UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ANÁLISE E DESENVOLVIMENTO DA TÉCNICA DUTY CYCLING PARA REDUÇÃO DO CONSUMO DE ENERGIA DA COMUNICAÇÃO A RÁDIO

TRABALHO DE CONCLUSÃO DE CURSO

Eduardo Luzzi

Santa Maria, RS, Brasil

2015

ANÁLISE E DESENVOLVIMENTO DA TÉCNICA DUTY CYCLING PARA REDUÇÃO DO CONSUMO DE ENERGIA DA COMUNICAÇÃO A RÁDIO

Eduardo Luzzi

Monografia apresentada à Universidade Federal de Santa Maria – UFSM, como requisito parcial para obtenção do título de **Engenheiro de Computação**.

Orientador: Dr. Carlos Henrique Barriquello

Santa Maria, RS, Brasil

2015

UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

A Comissão Examinadora, abaixo assinada,
aprova a Monografia

ANÁLISE E DESENVOLVIMENTO DA TÉCNICA DUTY CYCLING PARA REDUÇÃO DO CONSUMO DE ENERGIA DA COMUNICAÇÃO A RÁDIO

elaborado por
Eduardo Luzzi

como requisito parcial para obtenção de título de
Engenheiro de Computação

COMISSÃO EXAMINADORA:

Carlos Henrique Barriquello, Dr.
(Orientador)

José Eduardo Baggio, Dr (UFSM)

Mateus Beck Rutzig, Dr (UFSM)

Santa Maria, 14 de dezembro de 2015.

RESUMO

Trabalho de Conclusão de Curso
Curso de Engenharia de Computação
Universidade Federal de Santa Maria

ANÁLISE E DESENVOLVIMENTO DA TÉCNICA DUTY CYCLING PARA REDUÇÃO DO CONSUMO DE ENERGIA DA COMUNICAÇÃO A RÁDIO

AUTOR: Eduardo Luzzi
ORIENTADOR: Dr. Carlos Henrique Barriquello
Data e local da defesa: Santa Maria, 14 de dezembro de 2015

O principal objetivo deste trabalho é desenvolver uma técnica que reduza o consumo de energia da comunicação a rádio. Para isso optou-se pelo uso do protocolo *Duty Cycling*, no qual o rádio fica em modo *sleep* a maior parte do tempo e regularmente liga para verificar se outro dispositivo está tentando se comunicar com ele. Deste modo, desenvolveu-se esta técnica programando um dos rádios para ligar periodicamente em modo de recepção, e ao chegar algum pacote ele saberá que o seu par está requisitando os seus dados, então se tornará um transmissor e enviará as suas informações. Está técnica foi implementada de duas formas, uma operando com apenas um canal e outra com vários canais, pois permanecer em um canal fixo pode piorar o seu desempenho, caso passe a sofrer com a interferência de outros dispositivos. Portanto, se empregarmos mais canais e alternarmos entre eles, o rádio nunca ficará no mesmo canal congestionado. Com o desenvolvimento concluído, realizou-se os testes da quantidade necessária de envios até conseguir se comunicar e do tempo médio de comunicação, além do cálculo de consumo de bateria, comparando quanto se economiza em relação a um rádio que fica o tempo inteiro ligado. Enfim, conclui-se que apesar de perder mais que o dobro de tempo de comunicação, com o uso da técnica ganhou-se 90% de economia no consumo da bateria. O modelo multicanal tem desempenho pior que o Monocanal operando em uma faixa de frequência com pouca interferência, porém ainda é melhor que em um canal com muita distorção. Além disto, o multicanal mostrou-se mais robusto com a variação da distância, porém, o ideal seria fazer uma mescla entre as técnicas.

Palavras-chave: *Duty Cycling*, *sleep*, canal, bateria, tempo, desempenho.

ABSTRACT

Work From Completion of Course
Degree in Computer Engineering
Federal University of Santa Maria

ANALYSIS AND DEVELOPMENT OF DUTY CYCLING TECHNIQUE FOR REDUCTION OF ENERGY CONSUMPTION IN RADIO COMMUNICATION

AUTHOR: Eduardo Luzzi
ADVISOR: Dr. Carlos Henrique Barriquello
Date and defense place: Santa Maria, December 14, 2015.

The main objective of this work is to develop a technique that reduces the consumption of radio communication. For this to happen we chose to use the Duty Cycling protocol, in which the radio stays in the sleep mode most part of the time and regularly turns on to verify if another device is trying to communicate with it. That way, we developed this technique programming one of the radios periodically on the receiving mode, and when a packet comes the device will know its pair is requesting its data, then it will become a transmitter and will send its information. This technique was implemented in two forms, one operating with one channel and another one with many channels, because staying in one channel can worsen its performance, in case it starts to be interfered by another device. Therefore, if we use more channels and alternate between them, the radio will never stands on the congested channel. With the development finished, we did tests about the necessary quantity of transmissions until the devices get communication, the average time of communication and calculate the battery consumption, comparing with a radio which stays always on. Ultimately, we conclude that despite losing more than the double time of communication, using the technique 90% of battery consumption was reduced. The multichannel model has a worst performance than the single channel model operating on a low interference frequency band, however it is still better in a channel with a lot of noise and interference. Beyond that the multichannel proved being more resistant to the distance variation, however, the ideal way to use would be mixing the techniques.

Keywords: Duty Cycling, sleep, channel, battery, time, performance.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ANEXOS

LISTA DE SIGLAS

ACK – *Acknowledgement packet*
BO – *Beacon Order*
CCA – *Clear Channel Assessment*
CI – Circuito Integrado
CSMA-CA – *Carrier Sense Multiple Access-Collision Avoidance*
EEPROM – *Electrically-Erasable Programmable Read-Only Memory*
FCS – *Frame Check Sequence*
FIFO – *First In, First Out*
GPIO – *General Purpose Input/Output*
IF – *Intermediate Frequency*
LDO – *Low-dropout*
LNA – *Low Noise Amplifier*
LQI – *Link Quality Indication*
MAC – *Medium Access Control*
PAN – *Personal Area Network*
PLL – *Phase-Locked Loop*
PHY – *Physical*
PLCP – *Physical Layer Convergence Protocol*
PSDU – *PLCP Service Data Unit*
RF – Radiofrequência
RSSI – *Received Signal Strength Indication*
SN – *Sequence Number*
SO – *Superframe Order*
SPI – *Serial Peripheral Interface*
USB – *Universal Serial Bus*
VCO – *Voltage-controlled oscillator*

SUMÁRIO

INTRODUÇÃO

A origem deste projeto está na elaboração de um sistema de monitoramento hospitalar, cuja ideia principal é implantar um aparelho que atue como sensor, medindo os dados vitais dos pacientes. Assim, automatizando a obtenção destes sinais vitais, dispensa-se a necessidade de uma enfermeira verificar periodicamente temperatura, pressão, respiração, entre outros dados. Este aparelho envia os dados através de uma rede sem fio para um computador central que armazena todas as medidas em um banco dados, podendo ser acessado por uma página da web ou aplicativo *Android*, notificando imediatamente o médico responsável caso haja algo de errado com o seu paciente.

Este trabalho foca exclusivamente no desenvolvimento de um protocolo de comunicação que faça economia de energia, pois visa a criação de um aparelho portátil que possa ser carregado para qualquer lugar junto do paciente. O protocolo escolhido para este projeto consiste na ideia de "liga e desliga", chamado de *Duty Cycling*, no qual por um curto período de tempo o aparelho sensor ficará ligado procurando a comunicação e, após esse período, reduzirá a sua potência e ficará em modo *sleep*. Desta forma, programando o aparelho para ficar mais tempo desligado do que ligado, o protocolo faz o seu papel na economia da energia.

A transmissão será feita por um rádio de modelo MRF24J40 com taxa de transferência de 250 kbps (IEEE 802.15.4) que opera na mesma faixa de frequência da rede Wi-Fi (2,4GHz). Utilizaremos um kit de desenvolvimento com o microcontrolador PIC18F4520, o qual será o portador do software que controla o rádio.

Uma das partes deste projeto é o desenvolvimento do protocolo *Duty Cycling* fazendo uso de um canal fixo entre os 16 disponíveis no rádio. Levando em consideração que a frequência de operação é a mesma da Wi-Fi e a potência dos roteadores é maior que a máxima do MRF24J40, determinados canais sofrerão com a interferência e deixarão o desempenho do rádio comprometido. Para contornar esse problema será implementado um sistema *Duty Cycling Multi-Channel*, cuja técnica troca o canal da rede a cada envio de pacote. Assim, tenta-se evitar a utilização de um canal que tenha a interferência de outro sinal.

Como já foi dito este projeto consiste na elaboração de um sistema de monitoramento hospitalar, e considerando que um hospital possui diversos aparelhos e sinais que podem interferir na comunicação, com a utilização do *Duty Cycling* se criará um sistema mais confiável

em termos de envio e recebimentos de dados, buscando a mínima interferência de outros sinais. Além disso, como principal objetivo deste trabalho será feito um estudo comparando a entrega dos pacotes e o tempo médio de comunicação das técnicas desenvolvidas, *Duty Cycling* Monocanal e *Duty Cycling* Multicanal. E por último, serão realizados os cálculos de consumo de bateria do rádio proposto no trabalho e de um rádio que sempre fica ligado, para compará-los e constatar o quanto se economizaria.

Motivação

As principais motivações deste trabalho são:

- Aplicar o conhecimento obtido durante a graduação nos assuntos de redes e microcontroladores;
- Automatizar o serviço hospitalar monitorando em tempo real o estado dos pacientes, a fim de agilizar o tratamento caso haja algo de errado com o enfermo;
- Criar um sistema móvel de baixo consumo de energia, robusto em relação à entrega dos dados e com pouca necessidade de manutenção.

Objetivos

Os principais objetivos são:

- Estudar e implementar diferentes técnicas para redução do consumo de energia da comunicação por rádio;
- Avaliar o desempenho das técnicas também com relação aos seguintes quesitos: consumo de energia, entrega dos pacotes e latência da comunicação. E considerando as diferentes condições de operação, avaliar a interferência no sinal e o tráfego de dados no canal.

Estrutura do trabalho

Aqui encontra-se uma breve descrição de cada capítulo:

- Capítulo 1: Será feita uma revisão bibliográfica explicando como é o funcionamento do protocolo utilizado no ContikiMAC. Este foi o rádio utilizado como base para a elaboração da técnica proposta nesta monografia;
- Capítulo 2: Será apresentado quais foram os hardwares e softwares utilizados para o desenvolvimento do trabalho, citando algumas de suas características;
- Capítulo 3: Será feita a descrição de como foi configurado o rádio e de que forma foi desenvolvida a lógica que desempenha a técnica proposta;
- Capítulo 4: Serão apresentados os resultados obtidos com os testes e cálculos da entrega dos pacotes, tempo médio de comunicação e consumo de energia;
- Capítulo 5: Será feita a conclusão do trabalho citando qual seria a técnica ideal e propondo algumas melhorias para trabalhos futuros.

1. REVISÃO BIBLIOGRÁFICA

1.1 Introdução

Neste capítulo descreve-se detalhadamente a implementação do protocolo *Duty Cycling* do *ContikiMAC*, que foi o rádio utilizado como modelo para o desenvolvimento do trabalho. Nas próximas sessões descreve-se a lógica trabalhada pelos dois autores que mais influenciaram a elaboração deste TCC. Na seção 2.2, o "*The ContikiMAC Radio Duty Cycling Protocol*" (DUNKELS, 2011) utiliza o protocolo tratando apenas do uso de um canal fixo e na seção 2.3, o "*Multichannel Communication in Contiki's Low-power IPv6 Stack*" (AL NAHAS, 2013) mostra como o autor adaptou o protocolo para a utilização de vários canais. Além destes trabalhos, outros foram utilizados como referência, porém não obtiveram a mesma influência da elaboração do protocolo abordado nesta monografia. Desta forma, não serão descritos detalhadamente, mas estão inclusos no capítulo de referência para quem desejar fazer um estudo mais aprofundado.

1.2 *ContikiMAC*

Um dos artigos mais citados sobre o protocolo *Duty Cycling* é o "*The ContikiMAC Radio Duty Cycling Protocol*" (DUNKELS, 2011), no qual o autor propõe a utilização do *ContikiMAC*, rádio que utiliza o protocolo em questão ligando periodicamente para verificar se há algum pacote sendo transmitido para ele. O desenvolvimento do *ContikiMAC* foi realizado no sistema operacional *Contiki 2.5* utilizando o rádio de modelo CC2420 e o microcontrolador MSP430.

