quantidades de N, P, K disponíveis no solo para que seja possível para o agricultor tomar decisões informadas e administrar com precisão a irrigação e adubação do solo [10][11][12].

De acordo com ROSSEL et al. [15], para que seja possível obter informações sobre todas as propriedades necessárias para mensurar a fertilidade, é necessária uma combinação de sensores que se complementam e consigam dados de múltiplas grandezas diferentes através de suas medições. A resistividade elétrica (RE) pode ser empregada para determinar a distribuição de resistividade do volume de solo medido. Geralmente, as medições de RE requerem quatro eletrodos: dois para injetar a corrente (eletrodos de corrente) e dois para medir a diferença de potencial resultante (eletrodos de potencial). A resistividade do solo é determinada a partir disso e medições da condutividade elétrica aparente (CEa) são possíveis porque a resistividade é o inverso da condutividade. Sensores eletroquímicos foram desenvolvidos para medir íons específicos em solução. Os mais comuns são os eletrodos para medir o pH; no entanto, seu uso para medir vários outros íons está aumentando em aplicações ambientais. Sua durabilidade, portabilidade, resposta rápida e capacidade de medir em suspensões de solo não filtradas são vantagens-chave que permitem medições diretas das propriedades químicas do solo ISFETs, combinando a tecnologia ISE com a do transistor de efeito de campo (FET). A construção do ISFET é semelhante à do FET padrão; no entanto, o gate é substituído por um eletrodo separado (em contato com o eletrólito de análise) e o óxido isolante exposto (comumente SiO_2, mas também Al_2O_3, Ta_2O_5) também é

mantido em contato com o eletrólito sendo analisado. A carga desenvolvida na superfície do óxido (devido à interação com prótons) agora controla a corrente fonte-dreno do FET, que então é indicativa do eletrólito. As principais vantagens dos ISFETs de pH em relação aos eletrodos de pH de vidro padrão são o tamanho reduzido, maior durabilidade, resposta rápida e a capacidade de produção em massa usando técnicas de fabricação microeletrônica. RAMANATHAN et al. (2006) descrevem uma série de estudos de caso de redes sem fio para monitorar CO2, temperatura e umidade do solo. Esses sistemas também incorporam ISEs seletivos para amônio, cálcio, carbonato, cloreto, pH, oxidação-redução e nitrato. Os principais problemas com a detecção sem fio usando ISEs são a durabilidade do sensor e dificuldades com a calibração no local. Por exemplo, CHRISTY et al. (2004) relataram o uso de uma plataforma de sensor móvel que mede simultaneamente o pH do solo e a CEa. Um sensor NIR também foi adicionado a essa plataforma de múltiplos sensores (CHRISTY, 2008). O conteúdo de água no solo pode ser medido usando a resistência elétrica. Embora o conteúdo total de água no solo, medido por esses sensores, seja útil, as medições da capacidade de água disponível para as plantas (PAWC) são mais importantes para a agricultura. À (PAWC) é determinada no campo medindo as diferenças entre o conteúdo volumétrico de água nos limites superiores e inferiores após a extração completa da água pelas plantas. As redes de sensores sem fio (VELLIDIS et al., 2008) também podem ser usadas para isso. Levantamentos EMI medidos em diferentes momentos podem ser usados para aproximar a PAWC (JIANG et al., 2007; WONG et al., 2006), mas a calibração local e a interpretação cuidadosa são imperativas. Como medida de acidez, o nível de pH do solo é importante em muitos processos, incluindo a disponibilidade de nutrientes para as plantas e a eficácia de herbicidas. O pH do solo, a capacidade de tamponamento e a necessidade de calagem podem ser medidos usando sistemas de ISE ou ISFET. Por isso, serão utilizados dois tipos de sensores, um sensor de condutividade ou resistência elétrica do solo para medição do conteúdo de água no solo e um sensor de pH do solo para verificar a disponibilidade de nutrientes e necessidade de adubação conforme mostrado na Figura 2.

2.2.2 PERIODICIDADE DE LEITURAS

Segundo KONE et al. [18], a periodicidade das leituras em uma rede de sensores para a agricultura de precisão é uma das principais métricas e preocupações para garantir um sistema com baixo gasto energético, devido a maior parte da energia gasta de um módulo de sensoriamento ser nas transmissões de mensagens. Com isso em mente, é preciso ter uma frequência de amostragem dos dados pequena o suficiente para que o consumo de energia não seja tão alto (ainda mais pensando em obter o balanço energético neutro ou positivo), mas que ainda seja suficiente para garantir uma precisão satisfatória dos dados medidos. Além disso, devido às limitações de largura de banda e memória dos nós de sensores sem fio, uma alta frequência de amostragem pode causar altas latências e perdas de pacotes, resultando em um baixo nível de qualidade de serviço (QoS) e em um gasto de energia maior para o reenvio da mensagem. Assim, a frequência de amostragem deve ser um parâmetro ajustado para atender aos requisitos da aplicação agrícola.

Conforme KONE et al. [18], ele busca oferecer uma alta precisão de observações tanto em termos de tempo quanto de espaço para agricultura de precisão ou outras aplicações de redes de sensores sem fio. Isso é alcançado aumentando, sob demanda do Sistema de Suporte à Decisão (DSS), a frequência de amostragem de pelo menos uma Rede Pessoal de Área Local sem Fio - WPAN, ajustando os parâmetros do protocolo IEEE 802.15.4 MAC. Esse aumento é limitado pelos requisitos da aplicação em termos de confiabilidade, atraso e consumo de energia, garantindo assim a vida útil necessária da rede. O consumo máximo de energia no nó sensor é essencial para garantir uma carga útil mínima da rede como um todo (por exemplo, 30 dias no estudo). É notado que o consumo de energia aumenta com a demanda (frequência de envio de pacotes por sensor), independentemente do tamanho da WPAN. Isso ocorre porque aumentar a frequência de amostragem resulta em mais tentativas de transmissão, aumentando assim o consumo de energia. Além disso, é observado que o consumo médio de energia por nó sensor é menor em WPANs grandes do que em WPANs pequenas. Isso pode ser explicado pelo fato de que, em situações de alto congestionamento, os pacotes têm mais chances de colidir e serem retransmitidos, o que aumenta a

probabilidade do canal estar ocupado. Isso faz com que os nós sensores permaneçam mais tempo em espera ociosa, consumindo menos energia. Assim, um nó sensor em WPANs grandes consome menos energia do que em WPANs pequenas. Em conclusão, a frequência máxima de amostragem necessária para garantir a vida útil mínima exigida pela aplicação é maior em WPANs grandes (0,25 pacotes por segundo por nó sensor) do que em WPANs pequenas (0,05 pacotes por segundo por nó sensor), conforme ilustrado na Figura 3.

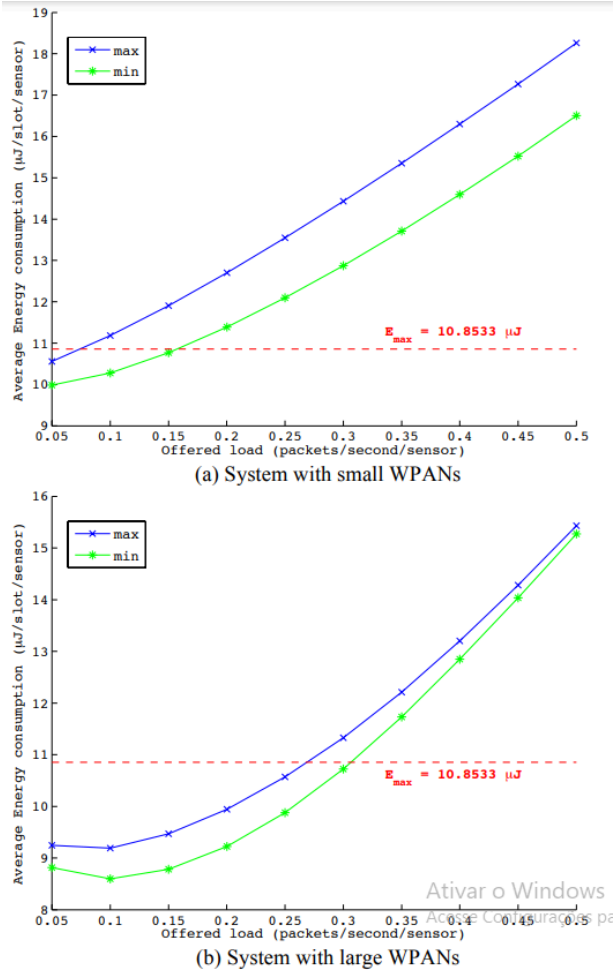

Figura 3 - Consumo médio de energia em função da frequência de envio de pacotes por sensor em grandes e pequenas WPANs [18].

Com esses dados, é possível ter uma noção de um valor inicial próximo do ideal para a frequência de envio de mensagens para ser utilizado nas simulações e testes da rede. Assim, será utilizado inicialmente um valor de 0,05 pacotes por segundo nas simulações para verificar o consumo energético, pois, segundo KONE

[18], este valor garante uma vida útil dos módulos de pelo menos um mês para uma bateria de 87912 J sem comprometer a confiabilidade do sistema implementado. Após as primeiras simulações esse valor será alterado caso seja necessário para manter o balanço energético neutro.

2.2.3 POSICIONAMENTO DOS SENSORES

Segundo ZHANG [16], as características de reflexão de cada campo agrícola podem afetar a distância de propagação de rádio. Nesse contexto, alguns dos sinais de rádio são refletidos ou absorvidos pela vegetação do solo. Isso causa diferenças na cobertura de rádio e tem um impacto na capacidade de alcance da rede de sensores sem fio. Alguns tipos de sensores podem ser embutidos no solo, mas os módulos de rádio, pelo menos suas antenas, devem ser colocadas acima do solo para evitar as plantas. Isso ocorre porque 2,4 GHz é a frequência de absorção da molécula de água. Por conta disso, as plantas próximas ao solo podem causar grande perda de sinal durante a sua propagação. No entanto, a unidade de rádio não pode estar muito longe do solo, caso contrário, pode aumentar a dificuldade ao implementar a rede de sensores. A altura ótima seria tanto econômica quanto boa para a propagação de rádio. Três conjuntos de dados foram obtidos em ambientes agrícolas típicos no centro de Illinois e resumidos na Tabela 1, apenas a imagem com o gráfico da soja é mostrada, pois é o interessante para este trabalho. Eles eram campos de solo nu, de soja e de milho. Com o alcance de rádio, pode-se determinar quantos sensores devem ser implantados para cobrir todo o campo, seu posicionamento conforme o limite de alcance do rádio e qual é a altura ideal das unidades de rádio para aplicações específicas. A Tabela 1 resume a altura de rádio ideal e seu alcance de rádio em diferentes ambientes agrícolas, note que quanto maior a altura das plantas cultivadas menor o alcance do rádio.

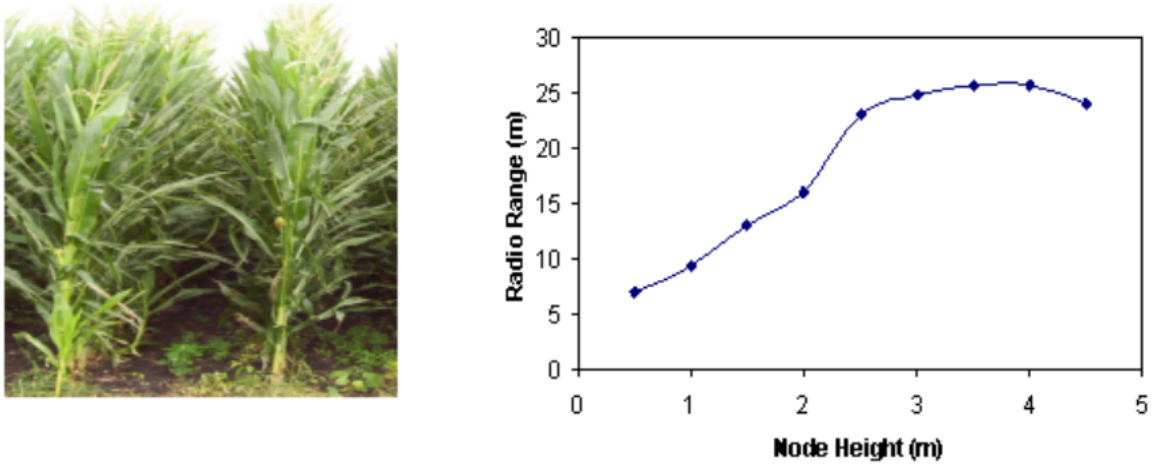

Figura 4 - Alcance de rádios Bluetooth e IEEE 802.11 na faixa de 2,4 GHz em função da altura da antena em plantações agrícolas [16].

Tabela 1 - Resumo alcance de rádios Bluetooth e IEEE 802.11 e altura ótima da antena [16].

Lavoura	Altura ótima da antena (m)	Alcance do rádio (m)
Solo sem cobertura	1,4	44
Plantação de soja	1,7	37
Plantação de milho	4,0	25

Além do limite de alcance do rádio, é necessário se preocupar com a precisão das informações devido ao posicionamento espacial dos sensores, pois as características do solo variam conforme a região do solo. Idealmente, para eliminar o máximo possível de erros de medição, seriam instalados sensores próximos de cada raiz das plantas na cultura. Embora, não seja viável economicamente atribuir um sensor para cada raiz, é possível instalar todos os sensores na profundidade e proximidade das raízes para garantir uma maior precisão espacial dos sensores.

GRISSO et. al. [17], por exemplo, criou um modelo que fornece leituras de CE (Condutividade Elétrica) de duas profundidades diferentes (30 cm a 90 cm). A distância entre as passagens de medição varia de 6 a 18 metros, dependendo da densidade de amostragem desejada ou da quantidade de variabilidade do solo dentro do campo. O autor recomenda que se colete dados com espaçamento entre passagens de medição não superior a cerca de 18 metros. O experimento feito pelo autor mostra que passagens de 12 a 18 metros fornecem um mapa de CE que

identifica adequadamente os padrões espaciais de um campo. Tal passagem pode representar metade até uma largura completa da barra de pulverização ou um múltiplo da largura do plantador ou da colheitadeira e, consequentemente, a menor área que a maioria dos agricultores irá gerenciar de forma variável.

Dessa forma, para as simulações será utilizado um espaçamento inicial de 15 metros e ajustado conforme o necessário para alcançar o balanço energético neutro.

2.3 CONTIKI

O Contiki [9] é um sistema operacional de código aberto, altamente modular e personalizável, desenvolvido especificamente para dispositivos da Internet das Coisas (IoT) e sistemas embarcados com recursos limitados, como memória e, no caso deste projeto, energia. Ele oferece suporte a tecnologias essenciais, como IPv6, redes Ad Hoc/Mesh e o protocolo de roteamento RPL. As aplicações para o Contiki são desenvolvidas em linguagem C, amplamente reconhecida e utilizada na programação de sistemas embarcados.

Além disso, o Contiki conta com um ambiente de simulação chamado Cooja, que emula o sistema operacional em cenários de aplicação. O Cooja permite simular transmissões no meio físico, possibilitando a criação e avaliação de cenários complexos sem demandar grandes investimentos financeiros e otimizando o tempo de desenvolvimento. Para este projeto, o Contiki será o sistema operacional dos módulos sensores, enquanto o Cooja será utilizado na concepção e avaliação dos protocolos de rede.

2.4 RPL

Como mencionado, será utilizado o protocolo de roteamento para redes de baixo consumo de energia e com perdas (RPL) [4][19]. Seu principal objetivo é ser eficiente para redes de sensores sem fio, minimizando o consumo de energia dos dispositivos da rede, adaptando-se conforme as configurações e características atuais da rede e suportando redes de tamanho variado, utilizando o IPv6 [4][19]. Também suporta topologias de rede em árvore (DODAG) e outras podem ser feitas

adaptando o DODAG, também, permite que a métrica utilizada para a formação de rotas seja modificada, minimizando o gasto energético, latência ou confiabilidade. O RPL é otimizado para capacidade de processamento limitado e pouca memória, que é o caso de redes de monitoramento [4][19]. No RPL, cada nó age como um roteador utilizando o IPV6 [19]. O RPL é um protocolo de roteamento baseado no algoritmo vetor distância.

Segundo TSVETKOV [19], com essa finalidade, é formado o Grafo Direcionado Acíclico Orientado ao Destino (Destination Oriented Directed Acyclic Graph, DODAG), que é direcionado para um único nó. De acordo com a especificação do RPL, esse nó específico é denominado a raiz do DODAG (DODAG root ou sink). O grafo é criado utilizando uma Função Objetivo (OF), que determina como a métrica de roteamento é calculada, baseado em um critério escolhido pelo projetista. Nessa estrutura, existem um ou mais nós que são os nós raízes da rede, e os outros nós são organizados hierarquicamente abaixo deles. Cada nó mantém um *Rank*, que é uma medida de sua posição em relação a um nó raiz. Para evitar loops de roteamento, o protocolo determina a posição de um nó em relação aos demais e a raiz do DODAG. Essa posição aumenta quando os nós se afastam da raiz e diminui quando se aproximam. O *Rank* pode ser uma simples contagem de saltos, ser calculado com base na métrica de roteamento ou levar em consideração outras restrições desejadas pelo projetista e é usado para determinar a rota preferida em direção à raiz. O DODAG é construído dinamicamente à medida que os nós da rede se comunicam. Cada nó descobre sua posição relativa na topologia e escolhe as rotas apropriadas para formar e manter a estrutura do DODAG.

De acordo com TSVETKOV [19], a especificação do RPL define quatro tipos de mensagens de controle para manutenção de topologia e troca de informações. O primeiro tipo é o Objeto de Informações DODAG (DIO), que é a principal fonte de informações de controle de roteamento. Ele pode conter dados como o *Rank* atual de um nó, a Instância RPL atual e o endereço IPv6 da raiz. O segundo tipo é o Objeto de Anúncio de Destino (DAO), utilizado para propagar informações de destino ao longo do DODAG. O terceiro tipo é a Solicitação de Informações (DODAG DIS), que permite que um nó solicite mensagens (DIO) de um vizinho alcançável. Por fim, o quarto tipo é o DAO-ACK, enviado em resposta a uma mensagem DAO. O RPL ajusta a taxa de envio de mensagens DIO por meio do algoritmo Trickle. As

mensagens DIS DODAG (*Information Solicitation*) são utilizadas pelo protocolo RPL para solicitar informações sobre a topologia da rede, descobrir DODAGs disponíveis na rede e determinar quais deles ele pode se juntar. Isso é crucial para a construção e manutenção do DODAG na rede. Um nó envia uma mensagem DIS para o endereço *multicast* de solicitação de informações (solicitado na solicitação de informações) em intervalos regulares ou sempre que necessário para atualizar sua visão da topologia da rede e tabela de roteamento. Os nós que recebem uma mensagem DIS e fazem parte de um DODAG enviam uma mensagem de informações do DODAG (DIO - DODAG *Information Object*) em resposta. Essas mensagens contêm informações sobre o DODAG, incluindo seu identificador, o endereço do nó raiz, parâmetros de configuração e outras informações relevantes. Com base nas informações recebidas nas mensagens DIO, um nó pode decidir se deseja se juntar a um DODAG específico e, em caso afirmativo, qual será seu próximo salto no DODAG.

Essencialmente, no RPL cada nó age como um roteador e mantém sua própria tabela de roteamento, na qual possuem apenas uma rota para se comunicar com cada rede (ou, no caso, nós raízes), sendo ela a mais eficiente com base na função objetivo em uso. Para esclarecimento, suponha uma rede utilizando o protocolo RPL com a configuração mostrada na Figura 5. As arestas representam quais nós da rede conseguem se comunicar e os pesos representam o número esperado de retransmissões necessárias para o recebimento da mensagem. Esta métrica é chamada de ETX (*Expected Transmission Counter*).

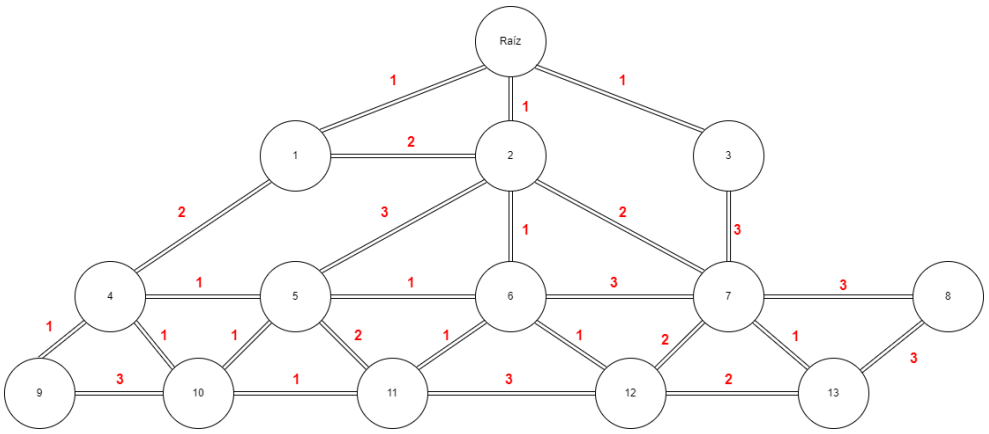

Figura 5 - Alcance dos nós da rede com o protocolo RPL e retransmissões necessárias.

Antes dos grafos serem formados nenhum nó da rede conhece seus vizinhos, para tal, ele envia uma mensagem DIS no endereço *multicast* da rede, os vizinhos o recebem e retornam a mensagem DIO contendo as rotas conhecidas por eles para alcançarem o nós raízes da rede (caso exista apenas um, é mandada apenas essa rota). Dessa forma, o nó que recebeu a mensagem DIO sabe que pode chegar no nó raíz através do nó que enviou a mensagem com n+1 saltos ou com X número de retransmissões esperadas. Assim, ele receberá essa informação de todos os vizinhos e fazendo uso da função objetivo descobrirá qual a melhor rota a ser seguida baseando-se nos critérios dela. Esta rota é então guardada na tabela de rotas do nó e as outras são descartadas. Na Figura 6 é possível ver ilustrações com o funcionamento desse mecanismo em ordem. Suponha que o nó quatro foi inserido na rede e ainda não conhece os vizinhos. Inicialmente, envia a mensagem DIS no endereço *multicast* da rede, em sequência, os nós 1 e 2 que estão no alcance do nó 4 recebem a mensagem e a respondem com o envio da mensagem DIO para o nó 4. A mensagem DIO enviada contém informações sobre a rota que os nós possuem para alcançar o nó raíz. A função objetivo escolhida no algoritmo calcula qual das rotas é a mais vantajosa para o nó 4 e a insere na sua tabela de rotas levando em consideração os dados do nó 4 e do nó escolhido para a rota. Neste caso, a métrica considerada foi o número esperado de transmissões para a mensagem chegar até o nó raiz (ETX), ou seja, a rota mantida foi através do nó 1 que chega no nó raiz com 1 salto. Dessa forma, o nó 4 mantém na sua tabela de rotas que chega no nó raiz através do nó 1 com 2 saltos.

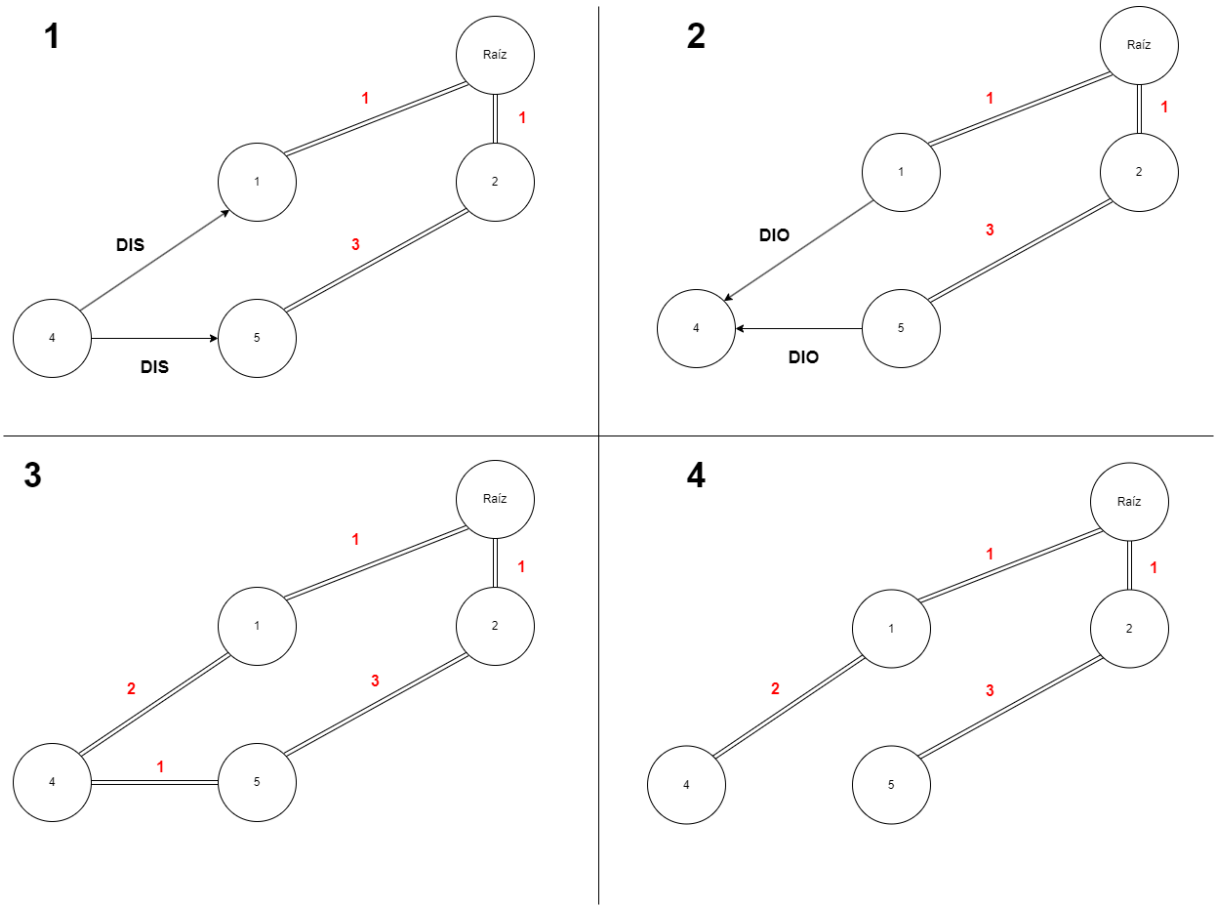

Figura 6 - Ilustração do funcionamento das mensagens DIS e DIO.

Além disso, as duas principais tecnologias que aprimoram o desempenho do RPL são a integração da função objetivo e o uso do algoritmo Trickle Timer, que serão discutidas nas seções abaixo.

2.4.1 FUNÇÃO OBJETIVO

As funções objetivo são fundamentais para o processo de otimização do RPL. Elas definem critérios específicos que guiam o algoritmo de roteamento na seleção das rotas mais eficientes, por exemplo, minimização do consumo de energia, maximização da vida útil da bateria dos dispositivos e redução de atraso na transmissão de dados. Dessa forma, elas permitem que o RPL se adapte dinamicamente às mudanças nas condições da rede e aos requisitos da aplicação, maximizando a eficiência no uso dos recursos disponíveis. Duas funções objetivo normalmente disponíveis são a OF0 e a MRHOF.

A função objetivo OF0 (*Objective Function* 0) é uma função objetivo simples, frequentemente empregada em redes de sensores sem fio e outras redes de baixo consumo de energia. A métrica considerada pela OF0 é o número de saltos necessários para encaminhar os pacotes de dados do nó de origem para o destino [4]. Um salto ocorre sempre que um pacote é encaminhado de um nó para outro na rede [4]. A ideia básica por trás da OF0 é encontrar o caminho mais curto em termos de saltos entre o nó de origem e o destino. Ao minimizar o número de saltos, espera-se reduzir a sobrecarga de comunicação e o consumo de energia, tornando a rede mais eficiente. Para determinar a rota mais adequada, cada nó mantém informações sobre seus vizinhos e os custos associados a alcançá-los [4][19]. Com base nessas informações, os nós podem calcular a rota mais eficiente em termos de número de saltos e assim construir a estrutura da rede e tabelas de roteamento dos nós [4][19]. A OF0 busca minimizar o *Rank* dos nós da estrutura DODAG ao longo da rota, garantindo assim um caminho mais curto [4][19]. Outra consideração importante ao escolher uma função objetivo é o *overhead* de controle associado. Isso inclui o tráfego de controle necessário para manter a estrutura da rede, como anúncios de rotas e atualizações de topologia, pois isso consome energia significativa. A OF0, ao priorizar o número de saltos, tende a minimizar o *overhead* de controle, pois rotas mais curtas geralmente exigem menos mensagens de controle. Suponha uma rede com a mesma configuração mostrada na Figura 7, se for aplicada a OF0 como função objetivo para a determinação das rotas, a topologia ficaria como mostrada na Figura 6. Note que os nós 1, 2 e 3 chegam ao nó raíz com 1 salto diretamente, mas o nó 4 por exemplo poderia chegar ao raíz através dos nós 1, 5, 9 e 10, entretanto o número de saltos através do nó 1 é menor do que através dos outros nós, portanto este nó é o único mantido na tabela de rotas do nó 4.

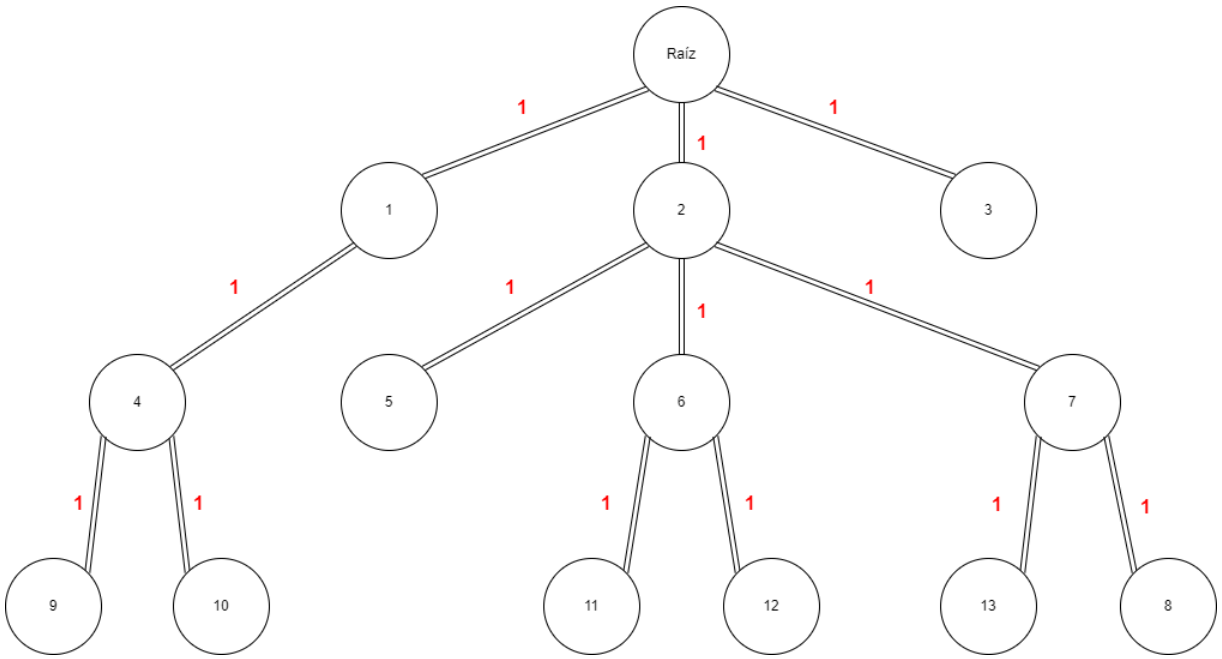

Figura 7 - Topologia da rede com o protocolo RPL usando a OF0.