No trabalho de Adam Dunkels (2011), quando o transmissor decide enviar um dado, ele repete o envio do mesmo até receber um pacote de reconhecimento (ACK). Já o receptor liga de tempo em tempo, e quando detecta alguma comunicação, permanece ligado até receber todo o pacote e enviar o ACK para confirmar ao transmissor que houve sucesso. Caso o transmissor queira enviar o mesmo pacote para todos os rádios da rede, ele repete o envio durante o tempo

de intervalo entre um *wake-up* e outro sem esperar o ACK, garantindo que todos os rádios da rede receberam o pacote. As figuras 1.1 e 1.2 ilustram a técnica utilizada pelo *ContikiMAC*.

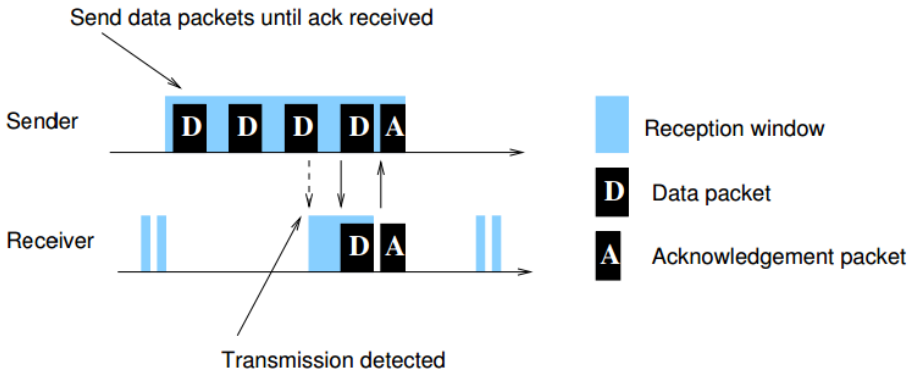


Figura 1.1 – *ContikMAC* transmitindo para um receptor.

(Fonte: DUNKELS, 2011)

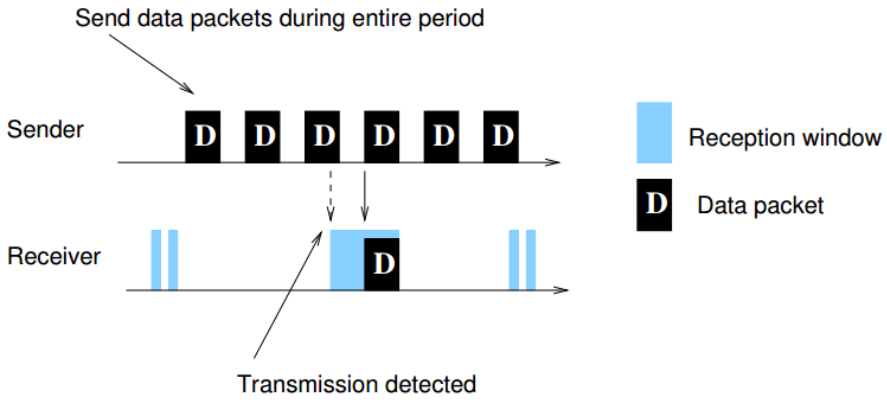


Figura 1.2 – *ContikMAC* transmitindo para todos os receptores.

(Fonte: DUNKELS, 2011)

O mecanismo do *ContikMAC* faz a verificação do *Clear Channel Assessment* (CCA) em um modo que usa como parâmetro o *Received Signal Strength Indication* (RSSI – medida da potência do sinal recebido). Caso a força do sinal no canal seja baixa (baixo RSSI), o CCA retorna indicando que o canal está limpo, portanto o rádio entende que não há dados sendo enviados no momento. Se o nível do RSSI for alto, então o CCA entende que o canal está

ocupado, ou seja, algo está tentando se comunicar com ele, então habilita o modo de recepção. A utilização do CCA para fazer a verificação do canal é justificável por consumir pouca energia.

Como pode-se perceber o receptor acorda duas vezes seguidas em um curto intervalo de tempo, o qual deve ser menor que o tempo de propagação do pacote de dados transmitidos e, também, maior que o intervalo entre um pacote e outro. Assim, se o receptor verificar o CCA no meio do intervalo entre os pacotes, ele certamente irá identificar a comunicação na segunda verificação do CCA.

Também é importante considerar como o rádio saberá se a oscilação no canal é devido a um pacote que está tentando ser enviado ou a alguma interferência. Para fazer esta descoberta o CCA mede quanto tempo o RSSI fica alto e se ultrapassar o tempo de envio de um pacote, então é considerado como interferência e, portanto, ele imediatamente retorna para o modo *sleep*.

Por último, uma das características mais importantes do *ContikiMAC* é a capacidade de guardar o tempo de *wake-up* dos rádios da rede em uma tabela, pois desta forma ele não precisa enviar uma grande quantidade de pacotes antes de conseguir a comunicação. Sabendo o tempo de *wake-up* de determinado destinatário, o transmissor começa a enviar um pouco antes do receptor ligar, economizando tempo de resposta e energia do rádio.

1.3 *ContikiMAC* Multicanal

O principal artigo para o embasamento teórico do protocolo *Duty Cycling* Muilticanal foi o "*Multichannel Communicationin Contiki's Low-power IPv6 Stack*" (AL NAHAS, 2013). O autor também propõe o uso do *ContikiMAC* com o mesmo rádio CC2420 e microcontrolador da *Texas Instruments* MSP430 abordados no tópico anterior, sendo a única diferença a versão do sistema operacional *Contiki 2.6*.

Como o rádio tratado neste tópico é o mesmo do trabalho apresentado no anterior, a lógica de comunicação não muda. Sendo assim, um rádio envia dados enquanto o outro dorme e, de tempo em tempo, verifica o CCA para saber se há alguma tentativa de comunicação. Se houver comunicação, o receptor mantém-se ligado até receber todo o pacote e responde ao transmissor com um ACK para indicar que houve sucesso na transmissão.

O grande diferencial deste trabalho está na utilização de vários canais e para isso uma das partes precisará ditar qual será o canal utilizado. O autor optou por ser o receptor o

responsável por este papel, portanto, a cada nova comunicação ambos os lados, receptor e transmissor, mudam o canal. O receptor escolhe o próximo canal e envia a escolha no ACK; o transmissor checa o pacote de reconhecimento e sintoniza no canal escolhido. Desta forma, ambos mudam para o mesmo canal e não perdem a comunicação – a figura 1.3 deixa esta lógica mais clara. Pode-se perceber que o transmissor está sempre enviando no mesmo canal até conseguir comunicação e receber um ACK com a indicação do próximo canal.

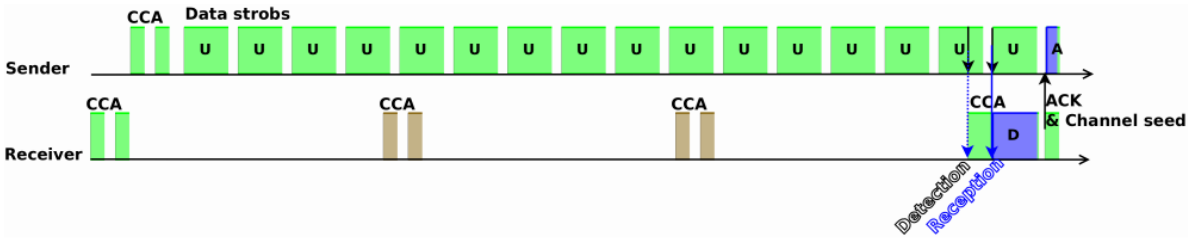


Figura 1.3 – Lógica básica do *ContikiMAC* Multicanal.

(Fonte: AL NAHAS, 2013)

Como pode-se notar na figura 1.3, inicialmente o transmissor não consegue comunicação até o receptor ligar em seu canal. Isto ocorre toda vez que os rádios forem ligados e não estiverem no mesmo canal ou quando o ACK não chegar com a informação do próximo canal, causando uma dessincronização. Como em cada novo *wake-up* o receptor liga em um canal diferente, a sincronização ocorrerá no momento em que ele ligar no mesmo canal do transmissor. A escolha do próximo canal é randômica para tentar impedir o rádio de manter um padrão que passe sempre pelos canais ruins e, também, tentar evitar que os vários rádios da rede se conectem ao mesmo canal.

No caso de uma rede com vários receptores, o transmissor guarda em uma tabela todas as informações dos destinatários com o canal e o tempo de *wake-up*, semelhante a explicação na seção 1.2. Desta forma, ele consegue se comunicar com o receptor desejado sem nenhum problema. Utilizando o tempo de *wake-up* e o canal do destinatário específico dificulta-se a interferência de outros rádios, pois dificilmente se encontrará outro dispositivo utilizando o mesmo canal e possuindo o mesmo ciclo de acordar.

Caso queira enviar o mesmo dado para todos os rádios da rede, a proposta é utilizar mais duas verificações do CCA com um canal exclusivo para o *broadcast*, pois assim cada receptor

verificaria duas vezes o seu canal *unicast* e mais duas o canal *broadcast*, que é o mesmo para todos na rede. A figura 1.4 deixa mais clara essa verificação do canal *broadcast*.

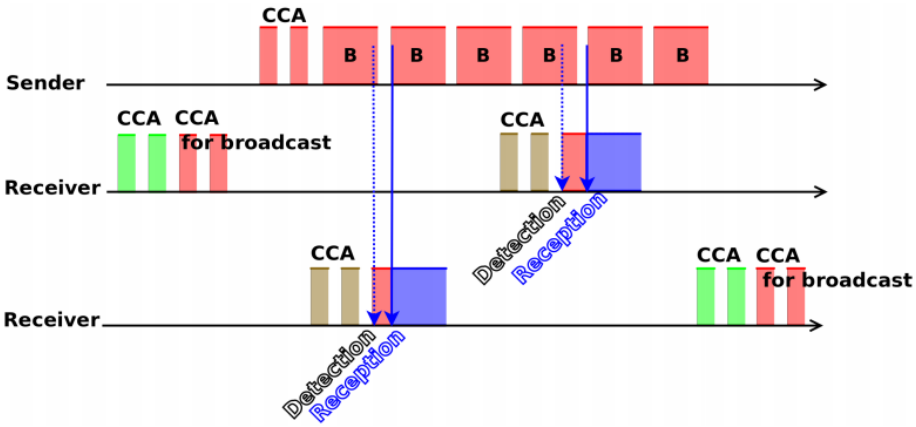


Figura 1.4 – *ContikiMAC* Multicanal com canal broadcast.

(Fonte: AL NAHAS, 2013)

2. HARDWARES E SOFTWARES UTILIZADOS

2.1 Introdução

É importante descrever todos os hardwares e softwares utilizados no desenvolvimento do trabalho para a melhor compreensão dos resultados obtidos. Neste capítulo serão abordadas as principais características do rádio MRF24J40, do kit PICGenios 18F, do microcontrolador PIC18F4520 e da placa criada para fazer a comunicação SPI do rádio com o kit, atendo-se apenas aos fatores que irão influenciar nos resultados do trabalho. Além disso, serão citados e brevemente descritos quais foram os softwares utilizados na compilação e na gravação dos códigos.

2.2 MRF24J40

MRF24J40 é um rádio que utiliza o protocolo IEEE 802.15.4-2013, cujo padrão serve basicamente para redes sem fio com baixas taxas de transmissão. Trabalha na faixa de frequência de 2,4GHz com transferência de 250kbps, podendo chegar a 625kbps (modo turbo) e contem 16 canais que podem ser vistos na tabela 2.1.

Channel Number	Frequency	Channel Number	Frequency
11	2.405 GHz	19	2.445 GHz
12	2.410 GHz	20	2.450 GHz
13	2.415 GHz	21	2.455 GHz
14	2.420 GHz	22	2.460 GHz
15	2.425 GHz	23	2.465 GHz
16	2.430 GHz	24	2.470 GHz
17	2.435 GHz	25	2.475 GHz
18	2.440 GHz	26	2.480 GHz

Tabela 2.1 – Canais do MRF24J40.

(Fonte: MRF24J40 *Data Sheet*)

O rádio possui as seguintes camadas do modelo OSI, física (PHY) e a camada de enlace de dados contendo a subcamada MAC. Na figura 2.1 pode-se visualizar o diagrama de blocos do MRF24J40.