.

A função objetivo MRHOF (*Minimum Rank with Hysteresis Objective Function*) é uma das funções objetivo mais avançadas e amplamente utilizadas no protocolo RPL (*Routing Protocol for Low-Power and Lossy Networks*). Ela visa otimizar a seleção de rotas em redes de baixo consumo de energia, considerando não apenas a métrica de *Rank*, mas também outras métricas como ETX (*Expected Transmission Count*), RSSI (*Received Signal Strength Indication*) e atraso. O *Rank* é uma medida atribuída a cada nó na rede, indicando sua "distância" do nó raiz (DODAG root). Quanto menor o *Rank*, mais próximo o nó está do nó raiz. Caso o MRHOF utilize a métrica *Rank*, o resultado será similar a OF0. Já o ETX é uma métrica que reflete a confiabilidade de um *link* de comunicação entre dois nós na rede. Ele representa o número esperado de transmissões necessárias para enviar com sucesso um pacote de um nó para outro. Quanto menor o valor de ETX, mais confiável é o *link*. A MRHOF pode usar o ETX como uma das métricas para calcular o *Rank* de um nó em relação à raiz do DODAG. Rotas com links de baixo ETX são preferidas, pois são mais confiáveis e têm menor probabilidade de perda de pacotes. Dando continuidade, o RSSI é uma medida da potência do sinal recebido por um nó de outro nó na rede. Ele fornece uma indicação da intensidade do sinal de rádio entre os nós. A MRHOF pode usar o RSSI como uma métrica para estimar a qualidade do

link entre os nós. *Links* com RSSI mais alto geralmente indicam uma conexão mais forte e podem ser preferidos ao calcular rotas. Finalmente, o atraso é o tempo necessário para um pacote de dados percorrer o caminho entre um nó de origem e um destino na rede. O atraso pode ser afetado por vários fatores, incluindo a carga da rede, o congestionamento e a latência dos *links*. Embora a MRHOF possa não priorizar diretamente o atraso como uma métrica principal, ele ainda pode ser considerado ao calcular o *Rank* de um nó. Em certos cenários, rotas com menor atraso podem ser preferidas, especialmente para aplicações de tempo real.

2.4.2 ALGORITMO DE TRICKLE

A outra principal tecnologia do RPL é o *Trickle Timer*, usado para determinar o intervalo entre a transmissão das mensagens de controle na rede, permitindo reduzir o gasto energético com essas comunicações. Dessa forma, ele possibilita que módulos da rede ajustem dinamicamente a taxa de transmissão das mensagens de controle com base na estabilidade da vizinhança e das rotas alternativas disponíveis. Isso não apenas economiza energia, como também aumenta a eficiência geral do envio de mensagens na rede, transmitindo mensagens apenas quando necessário.

O *Trickle Timer* permite também a configuração flexível dos seus parâmetros de intervalo de tempo mínimo e máximo entre duas transmissões consecutivas e o parâmetro de redução que controla o aumento do intervalo de tempo entre as transmissões. Nesse contexto, o *Trickle Timer* aguarda, durante a operação padrão, pelo menos o tempo do intervalo mínimo definido antes de permitir a transmissão da próxima mensagem. Conforme as mensagens de controle são transmitidas com sucesso e a rede mantém suas características, o intervalo entre transmissões de mensagens de controle aumenta. Isso ocorre porque o *Trickle Timer* multiplica o intervalo atual por um fator de redução após cada transmissão bem-sucedida, aumentando o intervalo de tempo de forma adaptativa. Ele, também, verifica se o intervalo entre duas transmissões é maior que o definido como máximo, em caso verdadeiro, ele reinicia o processo de temporização.

Conforme LEVIS et al. [20], em vez de sobrecarregar a rede com uma inundação de pacotes, o algoritmo regula a taxa de envio para garantir que cada nó

receba apenas um pequeno fluxo de pacotes suficiente para manter a consistência. Sua implementação é simples e requer poucos recursos, com atuais versões ocupando entre 4 e 11 bytes de RAM e consistindo de 50 a 200 linhas de código C. A solicitação de comentários (RFC) detalha o funcionamento do algoritmo *Trickle* e oferece orientações para sua implementação, incluindo requisitos para especificações de protocolo que o empregam. O temporizador *Trickle* funciona em um intervalo específico e é determinado por três parâmetros: o menor tamanho de intervalo *Imin*, o maior tamanho de intervalo *Imax* e a constante de redundância k. *Imin* representa o menor intervalo de tempo e é medido em unidades como milissegundos ou segundos. No exemplo de LEVIS et al. [20], um protocolo pode fixar *Imin* em 100 milissegundos. *Imax* é o maior intervalo de tempo, calculado como um múltiplo de *Imin*, usando a fórmula logarítmica (log base-2 da razão max/min). Por exemplo, um protocolo pode definir *Imax* como 16. Assim, se o menor intervalo for 100 ms, *Imax* seria 100 ms multiplicado por 65.536, resultando em 6.553,6 segundos, ou cerca de 109 minutos. k é uma constante de redundância, que é um número inteiro positivo.

Além desses parâmetros, o algoritmo *Trickle* utiliza três variáveis:

- I: O intervalo de tempo atual;
- t: Um ponto específico dentro do intervalo atual;
- c: Um contador.

Com esses elementos, o algoritmo *Trickle* segue seis regras:

1. Quando o *Trickle* começa a funcionar, ele escolhe um valor para *I* dentro do intervalo de *Imin* a *Imax*, ou seja, um valor que seja maior ou igual a *Imin* e menor ou igual a *Imax*, e então inicia o primeiro intervalo.
2. No início de um novo intervalo, o *Trickle* zera o contador *c* e escolhe um valor aleatório para *t* entre *I/2* e *I.*
3. Toda vez que o *Trickle* detecta uma transmissão que considera "consistente", ele incrementa o contador *c*.
4. No tempo *t*, o *Trickle* realiza uma transmissão somente se o valor de *c* for menor que a constante k.

5. Quando o intervalo *I* termina, o *Trickle* dobra a duração do intervalo. Se este novo intervalo exceder o valor especificado por *Imax*, *I* é ajustado para o tempo máximo permitido por *Imax*.
6. Se o *Trickle* detecta uma transmissão "inconsistente" e o valor de *I* é maior que *Imin*, ele redefine o temporizador para *Imin* e começa um novo intervalo. Se *I já* for igual a *Imin* ao ouvir uma transmissão "inconsistente", nenhuma ação é tomada. O *Trickle* também pode redefinir seu temporizador em resposta a "eventos" externos.

Segundo LEVIS et al. [20], os protocolos que empregam o algoritmo *Trickle* precisam definir valores padrão para *Imin*, *Imax* e *k*, além de especificar o que caracteriza uma transmissão "consistente" ou "inconsistente" e quais "eventos", além de transmissões inconsistentes, devem reiniciar o temporizador. O *Trickle* só realiza transmissões durante a quarta etapa de seu algoritmo. Isso significa que há um atraso deliberado entre a detecção de uma inconsistência (ajustando *I* para *Imin*) e a resposta a essa inconsistência (transmitindo no tempo *t* no novo intervalo). Esse atraso é proposital, pois uma resposta imediata à detecção de inconsistência poderia causar uma sobrecarga de transmissão, com muitos nós transmitindo simultaneamente e de maneira coordenada. Ao seguir o algoritmo *Trickle*, com um intervalo mínimo de tempo, um protocolo pode utilizar o mecanismo de supressão do *Trickle* e funcionar eficientemente em diferentes densidades de nós.

2.5 ENERGY HARVESTING

O coletor e armazenador de energia das ondas eletromagnéticas ambientes (*Energy harvesting)* [5][6] consiste em um circuito com uma ou múltiplas antenas de radiofrequência junto a um sistema de casamento de impedância para garantir a máxima transferência de potência, um filtro capacitivo para corrente contínua, um circuito de diodos para fazer a retificação da onda junto de um filtro indutivo para garantir que apenas a onda retificada passe para a parte do circuito que irá fazer o armazenamento da energia, que consiste de basicamente um capacitor capaz de armazenar bastante carga.

As especificações do circuito coletor serão baseadas na literatura, conforme projetados e confeccionados para as frequências utilizadas. Com isso em mente e pensando na aplicação rural da rede de sensoriamento, as ondas eletromagnéticas disponíveis na área são as de TV digital, redes celulares, 3G/4G/5G e transmissoras de rádio, cujas faixas de frequência estão entre 54 MHz para o VHF baixo das transmissoras de TV digital e 3600 MHz para o 5G. Dessa maneira, o coletor ideal deve operar com a maior faixa de frequência possível dentro desses limites, mas ainda mantendo uma boa eficiência de conversão. Dentro da literatura, foi encontrado o artigo de PHAM, Binh L. [21], que descreve uma antena capaz de operar em três bandas para um sistema de colheita de energia de RF. Esta antena tem frequências de ressonância em 940 MHz, 1,95 GHz e 2,44 GHz, com ganhos medidos de 0,3 dBi, 2,3 dBi e 3,5 dBi, respectivamente. Cada uma dessas frequências pode ser ajustada independentemente, sem afetar as outras. Além disso, foi apresentado um retificador de três bandas, altamente eficiente e sensível, projetado para funcionar em conjunto com a antena. Assim como na antena, cada frequência de operação do retificador pode ser ajustada independentemente, o que permite alcançar eficiências máximas de conversão de RF para DC de 80%, 46% e 42% em 940 MHz, 1,95 GHz e 2,44 GHz, respectivamente. O design e dimensão da antena proposta são mostrados na Figura 8.

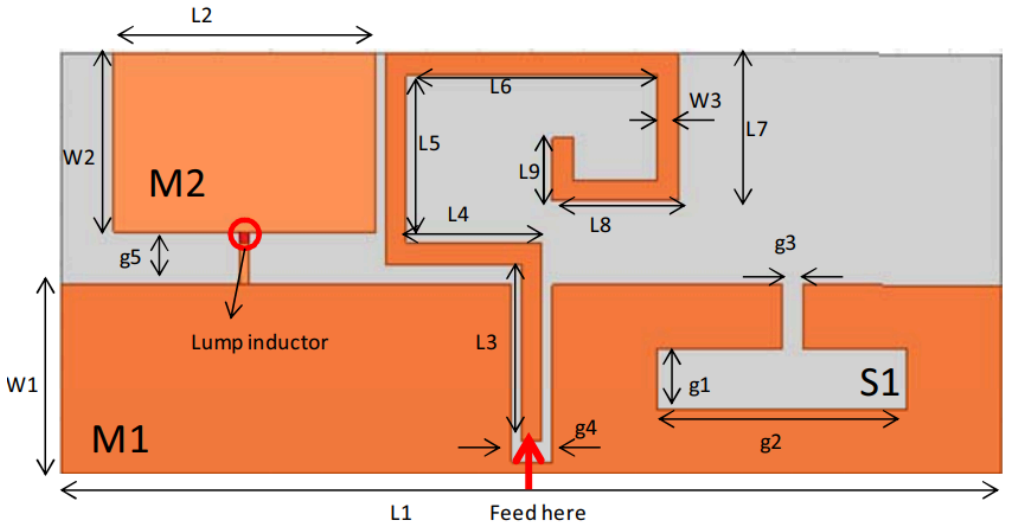

Figura 8 - Design da antena de três bandas.

As dimensões são: L1=90mm, L2=25mm, L3=17mm, L4=13mm, L5=16mm, L6=24mm, L7=14mm, L8=12mm, L9=6mm, W1=18mm, W2=17mm, W3=2mm, g1=6mm, g2=24mm, g3=2mm, g4=4mm, g5=5mm [21].

A antena tem dimensões totais de 40 mm x 90 mm e é fabricada utilizando um substrato 8-mil ROGER4003, com uma constante dielétrica ε_r de 3,55 e uma tangente de perda δ de 0,003. A antena segue um design planar, com apenas uma camada de metal na parte superior, enquanto o plano de terra fica na metade inferior, identificado como "M1" na Figura 8.

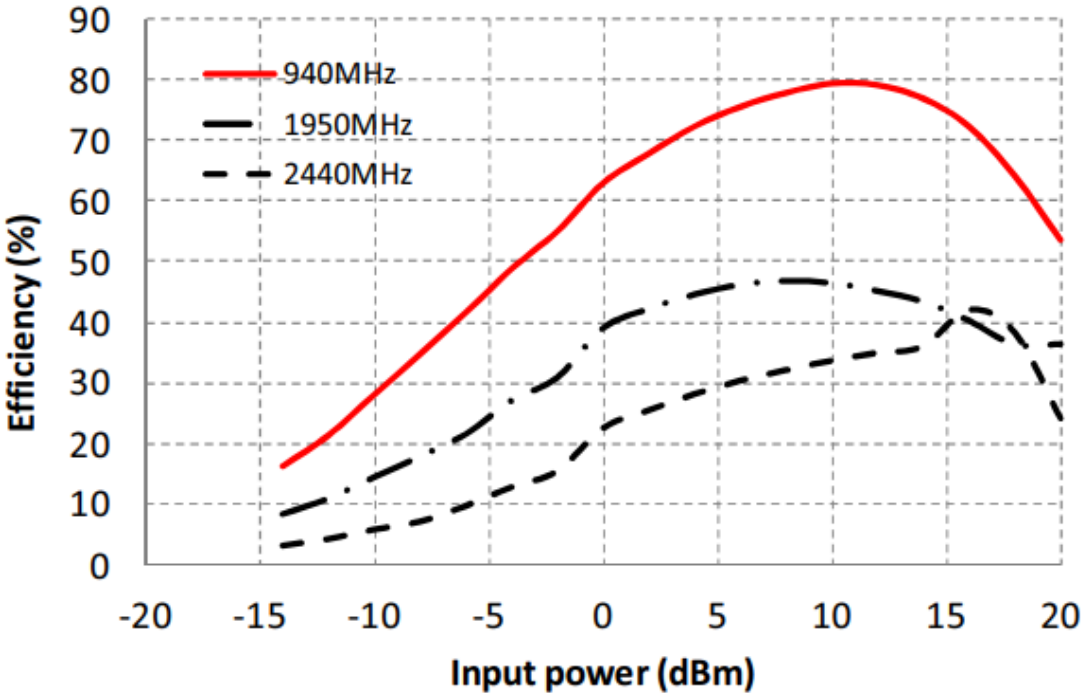

Figura 9 - Eficiência medida do retificador [21].

Segundo o autor [21], para maximizar a eficiência na colheita de energia de RF, o retificador deve ser sintonizado com a antena em três frequências distintas. Tradicionalmente, redes de casamento triplo são complexas e volumosas, mas uma nova abordagem simplifica o circuito e permite o ajuste individual de cada frequência. Essa solução utiliza um retificador de quatro estágios com uma rede LC na entrada e três indutores que controlam independentemente as frequências de casamento para 900 MHz, 1900 MHz e 2,4 GHz. Dependendo do curto-circuito de um dos indutores, o retificador ajusta-se a duas ou três dessas frequências. A eficiência máxima alcançada é de 80% a 940 MHz com 10 dBm de potência RF, 47% a 1950 MHz com 8 dBm, e 43% a 2,44 GHz com 16 dBm. Esses valores foram obtidos em medições internas com geradores de varredura *Agilent* e antenas log-periódicas.

Esses resultados serão usados como base para simulações no Cooja para determinar a captação de energia dos nós da rede e concluir o resultado final de qual deve ser a distância da fonte de emissão eletromagnética para que o sistema funcione.

2.6 TRANSCEPTOR DE RF

O CC2420 é um transceptor RF de chip único, compatível com o padrão IEEE 802.15.4, de baixa potência em 2,4 GHz, projetado para comunicação sem fio em redes de sensores. É amplamente utilizado em várias plataformas de hardware, oferecendo uma forma confiável e eficiente para os dispositivos se comunicarem sem fio. Ideal para o projeto de sistemas que exigem baixo consumo de energia, como:

- Sensores sem fio: Dispositivos que coletam dados ambientais e os transmitem para uma central, operando com baterias de pequena capacidade;
- Redes de área pessoal (PANs): Redes de curta distância que conectam dispositivos como *smartphones*, fones de ouvido e *wearables*;
- Internet das Coisas (IoT): Dispositivos conectados à internet que realizam diversas tarefas, desde controlar a iluminação até monitorar a saúde.

Ao escolher o modo de operação adequado do CC2420, é possível otimizar o consumo de energia do sistema, aumentando a vida útil da bateria e reduzindo custos.

Operação:

- Tensão: 2,1 V a 3,6 V;
- Modo OFF: < 1 µA, significa que o consumo de corrente é menor que 1 microampère, ou seja, extremamente baixo, ideal para economizar bateria em dispositivos que ficam longos períodos sem atividade;
- Modo Power Down: 20 µA, em redução de energia, o consumo aumenta para 20 microampères, ainda um valor muito baixo, mas o componente permanece pronto para ser rapidamente ativado;
- Modo Idle: 426 µA, em modo inativo, o consumo sobe para 426 microampères, indicando que o componente está em funcionamento, mas sem realizar transmissões ou recepções de dados;

- Modo RX: 18,8 mA, em recepção, o consumo aumenta significativamente para 18,8 miliampères, pois o componente está ativamente recebendo sinais;
- Modo TX: 17,4 mA, em transmissão a 0 dBm, o consumo é de 17,4 miliampères, indicando que o componente está transmitindo dados com uma determinada potência.

CC2420	2.1V to 3.6V	<1µA	OFF Mode
		20µA	Power Down
		426µA	IDLE Mode
		18.8mA	RX Mode
		17.4mA	TX Mode @ 0dBm

Figura 10 - Ganhos de corrente do Transceptor RF CC2420 [22].

2.7 PADRÃO IEEE 802.15.4

O padrão IEEE 802.15.4 [7] é uma especificação para comunicações sem fio de baixo consumo de energia para aplicações em redes de sensores, assim, os rádios projetados conforme esse padrão buscam eficiência energética em baixas distâncias, chegando a até alguns quilômetros dependendo do modelo. Por causa dessas características, eles serão utilizados no trabalho.

3 AVALIAÇÃO DE DESEMPENHO

3.1 ANÁLISE EXPLORATÓRIA

Esta seção descreve a configuração e metodologia para avaliar o desempenho da rede de sensores utilizando o protocolo RPL sob o sistema operacional Contiki. Foram realizados diversos testes e simulações no Cooja para analisar o comportamento da rede em diferentes cenários, utilizando diferentes funções objetivo e parâmetros do algoritmo *Trickle*.

3.1.1 CENÁRIOS DE TESTE

Os cenários de testes foram projetados para investigar o desempenho da

rede de sensores sob diversas condições e parâmetros, alinhando-se ao protocolo RPL e visando alcançar cenários no qual o balanço energético fosse neutro ou positivo.

As principais variáveis definidas para os cenários incluem:

- Número de Raízes: O número de nós raiz do protocolo RPL na rede que influencia as rotas possíveis e o tamanho das rotas na rede. Serão testados cenários variando de 1 a 4 raízes um em cada direção cardeal para avaliar a influência do nó raiz na economia de energia geral da rede, tratado em simulações diferentes após a análise de outros parâmetros.
- Alcance do Rádio e Espaçamento de nós: Configura a distância máxima para comunicação direta entre os nós e a distância mínima entre dois nós. O alcance máximo do rádio do Z1 conforme informado no *datasheet* é de 150 m [22] e será variado entre 50 m e 100 m nas simulações, com base nas características do rádio escolhido, ambiente usando o modelo de *Free Space* e considerando linha de visada. As distâncias entre os módulos sensores variam entre 5 metros a 100 metros. Também pode ser aumentado ou diminuído o espaçamento para melhorar o balanço energético.
- Número de Nós: A densidade de nós na rede afeta a carga de comunicação, as rotas disponíveis e o consumo de energia. As simulações incluem redes com 20 a 200 nós para observar o desempenho em diferentes escalas, modificando-os se necessário.
- Periodicidade de Mensagens: A frequência de envio de mensagens pelos nós impacta a utilização da rede e o consumo de energia. Serão avaliadas inicialmente periodicidades entre 20 s e 48 h. Elas serão ajustadas caso o resultado não cumpra o balanço energético neutro ou positivo.
- Tamanho das Mensagens: O tamanho dos pacotes de dados enviados influencia a largura de banda utilizada e a energia consumida. Serão testados tamanhos de mensagem de 64 bytes, 128 bytes e 256 bytes.

Neste trabalho, serão realizadas simulações para avaliar o desempenho de diferentes funções objetivo, incluindo MRHOF ETX, OF0 e MRHOF com bateria restante. Para isso, serão avaliados parâmetros como alcance do rádio, distância

entre nós, número de nós, consumo de energia, energia restante dos nós, número de retransmissões, PDR (%), e delay (ms). Adicionalmente, serão analisados os parâmetros do Trickle Timer, como DIO Interval e Interval Doublings, com o objetivo de identificar os intervalos ideais para o funcionamento eficiente do sistema. Em uma etapa futura, será estudada a histerese ideal para o sistema, buscando otimizar a estabilidade e o desempenho da rede.

Dessa maneira, o plano de simulações pode ser resumido da seguinte maneira:

Parâmetros			MRHOF ETX					OF0					MRHOF Bateria restante				
Alcance do rádio (m)	Distância entre nós (m)	Número de Nós	Consumo de enegia (J)	Energia restante dos nós (%)	Número de retransmissões	PDR(%)	Delay (ms)	Consumo de enegia (J)	Energia restante dos nós (%)	Número de retransmissões	PDR(%)	Delay (ms)	Consumo de enegia (J)	Energia restante dos nós (%)	Número de retransmissões	PDR(%)	Delay (ms)
50	5	20															
	25	20															
	50	20															
	5	100															
	25	100															
	50	100															
	5	200															
	25	200															
	50	200															
100	5	20															
	25	20															
	50	20															
	100	20															
	5	100															
	25	100															
	50	100															
	100	100															
	5	200															
	25	200															
	50	200															
	100	200															

Figura 11 - Plano de simulações da rede.

Parâmetros		Resultados Trickle				
DIO INTERVAL	INTERVAL DOUBLINGS	Consumo de enegia (J)	Energia restante dos nós (%)	Número de retransmissões	PDR(%)	Delay (ms)
12	8					
13	8					
14	8					
15	8					
16	8					
Melhor DIO interval	9					
Melhor DIO interval	10					
Melhor DIO interval	11					
Melhor DIO interval	12					
Melhor DIO interval	13					
Melhor DIO interval	14					

Figura 12 - Plano de simulações do algoritmo Trickle.

Assim, será seguido o plano de simulações para a aquisição de resultados. Para o algoritmo de trickle será avaliado qual DIO interval (Imin) resulta no menor consumo de energia, maior energia restante e maior PDR, com esse valor será alterado o Interval doubling (Imax) e assim determinado os melhores parâmetros do algoritmo.

3.1.2 SIMULAÇÕES INICIAIS

Para avaliar o desempenho da rede, foram realizadas simulações utilizando diferentes funções objetivo e configurações do algoritmo *Trickle*, além dos cenários mencionados acima:

1. Funções Objetivo, serão variadas as seguintes métricas em combinação com a variação de parâmetros mencionada anteriormente:
    - OF0 (*Objective Function Zero*): Esta função visa minimizar a métrica de caminho sem considerar a qualidade do *link*, o que pode resultar em rotas mais curtas, mas não necessariamente mais confiáveis;
    - MRHOF com ETX *(Minimum Rank with Hysteresis Objective Function* com *Expected Transmission Coun*t): Utiliza a contagem esperada de transmissões para avaliar a qualidade dos *links* e escolher rotas que minimizem o número de retransmissões necessárias;
    - MRHOF com RSSI (*Received Signal Strength Indicator*): Baseia a seleção de rotas na força do sinal recebido, favorecendo *links* com sinal mais forte e, teoricamente, menos propensos a perdas de pacotes;
    - MRHOF com Energia Restante no Nó: Considera a energia restante em cada nó ao escolher a rota, buscando maximizar a longevidade da rede ao equilibrar o consumo de energia entre os nós.
2. Configurações do Algoritmo *Trickle*:
    - *Imin*: É o menor intervalo de tempo entre transmissões de mensagens de controle. Serão avaliados valores de 100 ms a 1 min;
    - *Imax*: É o maior intervalo de tempo, que é um múltiplo do menor intervalo. Serão testadas configurações com 4 vezes a 16 vezes o valor de *Imin* para cumprir com os requisitos mencionados nos cenários de teste;
    - k: É a constante de redundância que controla o número de transmissões permitidas. Serão utilizados valores de 1 a 10 transmissões.
3. Distâncias de Rádio Base:
    - Potência para Recarregar Energia: A potência recebida necessária

para que os nós recarreguem suas energias. Serão testados valores entre 0 dBm a 100 dBm, traduzido para o código como potência recebida de 1 mW.

3.1.3 MÉTRICAS DE DESEMPENHO

Para cada simulação, foram analisadas as seguintes métricas de desempenho:

- Consumo de Energia:
  - Quantifica a energia total consumida pelos nós durante a simulação, proporcionando uma visão da eficiência energética da rede.
- Energia restante nos nós:
  - Quantifica a energia restante em cada nó durante a simulação, proporcionando uma visão da eficiência energética da rede, quais módulos são desligados e em quanto tempo.
- Número de Retransmissões:
  - Conta quantas vezes um pacote precisa ser transmitido, oferecendo *insights* sobre a qualidade dos *links* e a carga de rede.
- Taxa de Entrega de Pacotes (PDR):
  - Representa a proporção de pacotes que chegam ao destino em relação ao total enviado. Essa métrica indica a confiabilidade da rede.

3.1.4 CONFIGURAÇÕES INICIAIS PARA AS SIMULAÇÕES

Para iniciar as simulações, foram adotadas as seguintes configurações padrão:

- Configurações de funções objetivo:
  - OF0: Utiliza rotas mais curtas sem considerar a qualidade do *link*.
- Parâmetros *Trickle*:
  - *Imin*: variado entre 4 s;
  - *Imax*: 8 vezes *Imin*;
  - k: 4.
- Distâncias de rádio base ainda não implementadas;
- Distâncias entre os nós de 25 m;

- Alcance do rádio de 50 m;
- 20 nós;
- 1 nó raiz;
- Periodicidade aleatória em até 120 s.

3.2 RESULTADO PRELIMINARES

Inicialmente, uma revisão teórica aprofundada foi realizada para identificar os parâmetros críticos que afetam a qualidade do solo na cultura da soja. Os principais parâmetros identificados para monitoramento incluem temperatura e umidade do solo, parâmetros essenciais para o crescimento saudável das plantas de soja e indicador dos níveis de nutrientes (Nitrogênio, Potássio, e Fósforo) através do pH.

Com base na revisão teórica, foram selecionados os seguintes sensores e módulos para a implementação do consumo de energia deles na rede de monitoramento simulada:

- Sensores:
  - Sensor BNC PH4502C: Utilizado para medir a acidez (pH) do solo;
  - Sensor YL-69: Utilizado para medir a umidade do solo.
- Módulos de Comunicação:
  - Zolertia Z1 mote: Equipado com rádios compatíveis com o padrão IEEE 802.15.4, adequado para redes de sensores de baixa potência e já implementado no Cooja [22].

Foi realizada uma fase de familiarização com o sistema operacional Contiki e o emulador Cooja, necessários para a simulação e desenvolvimento da rede, a qual foram feitas a configuração do Contiki, a configuração do Cooja criando cenários de simulação para testar a rede de sensores em diferentes condições e a familiarização com a alteração dos parâmetros do RPL, que permitem melhorar a eficiência energética e o desempenho da rede.

Uma rede de simulação foi montada no ambiente Cooja com as características identificadas, permitindo a avaliação do consumo de energia dos nós

conforme mencionado anteriormente. A forma como a rede foi organizada é mostrada na Figura 13, junto com a ocorrência da transmissão de uma mensagem. Os resultados da potência consumida de cada nó são apresentados na Figura 12, permitindo se ter uma ideia inicial do funcionamento do Cooja e RPL, além do processo de obtenção de resultados para agir como um guia nas futuras simulações.

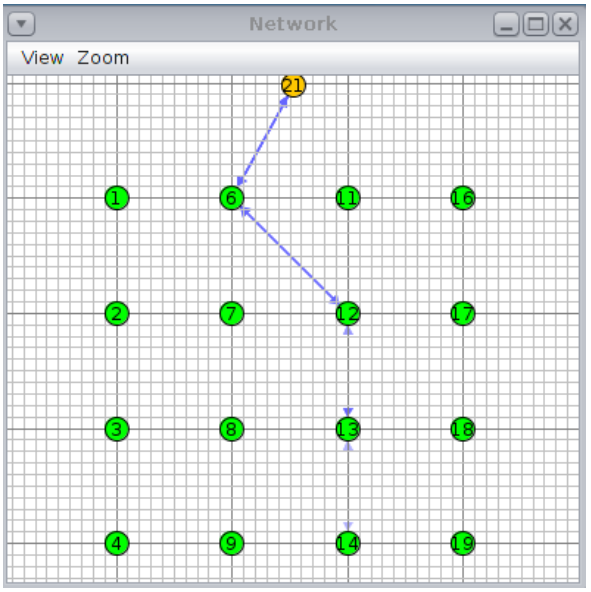

Figura 13 - Configuração da primeira simulação.