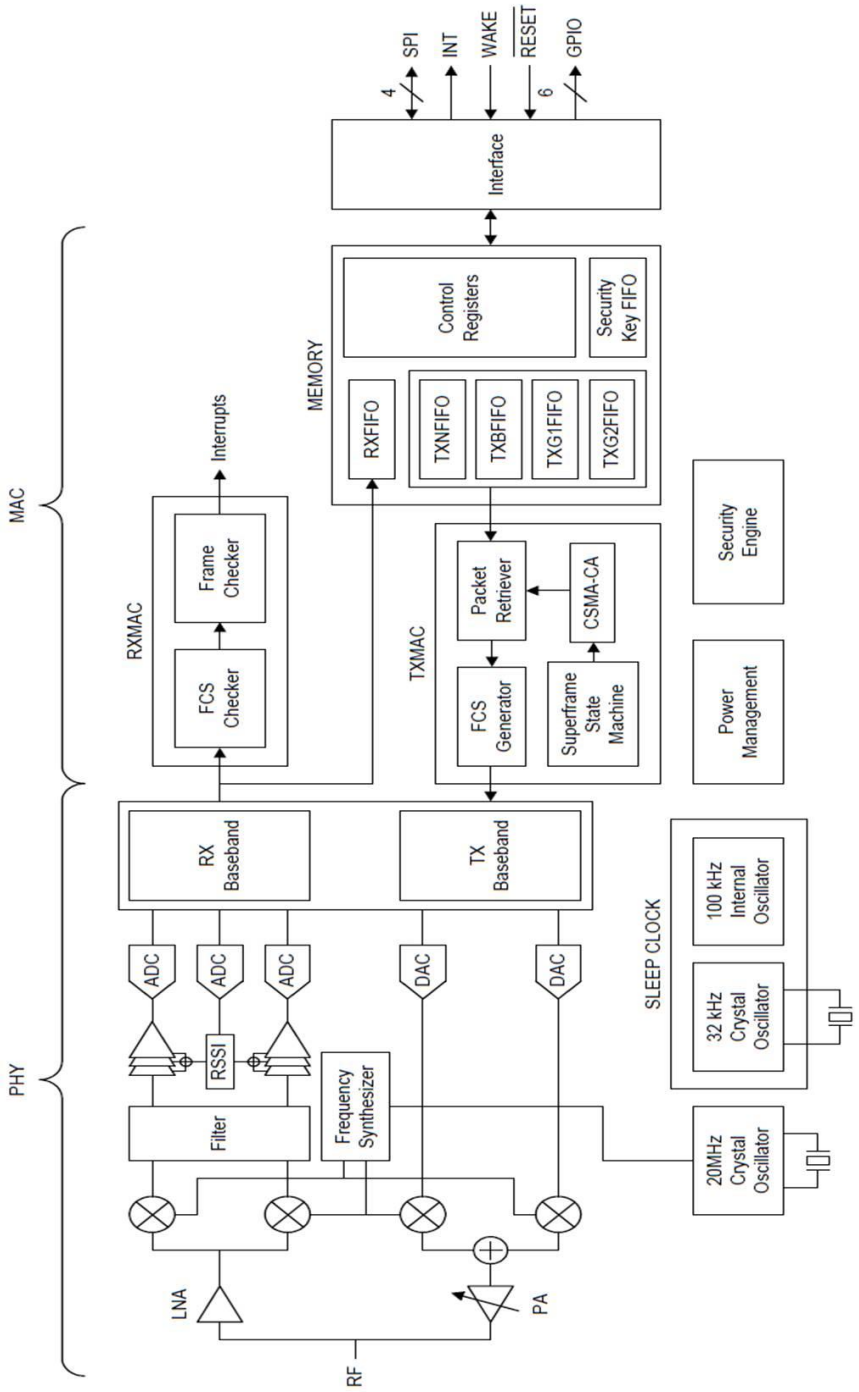


Figura 2.1 – Diagrama de blocos do MRF24J40.

(Fonte: MRF24J40 *Data Sheet*)

O sintetizador de frequência possui um oscilador de cristal de 20MHz que gera a radiofrequência (RF) de 2,4GHz. O consumo típico de corrente é de 19mA quando está recebendo, 23mA quando está transmitindo e 2µA no modo *sleep*.

O receptor possui uma arquitetura *low-IF*, ou seja, seguindo as explicações abordadas no trabalho de graduação de Soares V. F. (2008), em vez do sinal ser convertido diretamente para uma frequência RF, ele é transformado em uma frequência IF de algumas dezenas ou centenas de KHz antes de chegar na banda base. Desta forma, é permitida a utilização de filtros passa alta no caminho de recepção, rejeitando a maior parte do ruído e degradando pouco o sinal. Por outro lado, a grande desvantagem desse tipo de arquitetura é que, caso haja qualquer tipo de descasamento entre a fase e a amplitude dos canais, provoca-se a perda na capacidade de rejeição de outros sinais. Portanto, tem-se uma grande queda no desempenho quando há um sinal de maior potência utilizando o canal de mesma frequência. É interessante a utilização do *low-IF*, pois faz praticamente um meio termo entre outros dois receptores conhecidos, o super-heteródino e o homódino (ou zero-IF). O super-heteródino também transforma o seu sinal em uma frequência IF e é praticamente imune a ruídos. Além disso, por ser uma arquitetura antiga e consolidada, apresenta certas melhorias em relação às outras, sendo o grande entrave na sua utilização a necessidade de possuir um alto fator de qualidade, o que dificulta a sua adaptação em circuitos integrados. O receptor zero-IF não transforma o seu sinal em uma frequência IF e a grande vantagem desta arquitetura está na sua simplicidade e fácil adaptação em circuitos integrados, porém, por fazer a conversão direta do sinal para a banda-base, não é possível a utilização de um filtro passa-alta no seu caminho, tornando o sinal mais ruidoso. Por isso dizemos que o receptor *low-IF* faz um intermediário entre estas arquiteturas, pois converte o seu sinal para IF, possui boa resistência a ruídos e também um circuito simples e de fácil adaptação no CI.

Além disso, o bloco receptor contém um *Low Noise Amplifier* (LNA) utilizado para a amplificação do sinal de baixa potência de entrada, reduzindo ao máximo o ruído. Antes de passar no LNA o sinal passa pelo *mixer* para transformar o sinal de alta frequência em baixa frequência, reduzindo as capacitâncias e indutâncias parasitas, facilitando a amplificação (SOARES, 2008).

O receptor também possui um filtro polifásico, necessário para rejeitar a parte negativa do espectro do sinal. E por último, o bloco detém uma banda base com *received signal strength indication* (RSSI) para medir a força do sinal.

O transmissor possui uma arquitetura que converte diretamente o sinal com potência máxima de saída de 0 dBm, com variação de até 36 dB na potência do sinal transmitido.

O transmissor e receptor interno utilizam os pinos RFp e RFn para fazer a ligação com o *Matching Circuitry*(*Balun*), circuito utilizado para fazer o balanceamento do sinal da antena (AZEVEDO, 2002) – pode-se visualizar esta ligação na figura 2.2. Além disso, pode-se controlar um amplificador de potência e um LNA externo pelo GPIO.

Existem seis pinos de propósito geral de entrada e saída (GPIO) que podem ser configurados internamente para controle ou monitoramento, e também podem ser controlados externamente.

O bloco *Power Management* possui um regulador de tensão DC *Low-dropout* (LDO) possibilitando a configuração do MRF24J40 para trabalhar com baixa corrente (2µA) no modo *sleep*, contendo um oscilador interno de 100kHz e um oscilador de cristal externo de 32kHz que também podem ser usados para o modo *sleep*.

Os módulos RXMAC e TXMAC verificam se o formato dos pacotes está dentro do padrão da IEEE 802.15.4 e armazenam os dados em filas FIFO. Outras características inclusas no hardware são utilização de *superframe* (utilizado para o modo de sincronização do *Beacon*), filtro para os frames, segurança de dados e *Carrier Sense Multiple Access-Collision Avoidance* (CSMA-CA), um método de transmissão disposto para evitar colisões entre os pacotes, explicado com mais detalhes no tópico 3.3.

O controle do rádio é feito por *4-wire Serial Peripheral Interface* (SPI) e pelos pinos de interrupção, *wake* e *reset*. Pode-se visualizar na figura 2.2 o diagrama de blocos do MRF24J40 ligado com um microcontrolador PIC. A maioria das informações contidas neste tópico foram retiradas do *Data Sheet* do rádio e para maiores detalhes sobre o funcionamento do mesmo é aconselhada a leitura do documento.

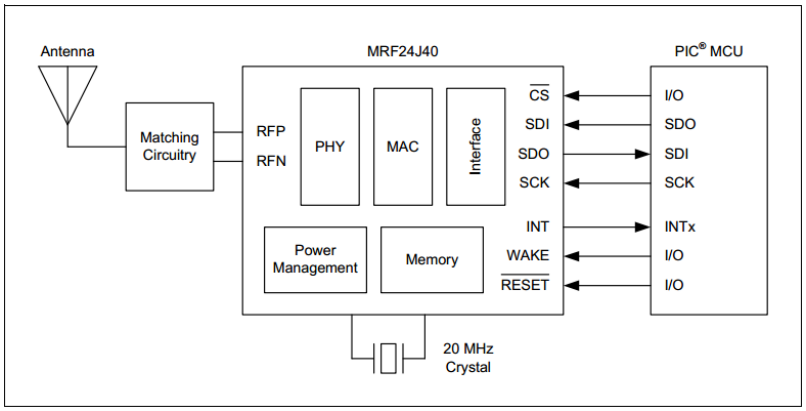


Figura 2.2 – Diagrama de blocos do MRF24J40 ligado ao microcontrolador PIC.

(Fonte: MRF24J40 *Data Sheet*)

2.3 PIC18F4520

O PIC18F4520 é um microcontrolador com o propósito de criar uma interface para o controle do rádio. Uma das grandes vantagens na utilização do PIC é a ótima performance e o baixo custo.

Como a proposta deste trabalho é testar o desempenho do rádio utilizando a técnica *Duty Cycling*, não será tratada as características do PIC com muitos detalhes. A tabela 2.2 resume as principais características do microcontrolador em questão.

Features	PIC18F4520
Operating Frequency	DC – 40 MHz
Program Memory (Bytes)	32768
Program Memory (Instructions)	16384
Data Memory (Bytes)	1536
Data EEPROM Memory (Bytes)	256
Interrupt Sources	20
I/O Ports	Ports A, B, C, D, E
Timers	4
Capture/Compare/PWM Modules	1
Enhanced Capture/Compare/PWM Modules	1
Serial Communications	MSSP, Enhanced USART
Parallel Communications (PSP)	Yes
10-bit Analog-to-Digital Module	13 Input Channels
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable High/Low-Voltage Detect	Yes
Programmable Brown-out Reset	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled
Packages	40-pin PDIP 44-pin QFN 44-pin TQFP

Tabela 2.2 – Características principais do PIC18F4520.

(Fonte: PIC18F2420/2520/4420/4520 *Data Sheet*)

Para a manipulação do PIC18F4520 utilizou-se o kit PICGenios 18F, o qual oferece diversos periféricos que podem ser operados pelo microcontrolador. Para o trabalho foi utilizado uma das portas de expansão, que faz a ligação com o rádio e o display LCD 16x2 modo 8bits, para indicar quando estava em modo recepção ou transmissão. O kit tem suporte para o gravador e debugador MicroICD e através da porta USB do computador é feita a gravação do arquivo hexadecimal com o auxílio do software PICkit 2 v2.60. Este programa também é utilizado para fazer a leitura da memória EEPROM do microcontrolador.

Já o compilador utilizado para a confecção do código e a geração do arquivo hexadecimal é o mikroC PRO for PIC, o qual oferece uma grande quantidade de bibliotecas para a manipulação do PIC, facilitando a criação do código. As bibliotecas oferecem diversas funções, como a utilizada para configuração da comunicação SPI ou para escrita da memória EEPROM. Para demais informação sobre o microcontrolador é aconselhável a leitura do *Data Sheet* do PIC.