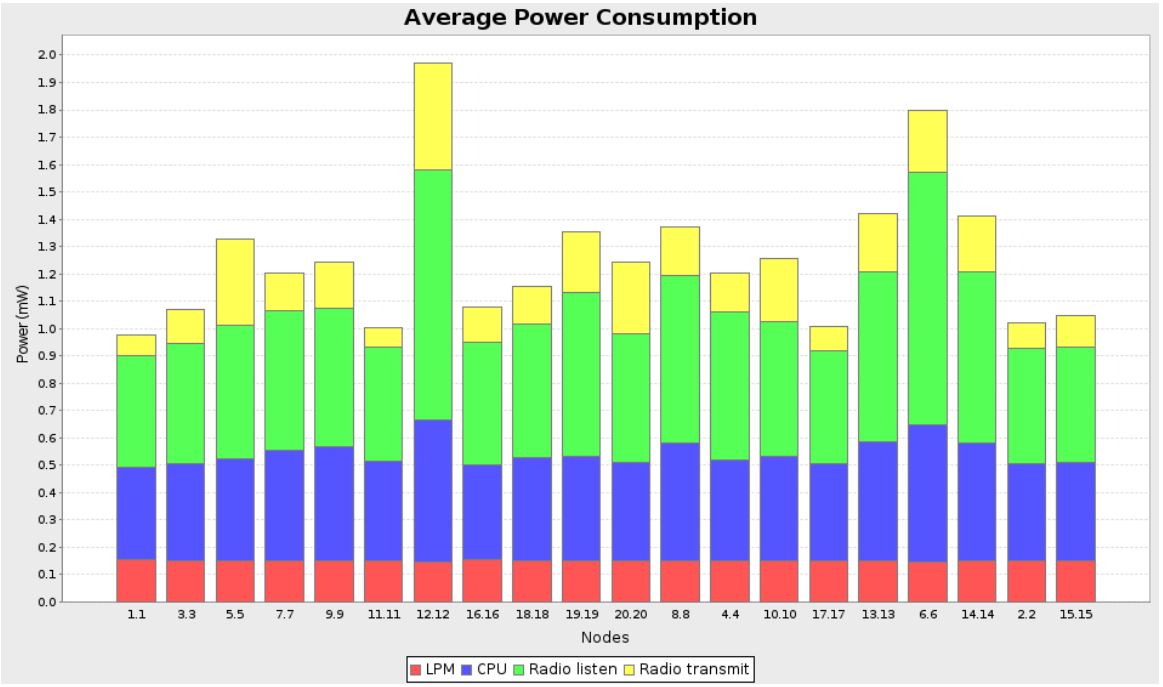

Figura 14 - Teste de obtenção de resultado da primeira simulação.

A Figura 14, representa a potência instantânea média consumida por cada nó da rede, nota-se que o nó 12 consumiu mais potência pois é um nó central que redireciona mais pacotes. Esse resultado foi obtido com o powertrace do Contiki 2.7. Em simulações futuras optou-se por usar o Energest do Contiki-NG.

3.2 DESENVOLVIMENTO DAS SIMULAÇÕES

Após a realização das simulações de familiarização com o emulador Cooja, iniciou-se a investigação acerca do desenvolvimento de aplicativos específicos para os módulos sensores. Esse estudo teve como objetivo principal criar soluções eficientes para coleta e disseminação de dados na rede, levando em consideração as particularidades dos sensores utilizados.

3.2.1 PROCESSOS DE SENSORIAMENTO

Como resultado, foram desenvolvidos no código dos módulos dois processos distintos para os sensores: um dedicado à leitura de valores de pH e outro à medição de umidade, para que a leitura de cada dado tornasse independente, permitindo que no futuro um processo acontecesse com uma frequência diferente do outro. Ambos os sensores foram implementados com foco na coleta periódica de dados e na integração com a estrutura da rede. Além disso, foi elaborado um terceiro processo responsável pela transmissão e recepção de mensagens. Essa parte do código tem como função garantir a comunicação eficiente entre os nós da rede, propagando as informações coletadas pelos sensores e assegurando que os dados sejam disponibilizados para processamento e análise em nós de coleta ou base. A implementação desses processos seguiu os padrões estabelecidos pelo sistema operacional Contiki, configurando e utilizando recursos como os conversores analógicos/digitais para a leitura dos sensores e os protocolos UDP e RPL para a comunicação na rede. Dessa forma, foi possível criar um sistema modular e escalável, apto a operar em redes de sensores com características de baixa potência e custo de energia.

Para efetuar todo o processo de leitura dos sensores, foi utilizado o timer de eventos (etimer). Essa ferramenta permite gerar eventos que podem ser processados dentro do loop de eventos do Contiki e foi desenvolvido para sistemas com limitação de memória e de potência de processamento. O etimer é amplamente utilizado em aplicações IoT que coletam dados periodicamente de sensores e permite que o sistema entre em estados de baixa energia entre eventos, retomando as ações apenas quando necessário e impactando diretamente na economia de energia. O etimer utiliza o clock do sistema operacional, cuja resolução depende da implementação do hardware e da configuração do próprio Contiki. A unidade básica de tempo é chamada de "*ticks*" e pode ser convertida para segundos.

A aplicação com os sensores de pH e umidade com monitoramento do consumo de energia no sistema Contiki foi feita de acordo com a seguinte lógica: Foram definidos os intervalos dos temporizadores, indicando a frequência de leitura dos sensores e de envio de dados ideais com base na literatura:

- A cada 10 minutos para o sensor de umidade.
- A cada 20 minutos para o sensor de pH.
- A cada 20 minutos para transmissão da mensagem contendo os dados amostrados pelos sensores via protocolo UDP.

A utilização dos sensores na aplicação só foi possível através da implementação de um driver que permite ativá-los, desativá-los, configurá-los e acessar todas as leituras desses dispositivos que estão conectados na placa Zolertia Z1 [22]. Essa plataforma é específica para redes de sensores sem fio e consiste de um microcontrolador de baixa potência, um transceptor conforme o padrão IEE 802.15.4 e sensores digitais integrados. Outros dispositivos também fazem parte da plataforma, mas não foram utilizados no projeto.

3.2.2 MEDIÇÃO DO CONSUMO DE ENERGIA

Com os módulos sensores e o processo de comunicação desenvolvidos, o próximo passo consistiu na implementação de um código voltado exclusivamente para a coleta de dados sobre o consumo energético dos nós da rede.

A medição de energia do sistema foi feita por meio da implementação do módulo Energest, uma ferramenta utilizada em aplicações voltadas ao gerenciamento de energia e que possibilita tanto o monitoramento como a otimização do consumo energético em determinado ambiente. Este mecanismo mede o tempo em que diferentes dispositivos do sistema passam em diferentes estados de energia, fornecendo uma visão detalhada do consumo energético durante a execução de um programa, permitindo identificar componentes ou funções que consomem mais energia e ajudando a otimizar o design do software ou do hardware.

O Energest é baseado na contagem de tempo para cada estado dos componentes e utiliza os "*ticks"* do *clock* do sistema para medir quanto tempo os componentes permanecem em estados específicos. Isso possibilita efetuar uma estimativa do consumo de energia. O tempo em "*ticks*" medido pelo Energest pode ser convertido em consumo de energia real, caso sejam conhecidos os consumos por estado do hardware em questão e todo esse processo é detalhado na sequência do trabalho.

Há cinco modos de consumo do Energest predefinidos que todas as plataformas Contiki devem suportar e estão representados na Tabela 2:

Tabela 2 - Modos de consumo do Energest.

Modos de Consumo	**Propósito**
ENERGEST_TYPE_CPU	A CPU está ativa.
ENERGEST_TYPE_LPM	A CPU está no modo de baixo ser medidas pelos sensores para avaliar a qualidade do solo e se é necessário tomar alguma medida para preservar e aumentar a produção consumo de energia.
ENERGEST_TYPE_DEEP_LPM	A CPU está em modo de baixo consumo de energia.
ENERGEST_TYPE_TRANSMIT	O rádio está transmitindo.
ENERGEST_TYPE_LISTEN	O rádio está escutando.

Os três primeiros são usados para rastrear os diferentes modos da CPU. Nem todas as CPUs suportam LPM e deep LPM e, nesse caso, o tipo não utilizado sempre relatará 0 como tempo. Os dois últimos modos são usados para rastrear quando o rádio está ligado ou em algum estado. A coleta de dados é feita de forma passiva, portanto não afeta o funcionamento padrão do programa e todos os tempos acumulados podem ser acessados usando as funções fornecidas pela API do Energest, apresentada na Tabela 3:

Tabela 3 - Funções fornecidas pela API do Energest.

Função	Propósito
ENERGEST_TIME_T ENERGEST_CURRENT_TIME()	Obtenha a hora do sistema a partir da fonte de hora Energest.
ENERGEST_SECOND	O número de tiques por segundo da fonte de tempo Energest.
ENERGEST_ON(type)	Defina o tipo especificado como ativado e comece a monitorar o tempo.
ENERGEST_OFF(type)	Defina o tipo especificado como desativado e atualize o tempo total.
ENERGEST_SWITCH (type_off, type_on)	Alterne do tempo de rastreamento do primeiro tipo para o segundo tipo.
ENERGEST_GET_TOTAL_TIME()	Obtenha o tempo total que o Energest está monitorando.
void energest_flush(void)	Descarregue os tempos do Energest para todos os componentes que estão ligados no momento.
uint64_t energest_type_time (energest_type_t type)	Leia o tempo total em que o tipo especificado ficou ligado.
void energest_init(void)	Inicialize o módulo Energest.

A macro ENERGEST_CURRENT_TIME retorna um tempo de sistema dependente da plataforma para uso pelo módulo Energest e a constante ENERGEST_SECOND especifica o número de “*ticks*” por segundo.

As macros ENERGEST_ON, ENERGEST_OFF, e ENERGEST_SWITCH são usadas para informar ao módulo Energest quando um componente muda de modo e normalmente são chamadas apenas por plataformas e *drivers* de dispositivo. Quando um componente muda de um modo para outro modo, é recomendado usar ENERGEST_SWITCH, pois fornece melhor precisão.

A macro ENERGEST_GET_TOTAL_TIME() retorna o tempo total que o módulo Energest vem rastreando. Isso normalmente é o mesmo, mas pode ser substituído por plataformas que fornecem modos de consumo adicionais do Energest.

A função energest_flush atualiza o tempo total para todos os modos de consumo que estão ativados no momento.

A função energest_type_time retorna o tempo total em que o tipo especificado ficou ativado.

A função energy_flush deve ser chamada antes de ler o tempo total para garantir que o tempo total foi atualizado.

A função energest_init() é chamada pelo sistema durante o procedimento de inicialização para inicializar o módulo Energest.

O código foi estruturado para capturar os dados fornecidos pelo Energest em intervalos regulares, registrando o tempo acumulado em cada estado. Esses dados foram então utilizados para calcular o consumo energético estimado, com base nas características específicas de consumo de energia do hardware utilizado (como tensões e correntes típicas dos estados). Os valores coletados incluíram:

- Tempo de CPU ativa: representa o tempo em que o nó está executando operações.

- Tempo de LPM: indica o período em que o nó está em modo de baixa potência, consumindo menos energia.
- Tempo de transmissão: contabiliza o tempo gasto enviando mensagens.
- Tempo de recepção: mede o tempo durante o qual o nó está escutando ou recebendo mensagens.

O objetivo principal dessa etapa foi coletar dados precisos sobre o comportamento energético dos nós, sem aplicar otimizações ou ajustes. Os dados obtidos serviram como base para análises futuras, proporcionando uma visão mais clara do perfil de consumo energético da rede em diferentes cenários operacionais.

De modo a realizar o controle e monitoramento de energia, foi definido um valor limite da energia da bateria e executado um cálculo da energia recuperada baseada no tempo decorrido desde a última atualização. Todos esses valores são visualizados através de um log gerado pelo sistema. O consumo de energia em Joules é resultado do produto do tempo pela corrente e a tensão do sistema:

$$E = I * t * V$$

A função update_energy_usage foi desenvolvida para atualizar o consumo energético dos diferentes estados do sistema, obter os tempos acumulados em cada estado através do módulo Energest, deduzir a energia consumida da bateria e atualizar o total de energia utilizada.

3.2.3 COMBINAÇÃO DAS FUNCIONALIDADES

Para o funcionamento das simulações é necessário que o código dos módulos sensores seja capaz de quantificar o gasto de energia, transmitir os dados amostrados e interromper o funcionamento quando sua bateria acabar. Dessa forma, os processos dos sensores foram implementados com a mesma lógica de funcionamento anterior, mas considerando as diferenças intrínsecas de cada componente no sistema (tempo de leitura, etc.) e implementando o consumo de energia:

- Leitura periódica dos valores dos sensores em função do etimer:

    - Espera o temporizador expirar, ativa os sensores e converte os valores lidos para um formato legível.
- Cálculo do consumo energético do sensor durante a sua operação:
    - Mede o tempo ativo e efetua o cálculo do consumo de energia considerando o gasto de energia do sensor.
- Atualização dos valores consumidos de energia da bateria:
    - Deduz a energia consumida da bateria e atualiza o total da energia utilizada.
- Visualização dos dados dos sensores e da bateria:
    - Apresenta os valores lidos pelos sensores e o estado da bateria.
- Ajuste para o consumo da bateria quando agindo como roteador.
- Implementação da recuperação de energia.

O código desenvolvido permite a integração dos sensores no sistema. Os componentes são ativados e desativados de acordo com a demanda, as leituras dos sensores são obtidas usando o conversor ADC e as funções controlam o estado dos conversores de modo a economizar energia. Na sequência, os dados coletados dos sensores são enviados via protocolo UDP e parte do processo consiste na verificação da energia restante, de modo a avaliar se é suficiente antes de realizar uma transmissão e enviar os dados para o endereço IP caso o dispositivo seja capaz de alcançar a raiz da rede. A função udp_rx_callback é chamada sempre que uma mensagem UDP é recebida e atualiza o contador de mensagens, além de apresentar o conteúdo recebido, durante sua chamada também é calculado o gasto de energia, para quando o nó sensor envia um pacote que originou de outro nó.

3.2.3 VALIDAÇÃO DA SIMULAÇÃO

Com o código desenvolvido, a etapa seguinte consistiu na validação do aplicativo por meio de simulações no emulador Cooja. Essa fase teve como objetivo garantir que todas as funcionalidades implementadas estavam operando corretamente, bem como avaliar o desempenho da rede em diferentes cenários. Foram realizados testes para verificar a transmissão e recepção de mensagens, validar o cálculo da taxa de entrega de pacotes e amostragem de retransmissões e analisar o consumo e a recuperação de energia. Os primeiros testes realizados

foram focados na transmissão e recepção de mensagens, bem como na validação da taxa de transmissão de pacotes (PDR - *Packet Delivery Ratio)*. Ambos os aspectos foram avaliados simultaneamente, uma vez que estão diretamente relacionados ao desempenho da comunicação na rede.

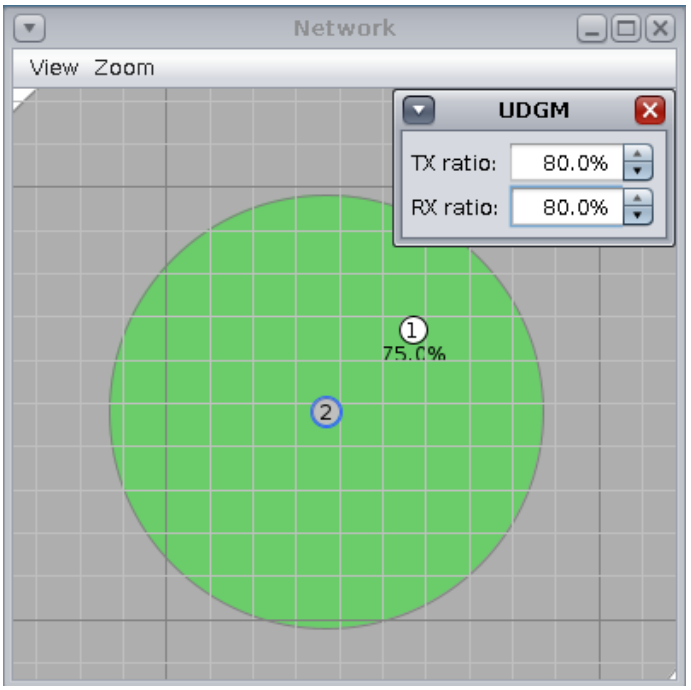

Figura 15 - Probabilidade de Entrega de Pacotes

A Figura 15, mostra o experimento de simulação usando o algoritmo UDGM (*Urban-Scale Deployment of Greedy Multipath Routing*) implementado pelo Cooja. Com o objetivo de testar o cálculo da taxa de entrega de pacotes, foram parametrizados TX [80%] e RX [80%]. Esses valores indicam as probabilidades de sucesso de transmissão (TX) e recepção (RX) para cada pacote de dados e são definidos na simulação. Na Figura 15, é possível também verificar uma taxa de 75% ao lado do nó 1. Essa é a probabilidade do nó 2 enviar com sucesso um pacote para o nó 1 (sem considerar sua resposta), da mesma forma, ao nó 1 enviar uma mensagem a onda percorre pelo mesmo caminho, então a probabilidade de envio é a mesma de 75%. Dessa forma, a probabilidade de entrega de pacotes teóricos é de 56,25% entre servidor e módulos. Na Figura 16, os valores obtidos para a taxa de entrega de pacotes da simulação podem ser vistos.

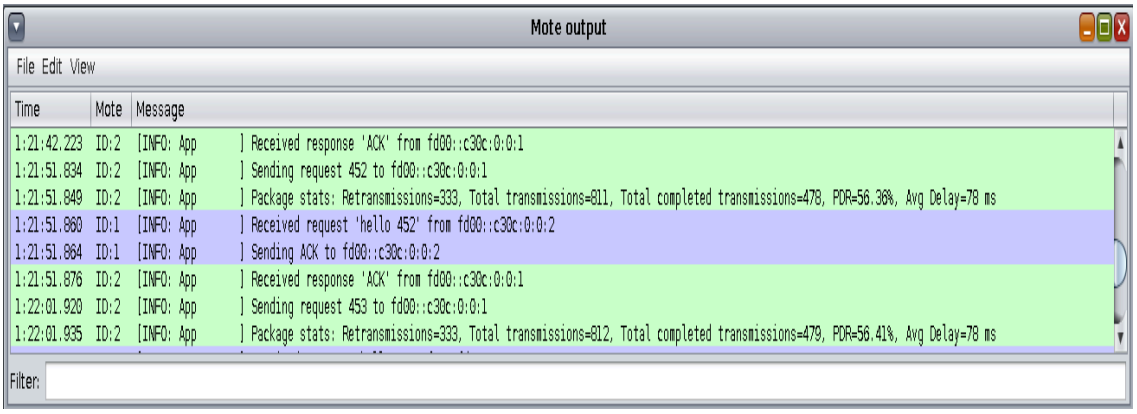

Figura 16 - Simulação de Entrega de Pacotes

Com base nos resultados fornecidos pela simulação de entrega de pacotes (PDR – *Packet Delivery Ratio*) no emulador Cooja, é possível realizar uma análise detalhada dos principais indicadores relacionados à transmissão de pacotes e ao desempenho da rede. A simulação registrou um total de 453 solicitações de envio de pacotes. Durante os testes, foram observadas 333 retransmissões, indicando que uma proporção significativa de pacotes precisou ser reenviada devido a falhas na entrega inicial. Isso resultou em um total de 811 transmissões, somando os pacotes enviados inicialmente e os retransmitidos. O total de transmissões concluídas, incluindo tentativas iniciais e retransmissões, foi de 478, maior que o número de 453 solicitações, porque o cálculo dessa quantidade é feito na camada de enlace (MAC) da rede, considerando então pacotes de controle. O cálculo do PDR, que reflete a eficiência da rede na entrega de pacotes, resultou em uma taxa de 56,36%, valor próximo à probabilidade teórica, a diferença se deve à aleatoriedade do processo.

Além disso, o tempo médio de atraso, medido em 78 milissegundos, indica o intervalo entre o envio de um pacote e sua entrega bem-sucedida, esse valor considera também o tempo das retransmissões necessárias para o envio.

Em sequência, foi validado a amostragem dos sensores, o consumo das baterias e a recuperação de energia, os valores podem ser vistos na Figura 17:

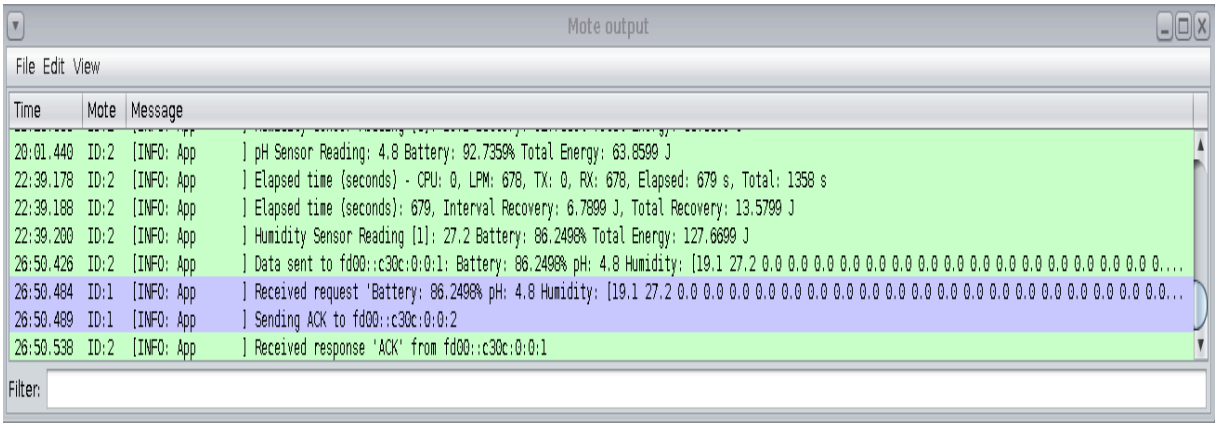

Figura 17 - Funcionamento do Módulo Sensor.

Em resumo, os dados obtidos foram:

- LPM = 678 s; TX=0 s; RX = 678 s.
- Elapsed = 679 s; Total = 1358 s.
- Battery = 86,24%.
- Total Energy = 127,6699 J em 22:39:200.
- Total recovery: 13,5799 J.
- Interval recovery: 6,7899 J.
- pH sensor reading: 4,8.
- Humidity Sensor reading [1]: 27,2%.

Com esse valores, é possível observar que o microcontrolador sempre esteve com o rádio ligado escutando durante o intervalo e esteve em modo de baixo consumo de energia (LPM e RX). Manter o rádio sempre ligado foi uma opção para evitar perda de pacotes que pode ser avaliada como uma possível mudança para a economia de energia. Além disso, é informado o tempo passado desde a última amostragem de dados (*Elapsed*) e o tempo total passado. É apresentado também a aquisição de dados dos sensores. Outros parâmetros são a bateria restante, o total de energia recuperada e o total recuperado do intervalo, valores coerentes, pois o total recuperado foi de 13,5799 J em 1358 s, assumindo uma potência recebida de 0,01 W. Por fim, o último dado relatado foi a energia total gasta pelo módulo de 127,6699 J. O que é coerente. pois para essa simulação foi considerado uma tensão

de 5 V e como o tempo é conhecido, é possível calcular a corrente média em amperes (mA) :

$$I = E / (t * V)$$

Onde, I é a corrente média consumida, E a energia consumida, t o tempo decorrido e V a tensão de operação do microcontrolador. Substituídos os valores, obtém-se que, em média, o sistema consumiu 18,788 mA durante a operação em tensão de 5 V. Conforme a Figura 10, verifica-se que o valor é coerente, pois como o rádio majoritariamente está ligado escutando o meio, o consumo deveria ser próximo do valor do *datasheet* do módulo usado, que é o que se verifica, a diferença nos valores se justifica pelo arredondamento dos valores na amostragem das variáveis.

4 RESULTADOS FUNDAMENTAIS E OBTIDOS

O desenvolvimento de um sistema de rede de sensores para monitoramento da qualidade do solo em cultivos de soja, com energia carregada exclusivamente por ondas eletromagnéticas, deve alcançar resultados fundamentais que assegurem a viabilidade e eficácia do sistema proposto. Estes resultados são essenciais para garantir que os objetivos do projeto sejam atingidos de forma eficiente e prática. Os resultados fundamentais esperados são:

1. IMPLEMENTAÇÃO DA APLICAÇÃO NO CONTIKI E SERVIDOR DE COLETA DE DADOS:
    - Desenvolver a aplicação de monitoramento utilizando o sistema operacional Contiki, e implementar um servidor para a coleta e armazenamento dos dados adquiridos. Este resultado deve incluir a configuração da rede para suportar o envio de dados sobre a qualidade do solo e outros parâmetros críticos.
2. DESENVOLVIMENTO DO MÓDULO DE COMUNICAÇÃO DA REDE NO SIMULADOR:
    - Projetar um módulo de comunicação robusto para a rede de sensores, garantindo a transmissão eficiente dos dados coletados para o

servidor, com atenção especial à integridade e consistência dos dados durante o processo de comunicação.

3. MINIMIZAÇÃO DO CONSUMO DE ENERGIA NO SISTEMA:
    - Implementar modificações na configuração do protocolo RPL (*Routing Protocol for Low-Power and Lossy Networks*) para reduzir o consumo de energia dos dispositivos de monitoramento. Este resultado deve demonstrar uma redução significativa no consumo de energia, permitindo que o sistema opere com a energia captada exclusivamente das ondas eletromagnéticas presentes no ambiente.
4. SIMULAÇÃO E AVALIAÇÃO DO SISTEMA NO AMBIENTE COOJA:
    - Realizar simulações detalhadas utilizando o ambiente Cooja para avaliar o desempenho do sistema de monitoramento, incluindo a coleta de dados de consumo de energia, atraso, perda e *jitter* na comunicação. Este resultado deve fornecer uma análise compreensiva sobre a eficiência energética e a viabilidade operacional do sistema em condições reais.

O desenvolvimento do sistema apresentou avanços significativos, mas também enfrentou desafios que limitaram a obtenção de alguns resultados esperados. A implementação da aplicação no sistema operacional Contiki e a criação de um servidor de coleta de dados foram realizadas com sucesso, permitindo a coleta e o armazenamento dos dados transmitidos pelos sensores. Além disso, foi desenvolvido um módulo de comunicação eficiente para a rede na simulação, garantindo a integridade e consistência da transmissão dos dados.

No entanto, a simulação e avaliação do sistema no ambiente Cooja foram apenas parcialmente concluídas devido a restrições relacionadas ao tempo necessário para as simulações e às limitações de memória do simulador e computador. Houveram também dificuldades ao longo do trabalho devido a mudança de versão do Contiki 2.7 para o Contiki-NG, no qual algumas chamadas de funções importantes para o funcionamento da aplicação sofreram modificações ou foram removidas, gerando um trabalho de atualização de chamadas de funções e criações de novas funções. Para contornar essas dificuldades computacionais, foram realizadas adaptações como a utilização de uma bateria com capacidade reduzida

em 100 vezes e um timer ajustado para ser 100 vezes mais rápido, o que permitiu análises preliminares, mas não viabilizou testes completos. Com parâmetros de bateria de 879,12 J, temporizadores de 20 minutos para o envio de mensagens e amostragem de pH, 10 minutos para amostragem de umidade, recuperação de 0,001 W, tensão de 3,3 V e 4 horas de simulação, foi possível executar algumas das simulações planejadas no plano, com duas amostras de cada simulação da função MRHOF ETX e uma amostra de cada simulação da OF0 e de cada combinação parâmetros de Trickle.

A minimização do consumo de energia, embora tenha sido objeto de estudo, não pôde ser implementada de forma plena devido às limitações nas simulações e no tempo disponível para o desenvolvimento. Ainda assim, identificou-se que o consumo energético elevado do rádio é um desafio crítico para a operação sustentável do sistema. Com relação à viabilidade de um balanço energético positivo, não foi possível demonstrar conclusivamente que a energia captada seria suficiente para manter o sistema em funcionamento contínuo. As análises preliminares indicam que a demanda energética do rádio tornaria improvável a operação autônoma do sistema sem fontes suplementares de energia. Dessa maneira, seguindo os objetivos fundamentais do trabalho e o plano de simulações estruturado. Foram iniciadas as simulações, com duas amostras de cada teste para a MRHOF ETX, e uma amostra para as outras simulações executadas. Apenas parte do plano de simulações foi concluído e não foi possível fazê-lo com o número de amostras desejadas de 5 para cada combinação de parâmetros. Do plano de simulações foram possíveis executar as simulações demonstradas nas Figuras 18, 19 e 20.

Parâmetros			MRHOF ETX				
Alcance do rádio (m)	Distância entre nós	Número de nós	Consumo de enegia (J)	Energia restante dos nós (%)	Número de retransmissões	PDR(%)	Delay (ms)
50	5	20	834,44780 ± 47,71430	8,3703 ± 0,07381	92,875 ± 111,35119	97,32125 ± 2,57698	Sem medidas <1
	25	20	887,40710 ± 142,60587	0,54421 ± 0,06458	294,87500 ± 741,28075	95,36375 ± 3,65868	0,25600 ± 0,15271

Figura 18 - Resultados obtidos MRHOF ETX

Parâmetros			OF0				
Alcance do rádio (m)	Distância entre nós (m)	Número de Nós	Consumo de enegia (J)	Energia restante dos nós (%)	Número de retransmissões	PDR(%)	Delay (ms)
50	25	10	801,47235	8,83243	6,80000	99,32250	2,75000

Figura 19 - Resultados obtidos OF0

Parâmetros		Resultados Trickle				
DIO INTERVAL	INTERVAL DOUBLINGS	Consumo de enegia (J)	Energia restante dos nós (%)	Número de retransmissões	PDR(%)	Delay (ms)
12	8	809,63390	9,47380	25,00000	98,96500	1,00000
13	8	814,34759	9,17595	34,50000	99,05750	1,25000
14	8	798,18553	9,20630	47,25000	98,79750	0,75000
15	10	794,45411	9,63075	51,00000	98,33000	1,00000

Figura 20 - Resultados obtidos Trickle timer

É possível perceber a partir da Figura 18 que quase toda a bateria foi consumida na duração das 4 horas de simulação, a recuperação também não foi o suficiente para a volta do sistema, de modo que todo o sistema seja desligado na sequência das simulações. Uma tendência percebida na simulação, foi que todos o nós sensores desligavam simultaneamente, mesmo aqueles que transmitiam mais pacotes por agirem como rotear, desligavam na mesma ordem de tempo que os outros nós da rede. Avaliando a simulação com distância de 25 m entre nós é possível perceber que a taxa de entrega de pacotes diminuiu em relação à distância de 5m e a energia restante dos nós também. Para o delay (atraso do envio de pacotes) o tempo foi contado em milissegundos, com a resolução de 1 milisegundo, então qualquer valor menor, foi truncado pela variável inteira e atribuído 0, por isso o valor na simulação de 5m de espaçamento, mas também nota-se um aumento do atraso para espaçamento de 25 m o que se justifica pelo aumento da distância percorrida pela onda de rádio. Assim como o número de retransmissões que aumentou significativamente também, devido às perdas no caminho da onda.