2.4 Ligação do hardware

Para a ligação entre o kit PICGenios 18F e o MRF24J40 foi necessário o controle da tensão de alimentação, pois o kit trabalha com 5V, enquanto o rádio suporta no máximo 3,6V, sendo 3,3V a tensão recomendada pelo fabricante (MICROCHIP, 2008). Desta forma, para fazer o intermediário entre os dois adaptou-se uma placa desenvolvida por integrantes do Grupo de Microeletrônica da UFSM, convertendo a tensão de 5V para 3,3V quando são enviados dados do PIC para o rádio, e de 3,3V para 5V quando o microcontrolador deseja ler algo que o MRF24J40 está recebendo. Esquemáticos e materiais utilizados para a manufatura da placa são encontrados nos anexos A, B e C.

Na figura 2.3 pode-se visualizar o rádio em azul ligado na placa que faz a conversão da tensão. Cada CI desta placa é responsável, junto com o regulador de tensão, por fazer as conversões: um dos CIs converte a tensão de cada pino de 5V para 3,3V e o outro faz o contrário.

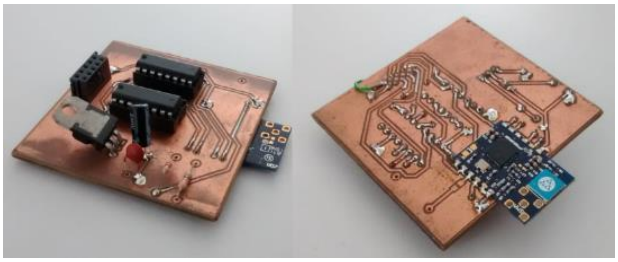


Figura 2.3 – Rádio MR24J40 junto com a placa reguladora de tensão.

A figura 2.4 ilustra todo o hardware utilizado. A placa maior é o kit PICGenios com o PIC18F4520, o gravador MicroICD é a placa vermelha e também pode-se identificar o rádio conectado ao kit.

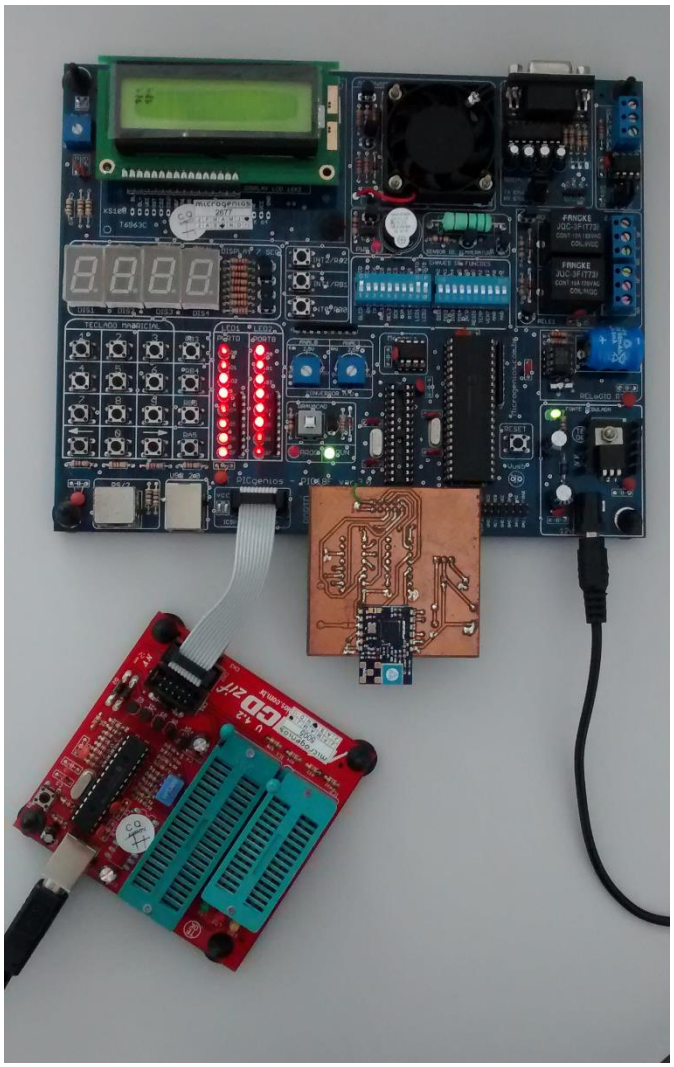


Figura 2.4 – Hardware utilizado no trabalho.

3. DESENVOLVIMENTO DO TRABALHO

3.1 Introdução

Neste capítulo será explicado como foi desenvolvida a técnica *Duty Cycling* desta monografia. Na seção 3.2 justifica-se porque não foi utilizado o *Contiki* para a construção do software. A seção 3.3 possui a descrição do que é preciso configurar inicialmente para deixar o rádio funcionando e como isto foi feito neste trabalho. Na seção 3.4 é demonstrado o formato dos pacotes de envio e recebimento e como eles foram tratados. Por último, nas sessões 3.5 e 3.6 será explicado como funciona a técnica desenvolvida para trabalhar com apenas um canal e com o multicanal.

3.2 Justificativa

Como já foi explicado, o *Duty Cycling* é um protocolo de comunicação que economiza energia através do seu mecanismo de ligar e desligar, no qual geralmente o receptor executa essa mecânica periodicamente para verificar se há algum dispositivo tentando a comunicação.

Para o desenvolvimento deste protocolo é comum a utilização do sistema operacional *Contiki*, que oferece todas as funções necessárias para operar o rádio. Sendo um sistema já consolidado e com inúmeras versões, a sua utilização seria ideal para o desenvolvimento do trabalho, porém, o grande entrave deste sistema é a sua pouca flexibilidade, estando disponível para apenas algumas plataformas específicas e necessitando de um microcontrolador com mais memória que o PIC18F4520.

O sistema tem portabilidade para a plataforma *seed-eye* (CONTIKI-OS), que seria a mais próxima do hardware utilizado neste trabalho. Esta plataforma trabalha com o mesmo rádio MRF24J40, porém utiliza o microcontrolador PIC32MX795F512L que possui 512KB de memória programável, 16 vezes mais que o PIC18F4520. Esta incompatibilidade gerou o principal desafio deste trabalho, tentar reproduzir do início a técnica *Duty Cycling* em cima de um software simples que apenas transmite e recebe dados.

3.3 Configurações Iniciais

A princípio, para a inicialização do rádio são definidas todas as variáveis de endereçamento, instauração dos dados que serão enviados, direção dos pinos de interação entre o microcontrolador e o rádio e configuração da comunicação SPI. Após esta etapa, é feita uma série de resets a fim de estabilizar o rádio, sendo que a primeira delas é o reset feito através do pino do rádio, o qual limpa todas as variáveis de controle dos registradores. Em seguida é feito o *software reset* que reinicializa os seguintes circuitos: gerenciador de energia, banda base e MAC. Por último é feita a reinicialização da máquina de estados RF. Na sequência desta bateria de resets são estabelecidas as partes alta e baixa do endereço do rádio e também o número de identificação do *Personal Area Network* (PAN).

Em seguida é feita a inicialização do rádio no modo *nonbeacon*, o qual faz o envio dos dados simplesmente utilizando o CSMA-CA sem sincronização e *superframe* (ROCHA, 2011). O CSMA-CA é um método de transmissão ordenado que busca reduzir a quantidade de colisões em uma rede, e para isto o rádio avisa que fará uma transmissão e quanto tempo irá demorar, fazendo com que os outros dispositivos não tentem transmitir durante este período, evitando a colisão. Portanto, para realizar esta inicialização são necessárias as configurações básicas do rádio, que habilitam as filas FIFO e definem qual o tempo para início de transmissão (configurado para 96µs) e quais os tempos de processamento de dados recebidos pela camada física (configurado para 144µs). Dentro das configurações básicas também estão as definições de controle do RF, as quais estabelecem as configurações do VCO (*Voltage-controlled oscillator*), habilitam o *Phase-Locked Loop* (PLL – sistema de realimentação utilizado para sincronizar a frequência de saída com a de entrada), ativam o filtro dos dados transmitidos, definem o tempo de recuperação do oscilador de 20 MHz após o *sleep* (configurado para 3ms) e definem a utilização do oscilador interno de 100kHz para o *sleep*. Para finalizar as configurações básicas, define-se como será o *clock* no modo *sleep*. Como recomendado no *data sheet*, desativou-se o *clock* de saída e dividiu-se o sinal do relógio por 2, que é o mínimo necessário para a utilização do oscilador interno.

Após as configurações básicas, segue-se configurando o rádio para operar no modo *nonbeacon*. Primeiro, define-se o modo do CCA para verificar apenas se o sinal no canal segue o protocolo IEEE 802.15.4, sem se importar com a energia deste sinal. Desta forma, forçamos o rádio a continuar tentando a comunicação, mesmo que o sinal tenha pouca potência. Em seguida configura-se o RSSI para armazenar o valor da força do sinal a cada novo pacote

recebido, habilita-se todas as interrupções, define-se qual será o canal utilizado e então damos um novo reset no RF. Para finalizar as configurações do modo *nonbeacon*, define-se o dispositivo como um coordenador PAN para ter total controle da rede, podendo enviar e receber dados de qualquer rádio. Após este processo definiu-se o rádio para trabalhar no modo *Unslotted* CSMA-CA, cujo algoritmo utilizado pode ser observado na figura 3.1. Para finalizar a configuração *nonbeacon*, configura-se o *Beacon Order* (BO) e o *Superframe Order* (SO) com os valores padrões 15, os quais definem o intervalo do frame de *beacon* e a duração do *superframe* que não são utilizados no modo *nonbeacon*, mas requerem configuração para o correto funcionamento do rádio.

Para encerrar as configurações iniciais do rádio define-se a potência do sinal com o valor máximo de -36,3dB, habilita-se o rádio para o modo normal de recepção que aceita qualquer pacote com bom *Cyclic redundancy check* (CRC – método de verificação de erro baseado na sequência de bits) e que esteja dentro do padrão IEEE 802.15.4 e por fim, configura-se o filtro de frame para receber qualquer tipo de frame (comando, dado ou *beacon*).

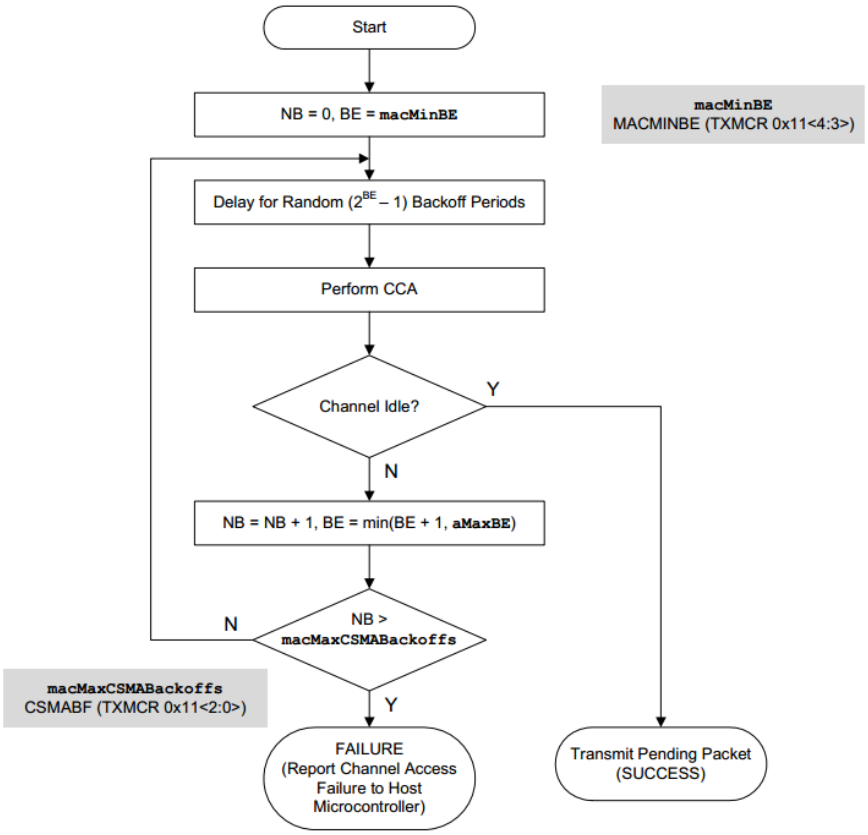


Figura 3.1 – Algoritmo *Unslotted* CSMA-CA.

(Fonte: MRF24J40 *Data Sheet*)

A figura 3.1 ilustra o algoritmo utilizado pelo CSMA-CA, que ao iniciar define as variáveis NB = 0 e BE = *macMinBE* (como padrão o *macMinBE* é 3), e em seguida dá um *delay* igual ao período que o rádio espera para desistir da comunicação. Logo após, é feita a verificação do CCA, e caso identifique alguma comunicação ele transmite o pacote, do contrário ele recalcula os valores de NB e BE e verifica se o NB é maior que o tempo máximo para desistir. Se for maior que este tempo, o rádio para de tentar a comunicação e relata ao microcontrolador uma falha. Para tentar evitar que o rádio encerre a comunicação, escolhemos o CCA no modo que verifica apenas se o pacote está dentro dos padrões IEEE 802.15.4. Se tivéssemos escolhido o modo que verifica a força do sinal, o rádio teria maior dificuldade de comunicação, pois a distância e as interferências de outros sinais diminui a sua potência, o que faria o algoritmo do CSMA-CA encerrar as operações do rádio.

3.4 Configuração do envio e recebimento dos pacotes

A técnica *Duty Cycling* proposta foi implementada com base em dois códigos existentes, que fazem a simples tarefa de transmitir e receber pacotes, sendo que um deles apenas transmite e o outro apenas recebe. Os códigos pertencem ao projeto desenvolvido pelo Grupo de Microeletrônica (Gmicro) da Universidade Federal de Santa Maria (UFSM), e como citado anteriormente, são bem simples e não têm implementado nem mesmo o uso do pacote de reconhecimento. Desta forma, a primeira tarefa foi configurar o rádio para habilitar o ACK automático do MRF24J40, e para realizá-la foi necessário estudar a estrutura dos pacotes de transmissão e recepção do rádio. Os tópicos seguintes servirão para demonstrar esta estrutura dos pacotes e como foram tratados neste trabalho.

3.4.1 Pacote de transmissão

A figura 3.2 ilustra o formato e o caminho dos pacotes que serão transmitidos. Percebe-se que a primeira etapa para o envio dos dados é escrever todo o pacote na fila de transmissão (TX FIFO). Neste trabalho apenas frames de dados foram enviados, então não haverá explicação sobre os pacotes de comando MAC e *Beacon*. Para esclarecer como é o formato do

pacote de dados utilizado, a figura 3.2 mostra a estrutura dos pacotes e a figura 3.3 ilustra o trecho de código que escreve na TX FIFO.

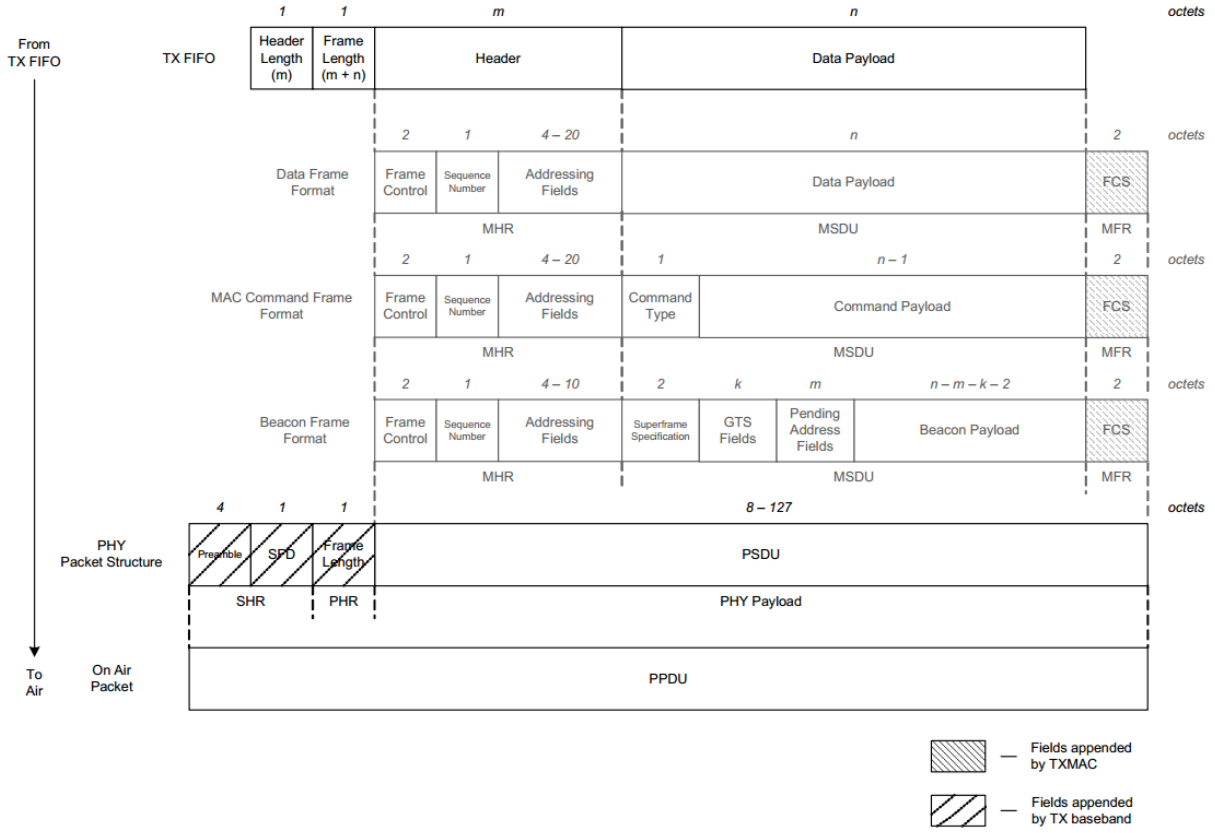


Figura 3.2 – Pacote de transmissão.

(Fonte: MRF24J40 *Data Sheet*)

```
void write_TX_normal_FIFO() {
  int i = 0;

  data_TX_normal_FIFO[0]  = HEADER_LENGHT;
  data_TX_normal_FIFO[1]  = HEADER_LENGHT + DATA_LENGHT;
  data_TX_normal_FIFO[2]  = 0x21;                         // control frame
  data_TX_normal_FIFO[3]  = 0x88;
  data_TX_normal_FIFO[4]  = SEQ_NUMBER;                   // sequence number
  data_TX_normal_FIFO[5]  = PAN_ID_2[1];                  // destinatoin pan
  data_TX_normal_FIFO[6]  = PAN_ID_2[0];
  data_TX_normal_FIFO[7]  = ADDRESS_short_2[0];           // destination address
  data_TX_normal_FIFO[8]  = ADDRESS_short_2[1];
  data_TX_normal_FIFO[9]  = PAN_ID_1[0];                  // source pan
  data_TX_normal_FIFO[10] = PAN_ID_1[1];
  data_TX_normal_FIFO[11] = ADDRESS_short_1[0];           // source address
  data_TX_normal_FIFO[12] = ADDRESS_short_1[1];

  data_TX_normal_FIFO[13] = DATA_TX[0];                   // data
  data_TX_normal_FIFO[14] = DATA_TX[1];                   // data
  data_TX_normal_FIFO[15] = DATA_TX[2];                   // data
  data_TX_normal_FIFO[16] = DATA_TX[3];                   // data
  data_TX_normal_FIFO[17] = DATA_TX[4];                   // data
  data_TX_normal_FIFO[18] = DATA_TX[5];                   // data
  data_TX_normal_FIFO[19] = DATA_TX[6];                   // data
  data_TX_normal_FIFO[20] = DATA_TX[7];                   // data
  data_TX_normal_FIFO[21] = DATA_TX[8];                   // data
  data_TX_normal_FIFO[22] = DATA_TX[9];                   // data
  data_TX_normal_FIFO[23] = DATA_TX[10];                   // data
  data_TX_normal_FIFO[24] = DATA_TX[11];                   // data
  data_TX_normal_FIFO[25] = DATA_TX[12];                   // data
  data_TX_normal_FIFO[26] = DATA_TX[13];                   // data
  data_TX_normal_FIFO[27] = DATA_TX[14];                   // data
  data_TX_normal_FIFO[28] = DATA_TX[15];                   // data

  // write frame into normal FIFO
  for(i = 0; i < (HEADER_LENGHT + DATA_LENGHT + 2); i++) {
    write_ZIGBEE_long(address_TX_normal_FIFO + i, data_TX_normal_FIFO[i]);
  }

  start_transmit();
}
```

Figura 3.3 – Código utilizado para a escrita do pacote na fila de transmissão.

O primeiro byte indica para o rádio qual é o tamanho do cabeçalho do pacote que será transmitido, a variável *HEADER_LENGHT* contém o tamanho do cabeçalho que é contado a partir do *control frame* (byte 2) até o último byte antes do primeiro dado (byte 12), totalizando 11 bytes de cabeçalho. O segundo byte indica o tamanho total do pacote que será transmitido, ou seja, tamanho do cabeçalho (*HEADER_LENGHT*) mais a quantidade de bytes de dados (*DATA_LENGHT* = 16 bytes).

Na sequência, os próximos dois bytes definem o *frame control*, e talvez esta seja a parte mais importante do pacote, pois o *frame control* é utilizado para definir o tipo do dado e

requisitar o ACK, entre outras funções. Para entender como funciona a configuração do *frame control*, a figura 3.4 demonstra a sua estrutura e a documentação da IEEE (2003) serve como referência teórica. O byte 2 do código configura os 8 bits menos significativos do *frame control* e definiu-se este byte com o valor hexadecimal 21, pois assim estabelece o valor do "*Frame type"* como 1, indicando que o pacote é um dado, e ativa o "*Ack. request"* também como 1, indicando que o pacote quer um reconhecimento para confirmar que conseguiu transmitir. O byte 3 do código é definido com o valor hexadecimal 88 e configura os 8 bits mais significativos do campo da figura 3.4, estabelecendo o "*Source addressing mode"* com 1 no bit mais significativo e 0 no outro, indicando que o endereço do rádio possui 16 bits. Além disso, o byte 3 coloca o bit mais significativo do "*Dest. addressing mode"* para 1 indicando que o quadro é destinado para o coordenador do PAN.

Bits: 0–2	**3**	**4**	**5**	**6**	**7–9**	**10–11**	**12–13**	**14–15**
Frame type	Security enabled	Frame pending	Ack. request	Intra-PAN	Reserved	Dest. addressing mode	Reserved	Source addressing mode

Figura 3.4 – Quadro do *frame control*.

(Fonte: IEEE, 2003)

Após os bytes de *frame control* vem o byte pertencente ao *sequence number*, que serve como um endereço do dado que está sendo enviado e será incrementado a cada novo dado. A seguir tem-se o campo de endereçamento, no qual nos quatro primeiros bytes (5 ao 8) colocamos sucessivamente o PAN ID e o endereço do destinatário, e nos outros quatro bytes seguintes (9 ao 12) coloca-se o PAN ID e o endereço do remetente. Como citado anteriormente, o tipo de pacote utilizado foi apenas o de dados, portanto os bytes seguintes após o campo de endereçamento pertencem aos dados. Para este trabalho decidiu-se utilizar 16 bytes com a finalidade de tornar os testes mais próximos da realidade, como se possuíssem diversos dados de diferentes sensores para serem enviados.

Seguindo o código da figura 3.3, após definir os bytes que serão enviados é realizado, através de um laço *for*, a escrita na TX FIFO. Em seguida chama-se uma função que configura o rádio para o recebimento do ACK e dá início a transmissão dos dados. Na figura 3.5 temos o trecho de código pertencente a função *start_transmit*, na qual escrevemos o valor em binário

00000101 no registrador TXNCON, o terceiro bit habilita a requisição do ACK e o bit menos significativo dá início a transmissão do frame.

```
void start_transmit() {
  write_ZIGBEE_short(TXNCON, 0b00000101);
}
```

Figura 3.5 – Função que inicia a transmissão do dado.

3.4.1.1 Tempo de envio e espera do ACK

Levando em consideração que o rádio possui uma taxa de transmissão de 250kbps (MRF24J40 *Data Sheet*) e são enviados 29 bytes, totalizando 232 bits, ao fazer uma simples regra de três nós descobrimos quanto tempo demora para o pacote ser enviado:

$$Tempo\ de\ envio\ do\ pacote = \frac{232\ bits \cdot 1s}{250\ kbits} = 0{,}928\ ms \qquad (1)$$

Após o envio do pacote, o rádio aguarda o ACK por 0,912ms (valor padrão). Se não chegar é feito um novo envio, e isto se repete por três vezes até que o rádio desista de enviar (MRF24J40 *Data Sheet*). Contando as três tentativas do envio de dados mais as três esperas do ACK, o rádio ficaria 5,52ms tentando a comunicação antes de desistir e enviar um novo dado. Após essas três tentativas, o rádio é ajustado para tentar um novo dado 9ms depois. Somados os 9ms aos 5,52ms teríamos 14,52ms, que é o tempo máximo gasto em cada tentativa de envio. Este tempo de 9ms foi descoberto através de testes e é necessário para garantir o funcionamento correto do rádio. A figura 3.6 ilustra essas tentativas.