No que diz respeito a OF0, nota-se na Figura 19 que o número de retransmissões, taxa de entrega de pacotes e atraso aumentaram. Isso se deve a OF0 tender a fazer menos transmissões que a MRHOF ETX, já que visa diminuir o número de saltos, com um número menor de transmissões, a taxa de entrega aumenta e o número de retransmissões também, pois há menor chance de um pacote ser perdido durante o processo de envio ao nó raiz. Sobre o aumento do atraso, isso é o esperado pois a ETX visa minimizar esse parâmetro enquanto a OF0 o desconsidera no seu roteamento. Além disso, a OF0 apresentou um menor consumo de energia que a MRHOF ETX, provavelmente devido às diferenças no número de nós da simulação.

Nesse viés, também foram executadas simulações com os parâmetros de Trickle. A partir do que é mostrado na Figura 20, é possível verificar que com o aumento do DIO Interval a energia restante média dos nós tende a diminuir e se estabilizar para valores maiores. Porém, com o aumento do Interval Doublings verifica-se um aumento da energia restante dos nós. Além disso, verifica-se um aumento do número de retransmissões conforme DIO interval e DIO doublings aumentaram, isso se deve a dificuldade da rede de redescobrir as rotas após um grande período de tempo sem novas informações acerca dela. Finalmente, tanto o atraso quanto a taxa de entrega de pacotes não parecem ser afetados pelos parâmetros de Trickle durante as simulações.

Finalmente, para a análise da máxima distância possível da fonte de emissão eletromagnética de maneira que o sistema ainda possua alimentação suficiente foram analisados o consumo de potência dos nós e a recuperação de energia necessária. Dessa forma, de acordo com o Ato nº 915, de 01 de fevereiro de 2024, publicado pela ANATEL [23], as máximas potências transmitidas permitidas para uma estação rádio base operando nas frequências de 940 MHz, 1,95 GHz e 2,44 GHz, são de 24 dBm (0,25119 W), 33 dBm (1,9953 W) e 40 dBm (10 W), respectivamente. Assim, utilizando essa informação e a equação de propagação de Friis, é possível calcular a distância da estação rádio base dada uma certa potência recebida. Essa informação é importante para definir a plausibilidade de aplicação do sistema e determinar a que distância as fontes de emissão podem estar, pois com as simulações foi possível determinar a potência recebida pelos coletores de energia para que o gasto energético e o desligamento dos módulos não comprometam o funcionamento da rede. Assim, conhecendo a potência recebida e assumindo uma potência transmitida como o máximo permitido pela Anatel, pode-se calcular as distâncias da seguinte forma:

$$Pr = Pt.Gt.Gr.(\lambda/4\pi d)^2 \quad [\text{W}]$$

Onde:

- Pr: Potência recebida no receptor.
- Pt: Potência transmitida pela antena do transmissor.
- Gt: Ganho real da antena transmissora.

- Gr: Ganho real da antena receptora.
- λ: Comprimento de onda do sinal (λ = c/f).
- d: Distância entre as antenas.
- c: Velocidade da luz ($3x10^8$ m/s).
- f: Frequência da portadora.

Rearranjando a equação:

$$d = (c/4\pi f).\sqrt{Pt.Gt.Gr/Pr} \quad [m]$$

Efetuando uma análise conservadora, desconsiderando que um nó poderia ficar desligado, foi possível calcular a potência média consumida em um único nó, de modo a obter a máxima distância da fonte de emissão de ondas eletromagnéticas para que toda a energia consumida fosse totalmente recuperada.

Durante as simulações, a cada momento em que uma mensagem era enviada, os valores de energia consumida e de bateria restante eram informados para todos os nós. Desta forma, foi obtido o consumo médio de um nó em um determinado intervalo de tempo, calculando a variação do consumo energético. Efetuando a divisão deste resultado pela média do intervalo de tempo, conseguimos obter o valor necessário da potência recebida: Pr = 0,03113 W. Proporcionalmente à máxima potência transmitida permitida pela Anatel, tem-se que para cada faixa de frequência a contribuição de cada fonte para a potência recebida deve ser:

- 940 MHz: Pr = 6,39 μW
- 1,95 GHz: Pr = 5,07 mW
- 2,44 GHz: Pr = 25,4 mW

Por fim, de posse de todos os dados necessários, foi possível calcular as máximas distâncias para as três faixas de frequência, considerando que as respectivas fontes estão transmitindo simultaneamente:

- 940 MHz: d = 5,21 m
- 1,95 GHz: d = 0,32 m
- 2,44 GHz: d = 0,29 m

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho aponta alternativas para avaliação de redes de sensores autossustentáveis através de simulações no ambiente Cooja. A implementação funcional da aplicação no sistema operacional Contiki e do módulo de comunicação comprovou a viabilidade desta estratégia para avaliação de desempenho de redes, de sensores auto sustentáveis no cenário de aplicações para agricultura de precisão. Esses resultados destacam o potencial de tais sistemas em aplicações reais, reforçando a relevância do uso de tecnologias que conciliem desempenho e eficiência energética.

As análises realizadas permitiram explorar o consumo energético dos módulos e identificar desafios associados à operação dos sistemas propostos. Entre os principais obstáculos, destaca-se o elevado custo energético do rádio, que constitui um dos fatores mais críticos em sistemas baseados em redes de sensores. Embora as restrições do ambiente de simulação limitem a possibilidade de realizar análises completamente fiéis à realidade, os dados coletados forneceram insights valiosos sobre o comportamento dos módulos em cenários simulados. Isso permitiu delinear tendências que podem servir de base para futuras investigações e otimizações.

Os desafios enfrentados durante o projeto também apontaram para diversas oportunidades de melhoria, especialmente no que diz respeito à maneira que as simulações foram realizadas. Investigações futuras poderiam incluir ajustes no código da simulação, como a possibilidade de representar baterias e temporizadores em escalas mais próximas das condições reais. Isso viabilizaria a realização de testes mais precisos, permitindo avaliar diretamente os parâmetros de configuração do protocolo RPL, incluindo o algoritmo Trickle e diferentes funções objetivo.

Além das melhorias no ambiente de simulação, há um amplo campo para o desenvolvimento de estratégias que otimizem o desempenho energético dos dispositivos. Técnicas como o desligamento temporário do rádio em períodos de inatividade, configurações mais avançadas de duty cycling, a implementação de algoritmos de wake-up rádio e a integração de sensores de menor consumo

energético representam caminhos promissores para reduzir o consumo energético e prolongar a vida útil dos dispositivos. Essas abordagens podem ser exploradas para adequar os sistemas desenvolvidos às demandas de aplicações práticas.

A continuação deste trabalho envolve:

1. medições de consumo de corrente em hardware real para calibrar a simulação.
2. Realizar a variação sistemática dos parâmetros do algoritmo de Trickle para determinar a melhor configuração para a aplicação de sensoriamento na agricultura de precisão.
3. Realizar a variação sistemática na função objetivo e métricas de desempenho.
4. Calcular os erros para o intervalo de confiança desejado, de forma a permitir a generalização de resultados.

Por fim, este trabalho ressalta a importância de continuar investigando soluções que integrem tecnologias inovadoras para reduzir o consumo energético e maximizar a eficiência dos sistemas de monitoramento.

REFERÊNCIAS

[1] Pretto, J. E.; et al. ACOMPANHAMENTO DA SAFRA BRASILEIRA GRÃOS 5º LEVANTAMENTO SAFRA 2023/24. Companhia Nacional de Abastecimento. Vol. 11, no. 5. ISSN: 2318-6852. Fevereiro, 2024.

[2] França-Neto, J. B.; Krzyzanowski, F. C.; Henning, A. A.; Pádua, G. P.; Lorini, I.; Henning, F. A. (Editores Técnicos). Tecnologia da produção de semente de soja de alta qualidade. Embrapa Soja, Londrina, PR, 2016. Documentos 380. Dezembro, 2016.

[3] Tibola, C. S.; Medeiros, E. P.; Simeone, M. L. F.; Oliveira, M. A. (Editores Técnicos). Espectroscopia no Infravermelho Próximo para Avaliar Indicadores de Qualidade Tecnológica e Contaminantes em Grãos. Brasília, DF: Embrapa, 2018.

[4] Winter, T, et al. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Internet Engineering Task Force (IETF). RFC655. Março, 2012.

[5] Dragoman, M.; Aldrigo, M.; Dinescu, A.; Vasilache, D.; Iordanescu, S.; Dragoman, D. (2023). Nanomaterials and Devices for Harvesting Ambient Electromagnetic Waves. Nanomaterials MDPI, vol. 13, página 595.

[6] Almoneef, T. S.; Erkmen, F.; Ramahi, O. M. (2017). Harvesting the Energy of MultiPolarized Electromagnetic Waves. Sci Rep vol. 7, no.14656.

[7] IEEE Standard for Low-Rate Wireless Networks Corrigendum 1: Correction of Errors Preventing Backward Compatibility, em IEEE Std 802.15.4-2020/Cor 1-2022 (Corrigendum to IEEE Std 802.15.4-2020 conforme emendado por IEEE Std 802.15.4z-2020, IEEE Std 802.15.4w-2020, IEEE Std 802.15.4y-2021, e IEEE Std 802.15.4aa-2022) , vol., no., pp.1-22, 10 Jan. 2023.

[8] Barros, G. S. C.; et al. PIB DO AGRONEGÓCIO CAI NO TERCEIRO TRIMESTRE E ACUMULA BAIXA DE 0,91% EM 2023. Centro de Estudos Avançados em Economia Aplicada, Confederação da Agricultura e Pecuária do Brasil. Dezembro 2023.

[9] George Oikonomou, Simon Duquennoy, Atis Elsts, Joakim Eriksson, Yasuyuki Tanaka, Nicolas Tsiftes, “The Contiki-NG open source operating system for next generation IoT devices”, SoftwareX, 18, 2022.

[10] FARIAS, J. R. B. et al. Ecofisiologia da Soja. Londrina: Embrapa CNPSO, 2007. 9p. (Circular Técnica, N° 48).

[11] EMBRAPA. Tecnologias de Produção de Soja: Região Central do Brasil 2004. Sistemas de Produção, n. 4. Londrina: Embrapa Soja, 2003.

[12] HUNGRIA, Mariangela; CAMPO, Rubens José; MENDES, I. de C. Fixação biológica do nitrogênio na cultura da soja. Londrina: Embrapa Soja, 2001. 48p. (Embrapa Soja. Circular Técnica, 35; Embrapa Cerrados. Circular Técnica 13.

[13] KIM, Hak-Jin; SUDDUTH, Kenneth A.; HUMMEL, John W. Soil macronutrient sensing for precision agriculture. **Journal of Environmental Monitoring**, v. 11, n. 10, p. 1810-1824, 2009.

[14] ADAMCHUK, Viacheslav I. et al. On-the-go soil sensors for precision agriculture. **Computers and electronics in agriculture**, v. 44, n. 1, p. 71-91, 2004.

[15] ROSSEL, RA Viscarra et al. Proximal soil sensing: An effective approach for soil measurements in space and time. **Advances in agronomy**, v. 113, p. 243-291, 2011.

[16] ZHANG, Zhuohui. Investigation of wireless sensor networks for precision agriculture. In: **2004 ASAE Annual Meeting**. American Society of Agricultural and Biological Engineers, 2004. p. 1.

[17] GRISSO, Robert D. et al. Precision farming tools. soil electrical conductivity. 2005.

[18] C. T. Kone, A. Hafid and M. Boushaba, "Performance Management of IEEE 802.15.4 Wireless Sensor Network for Precision Agriculture," in *IEEE Sensors Journal*, vol. 15, no. 10, pp. 5734-5747, Oct. 2015.

[19] TSVETKOV, Tsvetko. RPL: IPv6 routing protocol for low power and lossy networks. **Sensor nodes–operation, network and application**, v. 59, n. 2, 2011.

[20] LEVIS, Philip et al. **The trickle algorithm**. Request for Comments: 6206, Internet Engineering Task Force (IETF), 2011.

[21] PHAM, Binh L.; PHAM, Anh-Vu. Triple bands antenna and high efficiency rectifier design for RF energy harvesting at 900, 1900 and 2400 MHz. In: **2013 IEEE MTT-S International Microwave Symposium Digest (MTT)**. IEEE, 2013. p. 1-3.

[22] **ZOLERTIA.** *Z1 Hardware: datasheet revision C.* [S.l.]: Zolertia, 2010. Disponível em: https://issuu.com/zolertia/docs/z1_datasheet. Acesso em: 2 dez. 2024.

[23] **ANATEL.** *Ato nº 915, de 01 de fevereiro de 2024: requisitos técnicos e operacionais para uso das faixas de rádiofrequências associadas ao serviço limitado privado.* Brasília, DF: Anatel, 2024.Disponível em: https://informacoes.anatel.gov.br/legislacao/atos-de-requisitos-tecnicos-de-gestao-do-espectro/2024/1920-ato-915. Acesso em: 2 dez. 2024.

APÊNDICE A - CÓDIGO DO NÓ SENSOR

```
#include <stdio.h>
#include "contiki.h"
#include "sys/energest.h"
#include "sys/clock.h"
#include "os/lib/sensors.h"
#include "z1-phidgets.h"
#include "net/routing/routing.h"
#include "net/netstack.h"
#include "net/ipv6/simple-udp.h"
#include <stdint.h>
#include <inttypes.h>
#include "sys/log.h"
#include <stdio.h>
#include <string.h>
#include "lib/random.h"
#include "net/mac/csma/csma-output.h"

#define VOLTAGE 3.3

// Energy Consumption Constants
#define CURRENT_CPU_ACTIVE 0.01        // 10 mA
#define CURRENT_CPU_LOW_POWER 0.000023 // 23 uA
#define CURRENT_RADIO_RX 0.0188        // 18.8 mA
#define CURRENT_RADIO_TX 0.0174        // 17.4 mA

#define CURRENT_PH_SENSOR 0.01         // 10 mA
#define CURRENT_HUMIDITY_SENSOR 0.035  // 35 mA

#define BATTERY 879.12
#define ENERGY_RECOVERY_PER_SECOND 0.001

// UDP Configuration
#define LOG_MODULE "App"
#define LOG_LEVEL LOG_LEVEL_INFO
#define UDP_CLIENT_PORT 8765
#define UDP_SERVER_PORT 5678

// Sensor Reading Timers
#define HUMIDITY_TIMER (CLOCK_SECOND * 60*10)
#define PH_TIMER (CLOCK_SECOND * 60 * 2*10)

// UDP Transmission Timer
#define SEND_INTERVAL (CLOCK_SECOND * 60 * 2*10)
#define RECOVERY_TIMER (CLOCK_SECOND * 60 * 2)
#define JITTER_MAX (CLOCK_SECOND * 30)

#define MAX_VALUES 24
```

```
#define BATTERY_LIMIT 20
#define BATTERY_BACK 500

static struct simple_udp_connection udp_conn;

static uint16_t humidityValues[MAX_VALUES];
static uint16_t phValue = 0;
static int humidity_index = 0;

// Battery and Energy Tracking
static double remaining_battery = BATTERY;
static double total_energy_used = 0;
static uint32_t battery_int, battery_frac;
static uint32_t total_energy_int, total_energy_frac;

unsigned long elapsed_seconds =0;

static uint32_t total_completed_transmissions = 0;
static uint32_t total_retransmissions = 0;
static uint32_t total_transmissions = 0;
static uint32_t total_delay_ms = 0;
static uint32_t pdr_percent_x100=0;
static clock_time_t sent_time=0;

static double last_energy_update_time = 0;
static int recovering = 0;

void update_energy_usage(void);
void update_battery_percentage(void);

// Processes
PROCESS(ph_sensor_process, "pH Sensor Process");
PROCESS(humidity_sensor_process, "Humidity Sensor Process");
PROCESS(udp_transmission_process, "UDP Transmission Process");
AUTOSTART_PROCESSES(&ph_sensor_process,&humidity_sensor_process,
&udp_transmission_process);

/*---------------------------------------------------------------------------*/
// Function to log MAC layer metrics, including delay
static void log_mac_metrics(void) {
  set_recharge(recovering);
  total_retransmissions = mac_get_total_retransmissions();
  total_transmissions = mac_get_total_transmissions();
  total_completed_transmissions = mac_get_total_completed_transmissions();

  pdr_percent_x100 = (total_transmissions > 0)
                                ? (total_completed_transmissions * 10000 /
(total_transmissions))
                                : 0;
```

```
  uint32_t avg_delay = (total_completed_transmissions > 0) ? total_delay_ms /
total_completed_transmissions : 0;

  LOG_INFO("Package stats: Retransmissions=%" PRIu32 ", Total transmissions=%"
PRIu32
           ", Total completed transmissions=%" PRIu32 ", PDR=%" PRIu32 ".%02"
PRIu32 "%%, Avg Delay=%" PRIu32 " ms\n",
           total_retransmissions, total_transmissions,
total_completed_transmissions,
           pdr_percent_x100 / 100, pdr_percent_x100 % 100, avg_delay);
}

// Function to handle energy recovery per second
void recover_energy_per_second(unsigned long total_time) {

  static double total_recovery = 0;           // Cumulative recovered energy

  if (total_time > 0) {
    // Calculate recovery only for the elapsed time
    double interval_recovery = total_time * ENERGY_RECOVERY_PER_SECOND;

    // Add recovery to the battery, capping at BATTERY
    remaining_battery += interval_recovery;
    if (remaining_battery > BATTERY) {
      remaining_battery = BATTERY;
    }

    // Update total recovery
    total_recovery += interval_recovery;

    // Convert recovery values to integer and fractional parts for logging
    uint32_t interval_recovery_int = (uint32_t)interval_recovery;
    uint32_t interval_recovery_frac = (uint32_t)((interval_recovery -
interval_recovery_int) * 10000);
    uint32_t total_recovery_int = (uint32_t)total_recovery;
    uint32_t total_recovery_frac = (uint32_t)((total_recovery -
total_recovery_int) * 10000);

    update_battery_percentage();

    // Log recovery details
    LOG_INFO("Battery: %lu.%04lu%%, Interval Recovery: %lu.%04lu J, Total
Recovery: %lu.%04lu J, Total energy consumed: %lu.%04lu\n",
             battery_int, battery_frac, interval_recovery_int,
interval_recovery_frac,
             total_recovery_int, total_recovery_frac, total_energy_int,
total_energy_frac);


  }
```

```
}


// UDP Callback function
static void udp_rx_callback(struct simple_udp_connection *c,
                            const uip_ipaddr_t *sender_addr,
                            uint16_t sender_port,
                            const uip_ipaddr_t *receiver_addr,
                            uint16_t receiver_port,
                            const uint8_t *data,
                            uint16_t datalen) {
  LOG_INFO_6ADDR(sender_addr);
    // Update delay upon any response
  clock_time_t rtt = clock_time() - sent_time;
  uint32_t rtt_ms = (rtt * 1000) / CLOCK_SECOND;
  total_delay_ms += rtt_ms;
    // Update energy usage for reception
  update_energy_usage();
}

// Function to convert ticks to seconds
static inline unsigned long to_seconds(uint64_t time) {
  return (double)(time / ENERGEST_SECOND);
}

// Calculate energy based on current and time
double calculate_energy(double current, unsigned long time) {
  return current * time * VOLTAGE;
}

// Function to convert raw humidity data
static uint16_t convert_humidity(uint16_t raw_value) {
  uint16_t integer_part = (uint16_t)((raw_value) / 4.095 + 6);
  uint16_t fractional_part = (uint16_t)(((raw_value) * 10) / 4.095) % 10;
  return integer_part * 10 + fractional_part;
}

// Function to convert raw pH data
static uint16_t convert_ph(uint16_t raw_value) {
  uint16_t integer_part = (uint16_t)((raw_value) / 40.95 + 6);
  uint16_t fractional_part = (uint16_t)(((raw_value) * 10) / 4.095) % 10;
  return integer_part * 10 + fractional_part;
}

// Function to update battery percentage as integer and fractional parts
void update_battery_percentage() {
  double battery_percentage = (remaining_battery / BATTERY) * 100;
  battery_int = (uint32_t)battery_percentage;
  battery_frac = (uint32_t)((battery_percentage - battery_int) * 10000);
}
```

```
// Function to update total energy used as integer and fractional parts
void update_total_energy() {
  total_energy_int = (uint32_t)total_energy_used;
  total_energy_frac = (uint32_t)((total_energy_used - total_energy_int) *
10000);
}

// Calculate energy consumption since the last measurement
void update_energy_usage() {
  //static double last_energy_update_time = 0; // Track the last energy update
time
  static uint64_t prev_cpu = 0, prev_lpm = 0, prev_tx = 0, prev_rx = 0;

  // Flush energest data to get the most recent values
  energest_flush();

  // Get current energest readings
  uint64_t current_cpu = energest_type_time(ENERGEST_TYPE_CPU);
  uint64_t current_lpm = energest_type_time(ENERGEST_TYPE_LPM);
  uint64_t current_tx = energest_type_time(ENERGEST_TYPE_TRANSMIT);
  uint64_t current_rx = energest_type_time(ENERGEST_TYPE_LISTEN);

  // Calculate the delta times for each state since the last update
  unsigned long cpu_time = (unsigned long)to_seconds(current_cpu - prev_cpu);
  unsigned long lpm_time = (unsigned long)to_seconds(current_lpm - prev_lpm);
  unsigned long tx_time = (unsigned long)to_seconds(current_tx - prev_tx);
  unsigned long rx_time = (unsigned long)to_seconds(current_rx - prev_rx);

  // Calculate energy consumption for each state
  double cpu_energy = calculate_energy(CURRENT_CPU_ACTIVE, cpu_time);
  double lpm_energy = calculate_energy(CURRENT_CPU_LOW_POWER, lpm_time);
  double tx_energy = calculate_energy(CURRENT_RADIO_TX, tx_time);
  double rx_energy = calculate_energy(CURRENT_RADIO_RX, rx_time);

  // Calculate the total energy consumed during this interval
  double interval_energy = cpu_energy + lpm_energy + tx_energy + rx_energy;

  // Deduct energy from remaining battery
  remaining_battery -= interval_energy;
  if (remaining_battery < 0) {
    remaining_battery = 0;
  }
  total_energy_used += interval_energy;

  // Update previous energest values for the next calculation
  prev_cpu = current_cpu;
  prev_lpm = current_lpm;
  prev_tx = current_tx;
  prev_rx = current_rx;
```

```
  // Update battery percentage and total energy used
  update_battery_percentage();
  update_total_energy();
}


/*---------------------------------------------------------------------------*/
// pH Sensor Process with Energy Consumption Tracking
PROCESS_THREAD(ph_sensor_process, ev, data)
{
  static struct etimer ph_timer;
  static struct etimer recovery_timer;


  PROCESS_BEGIN();

  SENSORS_ACTIVATE(phidgets);
  etimer_set(&ph_timer, PH_TIMER);
  etimer_set(&recovery_timer, RECOVERY_TIMER);

  while (1) {

    if (remaining_battery < BATTERY_LIMIT) {
      recovering = 1;
      set_recharge(recovering);
      LOG_WARN("Battery limit PH (%lu.%04lu%%). Pausing sensor operations.\n",
battery_int, battery_frac);
      NETSTACK_MAC.off();
      NETSTACK_RADIO.off();

    } else if (recovering && remaining_battery >= BATTERY_BACK) {
      recovering = 0;
      set_recharge(recovering);
      LOG_INFO("Battery sufficient PH (%lu.%04lu%%). Resuming sensor
operations.\n", battery_int, battery_frac);
      NETSTACK_MAC.on();
      NETSTACK_RADIO.on();

      etimer_reset(&ph_timer);
    }

    if (!recovering) {
      PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&ph_timer));

      energest_flush();
      uint64_t start_time = to_seconds(ENERGEST_GET_TOTAL_TIME());

      uint16_t raw_ph = phidgets.value(PHIDGET5V_2);
      phValue = convert_ph(raw_ph);
```

```
      energest_flush();
      unsigned long sensor_active_time = to_seconds(ENERGEST_GET_TOTAL_TIME()) -
start_time;
      double sensor_energy = calculate_energy(CURRENT_PH_SENSOR,
sensor_active_time);

      remaining_battery -= sensor_energy;
      total_energy_used += sensor_energy;

      if (remaining_battery < 0) remaining_battery = 0;

      LOG_INFO("Recovering: %s pH Sensor Reading: %u.%u Battery: %lu.%04lu%%
Total Energy: %lu.%04lu J\n",
               recovering ? "true" : "false", phValue / 10, phValue % 10,
battery_int, battery_frac, total_energy_int, total_energy_frac);

      etimer_reset(&ph_timer);
    }

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&recovery_timer));
    etimer_reset(&recovery_timer);
  }

  PROCESS_END();
}


/*---------------------------------------------------------------------------*/
// Humidity Sensor Process with Energy Consumption Tracking
PROCESS_THREAD(humidity_sensor_process, ev, data)
{
  static struct etimer humidity_timer;
  static struct etimer recovery_timer;
  double total_time = 0;
  double elapsed_time = 0;

  PROCESS_BEGIN();

  SENSORS_ACTIVATE(phidgets);
  etimer_set(&humidity_timer, HUMIDITY_TIMER);
  etimer_set(&recovery_timer, RECOVERY_TIMER);

  while (1) {
    total_time = to_seconds(ENERGEST_GET_TOTAL_TIME());
    elapsed_time = total_time - last_energy_update_time;
    last_energy_update_time = total_time;

    recover_energy_per_second(elapsed_time);
```

```
    if (remaining_battery < BATTERY_LIMIT) {
      recovering = 1;
      set_recharge(recovering);
      LOG_WARN("Battery limit Humidity (%lu.%04lu%%). Pausing sensor
operations.\n", battery_int, battery_frac);
      NETSTACK_MAC.off();
      NETSTACK_RADIO.off();

    } else if (recovering && remaining_battery >= BATTERY_BACK) {
      recovering = 0;
      set_recharge(recovering);
      LOG_INFO("Battery sufficient Humidity (%lu.%04lu%%). Resuming sensor
operations.\n", battery_int, battery_frac);
      NETSTACK_MAC.on();
      NETSTACK_RADIO.on();

      etimer_reset(&humidity_timer);
    }

    if (!recovering) {
      PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&humidity_timer));

      energest_flush();
      uint64_t start_time = to_seconds(ENERGEST_GET_TOTAL_TIME());

      uint16_t raw_humidity = phidgets.value(PHIDGET5V_1);
      humidityValues[humidity_index] = convert_humidity(raw_humidity);

      energest_flush();
      unsigned long sensor_active_time = to_seconds(ENERGEST_GET_TOTAL_TIME()) -
start_time;
      double sensor_energy = calculate_energy(CURRENT_HUMIDITY_SENSOR,
sensor_active_time);

      remaining_battery -= sensor_energy;
      total_energy_used += sensor_energy;

      if (remaining_battery < 0) remaining_battery = 0;

      LOG_INFO("Recovering: %s Humidity Sensor Reading [%d]: %u.%u Battery:
%lu.%04lu%% Total Energy: %lu.%04lu J\n",
               recovering ? "true" : "false", humidity_index,
humidityValues[humidity_index] / 10, humidityValues[humidity_index] % 10,
               battery_int, battery_frac, total_energy_int, total_energy_frac);

      humidity_index = (humidity_index + 1) % MAX_VALUES;

      etimer_reset(&humidity_timer);
    }
```

```
    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&recovery_timer));
    etimer_reset(&recovery_timer);
  }

  PROCESS_END();
}

/*---------------------------------------------------------------------------*/
PROCESS_THREAD(udp_transmission_process, ev, data)
{
  static struct etimer transmission_timer;
  static struct etimer recovery_timer;
  static uip_ipaddr_t dest_ipaddr;
  char message_buffer[512];


  PROCESS_BEGIN();

  simple_udp_register(&udp_conn, UDP_CLIENT_PORT, NULL, UDP_SERVER_PORT,
udp_rx_callback);
  etimer_set(&transmission_timer, SEND_INTERVAL+ (random_rand() % (2 *
JITTER_MAX)) - JITTER_MAX);
  etimer_set(&recovery_timer, RECOVERY_TIMER);

  while (1) {

    if (remaining_battery < BATTERY_LIMIT) {
      recovering = 1;
      set_recharge(recovering);
      LOG_WARN("Battery limit UDP (%lu.%04lu%%). Pausing transmissions.\n",
battery_int, battery_frac);
      NETSTACK_MAC.off();
      NETSTACK_RADIO.off();

    } else if (recovering && remaining_battery >= BATTERY_BACK) {
      recovering = 0;
      set_recharge(recovering);
      LOG_INFO("Battery sufficient UDP (%lu.%04lu%%). Resuming
transmissions.\n", battery_int, battery_frac);
      NETSTACK_MAC.on();
      NETSTACK_RADIO.on();

      etimer_reset(&transmission_timer);
    }

    if (!recovering) {
      PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&transmission_timer));

      sent_time = clock_time();
      if (NETSTACK_ROUTING.node_is_reachable() &&
```

```
NETSTACK_ROUTING.get_root_ipaddr(&dest_ipaddr)) {
        snprintf(message_buffer, sizeof(message_buffer),
                 "Battery: %lu.%04lu%% pH: %u.%u Humidity: [", battery_int,
battery_frac, phValue / 10, phValue % 10);

        for (int i = 0; i < 24; i++) {
          char humidity_str[8];
          snprintf(humidity_str, sizeof(humidity_str), "%u.%u%s",
                   humidityValues[i] / 10, humidityValues[i] % 10, (i < 23) ? ",
" : "");
          strncat(message_buffer, humidity_str, sizeof(message_buffer) -
strlen(message_buffer) - 1);
        }

        snprintf(message_buffer + strlen(message_buffer), sizeof(message_buffer)
- strlen(message_buffer),
                 "] PDR=%" PRIu32 ".%02" PRIu32 "%%", pdr_percent_x100 / 100,
pdr_percent_x100 % 100);

        simple_udp_sendto(&udp_conn, message_buffer, strlen(message_buffer),
&dest_ipaddr);
        LOG_INFO("Data sent to ");
        LOG_INFO_6ADDR(&dest_ipaddr);
        LOG_INFO_("\n");
        LOG_INFO("Recovering: %s \n", recovering ? "true" : "false");
      } else {
        LOG_WARN("Node not reachable.\n");
      }

      log_mac_metrics();
      etimer_reset(&transmission_timer);
    }
    LOG_INFO("Package stats: Retransmissions=%" PRIu32 ", Total transmissions=%"
PRIu32
           ", Total completed transmissions=%" PRIu32 ", PDR=%" PRIu32 ".%02"
PRIu32 "%% \n",
           total_retransmissions, total_transmissions,
total_completed_transmissions,
           pdr_percent_x100 / 100, pdr_percent_x100 % 100);

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&recovery_timer));
    etimer_reset(&recovery_timer);
  }

  PROCESS_END();
}
```