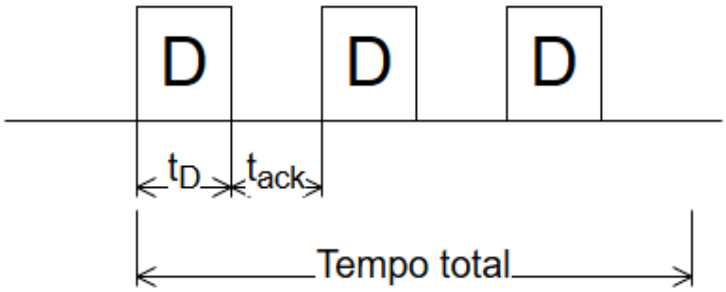


Figura 3.6 – Tempo de envio de dados e espera do ACK.

$t_D = 0{,}928ms$, tempo para enviar o pacote de dados.
$t_{ack} = 0{,}912ms$, tempo de espera do ACK.
Tempo total = 5,52ms

3.4.2 Pacote de recepção

Como no tópico anterior, a descrição do funcionamento do pacote de recepção será feita com base no código implementado para o mesmo. A figura 3.7 demonstra o formato e o caminho do pacote de recepção e a figura 3.8 possui o código implementado para realizar esta tarefa.

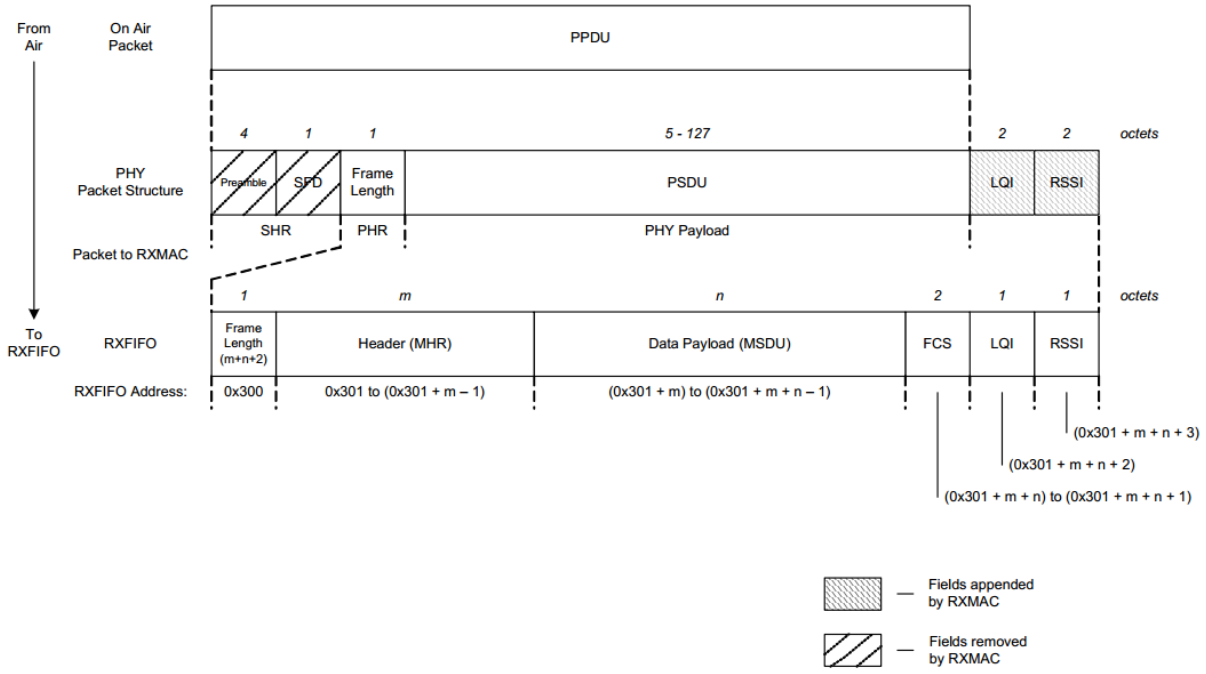


Figura 3.7 – Pacote de recepção.

(Fonte: MRF24J40 *Data Sheet*)

```
void read_RX_FIFO() {
  unsigned short int temp = 0;
  int i = 0;

  temp = read_ZIGBEE_short(BBREG1);      // disable receiving packets off air.
  temp = temp | 0x04;                    // mask for disable receiving packets
  write_ZIGBEE_short(BBREG1, temp);

  for(i=0; i<128; i++) {
    if(i <  (1 + DATA_LENGHT + HEADER_LENGHT + 2 + 1 + 1))
        // reading valid data from RX FIFO
        data_RX_FIFO[i] = read_ZIGBEE_long(address_RX_FIFO + i);
    if(i >= (1 + DATA_LENGHT + HEADER_LENGHT + 2 + 1 + 1))
        // reading invalid data from RX FIFO
        lost_data = read_ZIGBEE_long(address_RX_FIFO + i);
  }

  SN =  data_RX_FIFO[3];      //read sequence number

  DATA_RX[0] = data_RX_FIFO[HEADER_LENGHT + 1];              // coping valid data
  DATA_RX[1] = data_RX_FIFO[HEADER_LENGHT + 2];              // coping valid data
  DATA_RX[2] = data_RX_FIFO[HEADER_LENGHT + 3];              // coping valid data
  DATA_RX[3] = data_RX_FIFO[HEADER_LENGHT + 4];              // coping valid data
  DATA_RX[4] = data_RX_FIFO[HEADER_LENGHT + 5];              // coping valid data
  DATA_RX[5] = data_RX_FIFO[HEADER_LENGHT + 6];              // coping valid data
  DATA_RX[6] = data_RX_FIFO[HEADER_LENGHT + 7];              // coping valid data
  DATA_RX[7] = data_RX_FIFO[HEADER_LENGHT + 8];              // coping valid data
  DATA_RX[8] = data_RX_FIFO[HEADER_LENGHT + 9];              // coping valid data
  DATA_RX[9] = data_RX_FIFO[HEADER_LENGHT + 10];             // coping valid data
  DATA_RX[10] = data_RX_FIFO[HEADER_LENGHT + 11];            // coping valid data
  DATA_RX[11] = data_RX_FIFO[HEADER_LENGHT + 12];            // coping valid data
  DATA_RX[12] = data_RX_FIFO[HEADER_LENGHT + 13];            // coping valid data
  DATA_RX[13] = data_RX_FIFO[HEADER_LENGHT + 14];            // coping valid data
  DATA_RX[14] = data_RX_FIFO[HEADER_LENGHT + 15];            // coping valid data
  DATA_RX[15] = data_RX_FIFO[HEADER_LENGHT + 16];            // coping valid data

  LQI   = data_RX_FIFO[1 + HEADER_LENGHT + DATA_LENGHT + 2];  // coping valid data
  RSSI2 = data_RX_FIFO[1 + HEADER_LENGHT + DATA_LENGHT + 3];  // coping valid data

  temp = read_ZIGBEE_short(BBREG1);      // enable receiving packets off air.
  temp = temp & (!0x04);                 // mask for enable receiving
  write_ZIGBEE_short(BBREG1, temp);
}
```

Figura 3.8 – Código utilizado para a leitura do pacote na fila de recepção.

Quando o transmissor envia um pacote causa uma interrupção no receptor, fazendo que o código do mesmo entre na função *read_RX_FIFO* da figura 3.8. A primeira ação desta função é desabilitar o recebimento de novos pacotes pelo ar, evitando novas interrupções. Em seguida,

um laço *for* faz a leitura do pacote que foi recebido e que está na lista RX FIFO. Como pode-se perceber, a leitura possui condições que indicarão quais são os dados válidos, ou seja, aqueles enviados pelo transmissor, e quais são os dados inválidos, que não terão utilidade. É feita a implementação do código que faz a leitura dos dados inválidos, pois o frame que chega com a transmissão possui o tamanho máximo da camada física de dados disponível pelo rádio. Esta camada denominada *PLCP Service Data Unit* (PSDU) pode ser vista na figura 3.7. Percebe-se que, dentro da condição do laço, o tamanho do pacote é representado pela seguinte soma: (1 + *DATA_LENGHT* + *HEADER_LENGHT* + 2 + 1 + 1). O primeiro '1' representa o tamanho do *Frame Lenght*; *HEADER_LENGHT e DATA_LENGHT* representam o tamanho do cabeçalho e da quantidade de dados do pacote; e a soma "2 + 1 + 1" é a representação do *Frame Check Sequence* (FCS), do *Link Quality Indication* (LQI) e do RSSI, respectivamente.

Encerrando esse processo de leitura da RX FIFO, armazena-se os valores em variáveis para facilitar seu tratamento no decorrer do código e, então, habilita-se outra vez o recebimento de pacotes pelo ar, tornando o rádio disponível para a comunicação novamente.

3.5 *Duty Cycling* Monocanal

A técnica *Duty Cycling*, como já explicado anteriormente, trabalha com um regime de "liga e desliga". Neste trabalho serão utilizados dois tipos de rádio, um ficará ligado ao computador, recebendo os dados dos pacientes e não precisará da utilização do *Duty Cycling*, visto que o computador que armazena os dados ficará sempre ligado, alimentando o rádio. O outro rádio coletará os sinais vitais do paciente e para melhor comodidade será um rádio portátil, portanto necessitará da técnica *Duty Cycling* para aumentar a durabilidade da bateria. A figura 3.9 ilustra esse ambiente com os dois rádios e, para facilitar o entendimento da técnica, trataremos os rádios com a nomenclatura utilizada nesta imagem (sensor e computador).

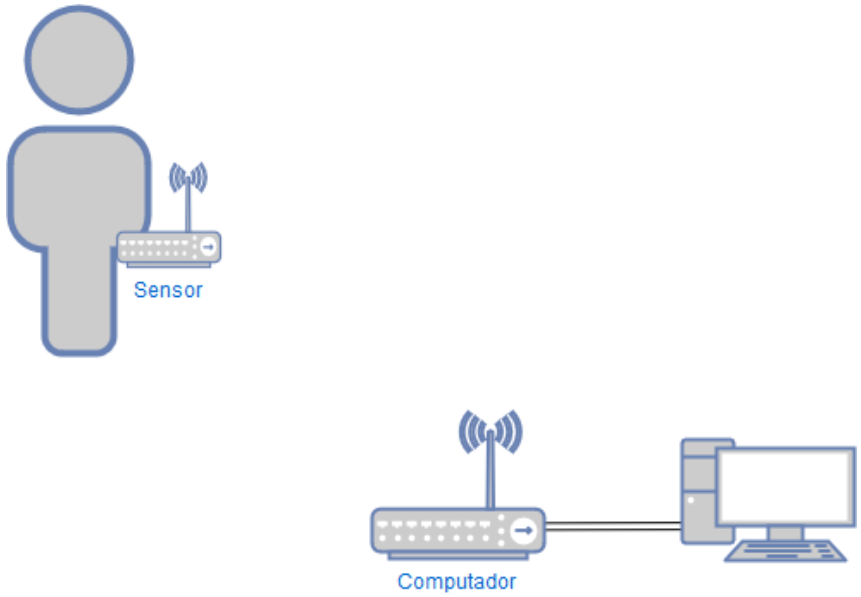


Figura 3.9 – Paciente com rádio sensor e central com rádio ligado ao computador.

No primeiro capítulo citou-se o *ContikiMAC* como base da elaboração deste trabalho, no qual o receptor faz uso da técnica para economia de energia e, periodicamente, acorda para verificar se o transmissor está tentando enviar algo. Neste trabalho o papel deve ser invertido, o sensor que envia os dados é quem faz uso da técnica *Duty Cycling*. Porém, não adianta o sensor ligar de tempo em tempo e enviar dados sem saber se o computador quer recebê-los, pois isto faria ele perder energia à toa.

Para contornar esta situação elaborou-se uma lógica de inversão de papéis, assim como no protocolo utilizado no *ContikiMAC*, no qual o sensor que faz uso da técnica *Duty Cycling* iniciará como receptor e ficará aguardando um pedido do computador. Se o computador quiser os dados ele transmitirá um pacote como aviso e esperará o reconhecimento. Caso venha o ACK, ele saberá que o sensor está ligado e apto para transmitir os dados do paciente. Desta forma, o computador inverte o seu papel e vira um receptor, enquanto o sensor, ao perceber que recebeu um pacote, saberá que precisa enviar um dado para o computador, então torna-se transmissor. Para esclarecer esta lógica, a figura 3.10 apresenta o diagrama de tempo da interação entre os rádios.

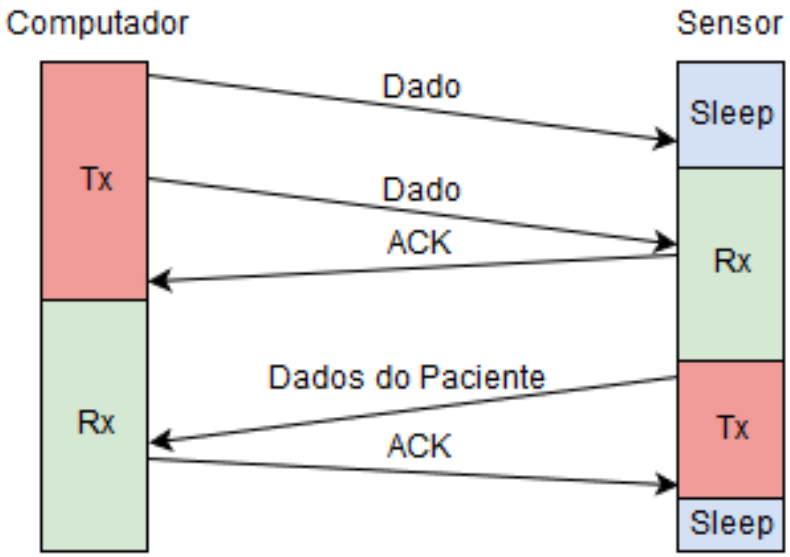


Figura 3.10 – Sucesso na comunicação entre o sensor e o computador.

Recapitulando com base na figura 3.10, toda vez que o computador quiser os dados de um determinado sensor, começa a transmitir pacotes de dado até encontrá-lo acordado e receber o ACK. Ao receber o ACK, o computador saberá que o sensor acordou, então passará para o modo receptor com a intensão de receber o dado do paciente. O sensor, ao receber um dado do computador, espera por mais um tempo para saber se o ACK enviado chegou e, então, o computador já estará pronto para recebê-lo. Esta espera é extremamente importante, pois se o ACK não chegar o computador continuará transmitindo os dados e não trocará para receptor. Após a espera, ao perceber que o computador parou de transmitir dados, o sensor muda para transmissor e envia os dados do paciente. É importante ressaltar que o sensor só tenta enviar um dado (lembrando que cada dado possui três tentativas) e volta pra *sleep*. A figura 3.11 ilustra um caso de erro no recebimento do ACK, no qual percebemos que o sensor recebeu um dado e ficou esperando se receberia outro, até que o dado parou de chegar, tornando-o apto a mudar para transmissor.

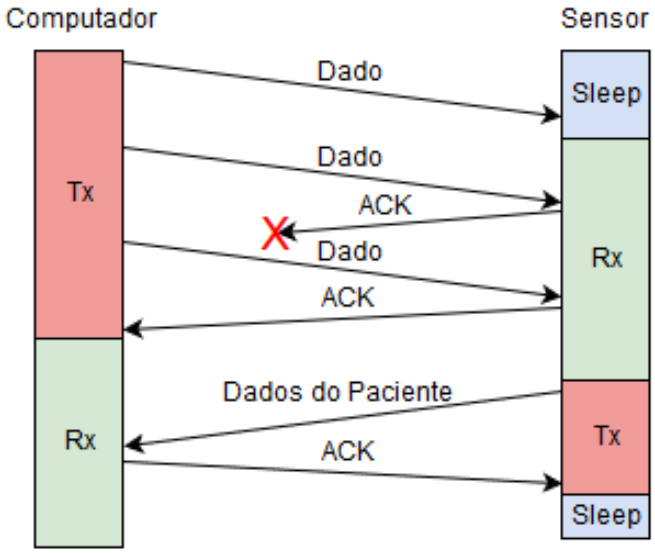


Figura 3.11 – Erro no ACK da comunicação entre o sensor e o computador.

3.5.1 Tempo de *sleep*

A quantidade de tempo escolhido para o rádio ficar ligado é entre 4% e 13% do tempo, garantindo em média 90% de economia de energia em relação a um rádio que opera o tempo inteiro. O rádio sensor liga e fica 14,56ms aguardando a comunicação, isto cobre o tempo de envio máximo do dado, calculado no tópico 3.4.1.1 como 14,52ms. Caso ele consiga comunicação, aguarda mais 6,37ms para garantir que o ACK chegou ao computador e mudar para o modo transmissor. Para fazer a inversão de modo são necessários 13ms para redefinir as configurações iniciais, e então, leva-se mais 9ms para começar a enviar o dado do paciente, mais 0,928ms para enviar o dado e 0,912ms para esperar o ACK e aí voltar a desligar. Desta forma, caso o sensor ligue e consiga comunicação, ele ficará o tempo máximo de 44,73ms acordado, considerando que precisou dos 14,56ms para conseguir se comunicar.

O tempo que o sensor fica em modo *sleep* é de 306ms, portanto, caso o rádio precise dos 44,73ms, ficará acordado 12,75% do tempo (306ms + 44,73ms = 350,73ms). Em compensação, se o computador não requisitar transmissão no tempo em que ele ficar ligado, o sensor ficará apenas 14,56ms, que em relação a soma entre os 306ms de *sleep* mais os 14,56ms ligado daria apenas 4,54% do tempo acordado. As figuras 3.12 e 313 esclarecem essas duas condições.

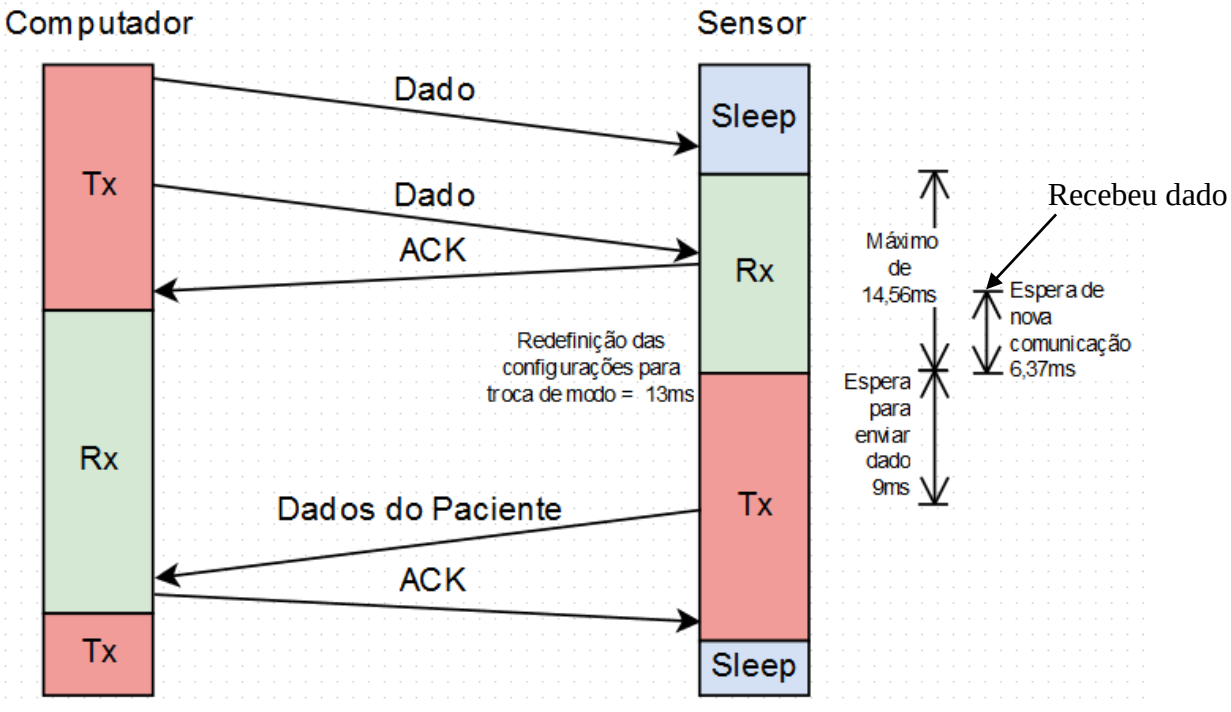


Figura 3.12 – Diagrama de tempo com os tempos utilizados no trabalho.

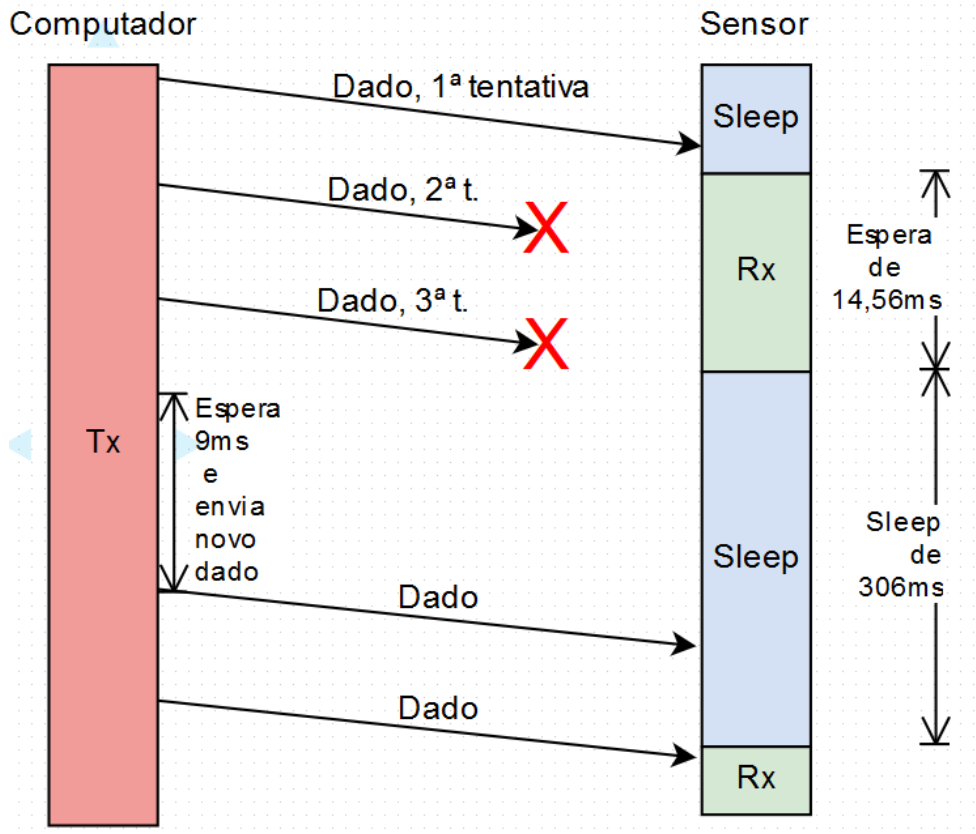


Figura 3.13 – Diagrama de tempo com os tempos utilizados no trabalho e erro no ciclo de *wake-up*.

3.6 *Duty Cycling* Multicanal

Manter um rádio sempre no mesmo canal pode ser bom quando há pouca interferência, mas neste mesmo canal pode ocorrer migração de outros rádios e, então, ele sofrerá com a interferência dos diversos sinais que estarão usando a mesma faixa de frequência. A alternativa para manter este rádio sempre na média entre o pior e o melhor canal é ficar trocando o seu canal de operação. Utilizou-se esta técnica de troca de canal junto com a *Duty Cycling* explicada no tópico anterior, e para que os dois rádios operassem sempre no mesmo canal o sensor foi escolhido para ditar qual será o canal da próxima transmissão.

Como já foi explicado anteriormente, quando o computador desejar os dados do sensor, ele passará a emitir pacotes até achá-lo acordado. Quando isso acontecer, o computador trocará para o modo de recepção e o sensor para o modo de transmissão. Junto com a transmissão dos dados do paciente o sensor enviará um byte responsável pela escolha do novo canal. Esta escolha é feita sempre incrementando o valor do canal atual, e ele só se conectará neste novo canal quando voltar a acordar no próximo ciclo, portanto o computador também trocará o canal quando voltar ao modo de transmissão. A figura 3.14 ilustra este mecanismo.

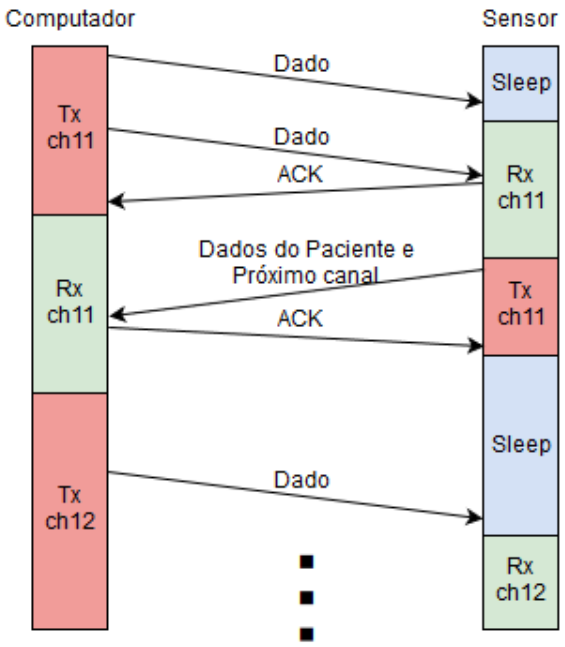


Figura 3.14 – Caso de sucesso do *Duty Cycling* Multicanal.

Caso ocorra um problema na transmissão do pacote e o novo canal não chegue para o computador, acontecerá uma falha na sincronia dos rádios, pois o computador permanecerá no canal antigo e o sensor passará para o novo. Para resolver este problema, realizou-se um mecanismo de recuperação, no qual depois de um tempo, quando o computador perceber que já tentou 85 vezes se comunicar sem conseguir, ele trocará para o próximo canal e continuará varrendo os canais até achar o correto. Porém, se o sensor acabou ficando em um canal com muito ruído, dificilmente o computador conseguirá comunicação tornando incerta a sincronização. Para contornar este problema o sensor sempre guarda o canal que teve menor RSSI, ou seja, menor ruído, e então depois de 200 ciclos de *wake-up* sem comunicação, ele trocará para o canal de menor RSSI e facilitará a sincronização. As figuras 3.15 e 3.16 esclarecem estes casos.

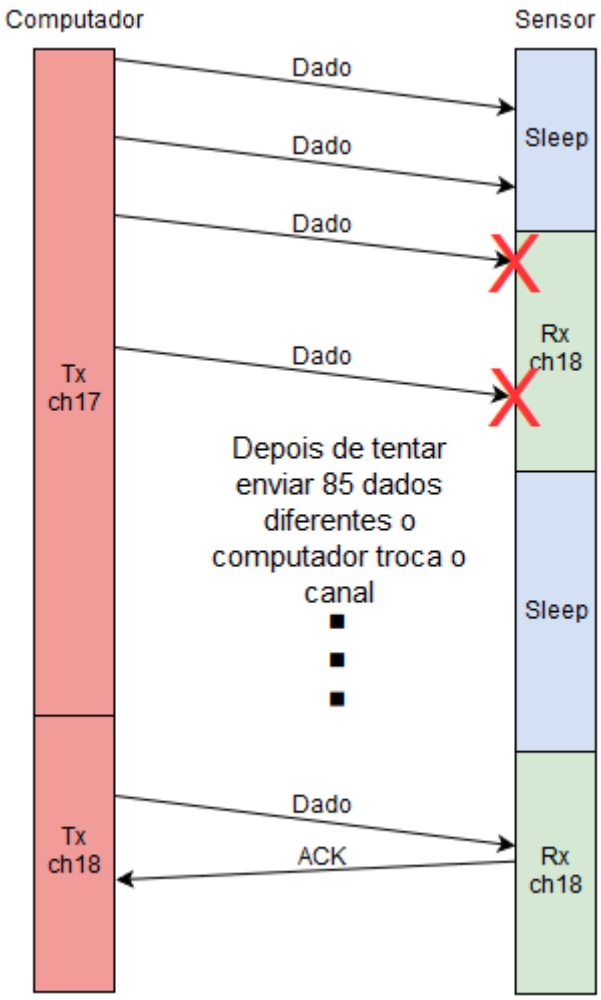


Figura 3.15 – Computador varrendo o canal até achar o correto.

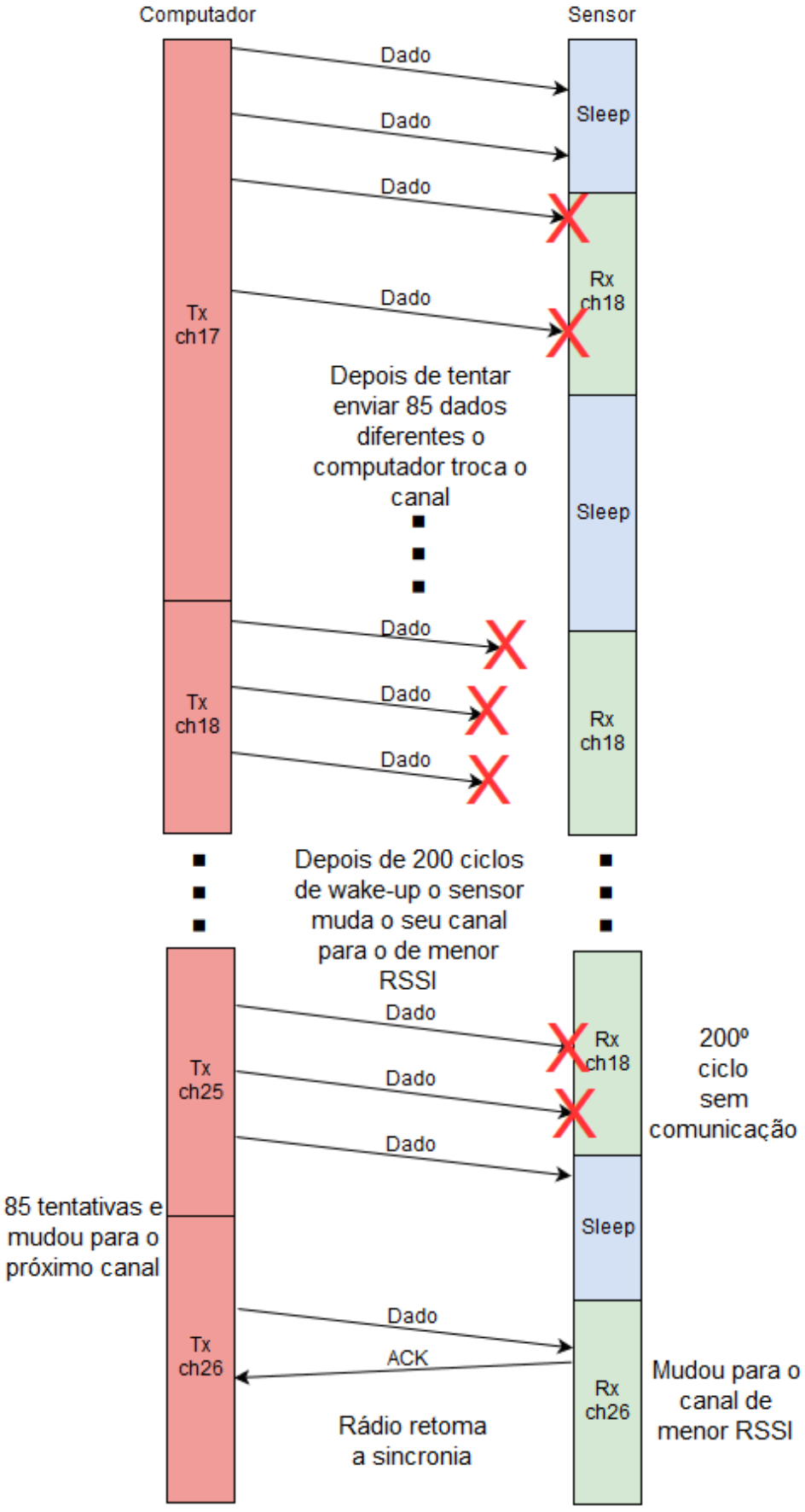


Figura 3.16 – Sensor muda para canal de menor RSSI.

4. RESULTADOS

4.1 Introdução

Este capítulo servirá exclusivamente para demonstrar e analisar os resultados obtidos da quantidade de envios necessários para conseguir comunicação, tempo médio de comunicação e consumo de energia. Todos os testes foram realizados em uma sala que contém cerca de 20 computadores e, para conseguir resultados melhores, os rádios foram postos em linha reta e sem obstáculos entre eles. Para observar o desempenho em diferentes situações, efetuou-se os testes em distâncias de 2m, 5m e 10m entre os rádios; o horário e o dia da semana não foram considerados, porém, buscou-se obter apenas os resultados em turnos com tráfego normal da rede.

Para fazer uma análise precisa dos dados coletados é importante conhecer quais são os canais com maior interferência na rede. Os sinais de Wi-Fi (IEEE 802.11) ocupam a mesma banda de frequência do rádio (IEEE 802.15.4), portanto interferem em seu sinal. O software *Acrylic*, de análise da rede Wi-Fi, foi adotado para observar quantos roteadores ocupavam determinados canais e qual a sua força dentro da sala. A figura 4.1 possui comparação feita pela Microchip (2010) da faixa de frequência dos 16 canais do rádio com os 3 canais mais utilizados entre os 11 da rede Wi-Fi.

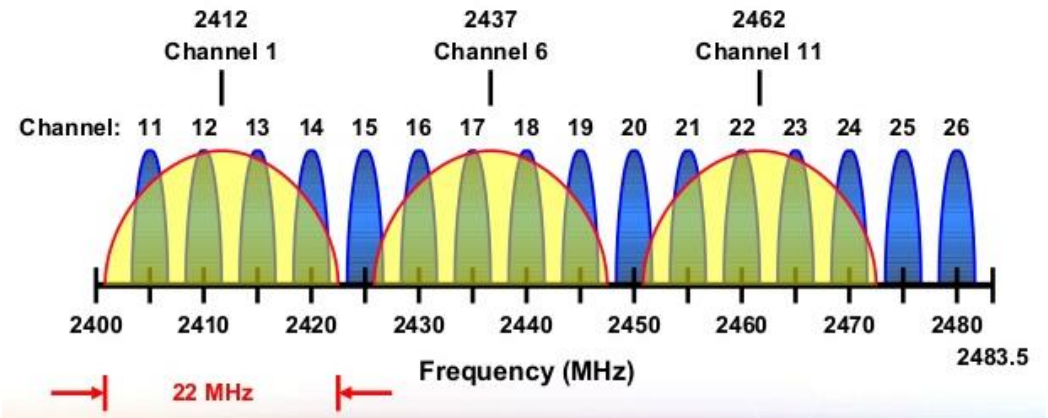


Figura 4.1 – IEEE 802.11 vs. IEEE 802.15.4.

(Fonte: Microchip 2010 Masters Conference)

As análises se darão da seguinte forma: primeiro, no tópico 4.2, será feita a análise do desempenho do rádio utilizando a técnica explicada na seção 3.5 *Duty Cycling* Monocanal, e também será preciso considerar o rádio operando em três canais diferentes. No tópico 4.3, serão estudados os testes feitos com a técnica *Duty Cycling* Multicanal, com 16, 8, 4 e 2 canais. Na seção 4.4 será feita a análise do consumo de energia e, por fim, na seção 4.5 serão comparadas as duas técnicas implementadas neste trabalho.

4.2 Desempenho da técnica *Duty Cycling* Monocanal

Para realizar os testes de entrega dos pacotes programou-se o rádio computador para fazer um número máximo de 100 tentativas em torno de 1,5 segundos. Desta forma, a cada novo dado que o computador tenta enviar para conseguir comunicação, ele incrementa o *sequence number*. Ao conseguir comunicação e receber o pacote de dados do sensor, o computador encerra o envio de dados e aguarda o término do tempo de 1,5 segundos. Após este intervalo é gravada na memória EEPROM do microcontrolador a quantidade de dados que precisaram ser enviados até conseguir se comunicar, e assim que terminar de gravar, o computador volta a tentar uma nova comunicação. Isto se repete por 256 vezes até encher a memória EEPROM; para cada caso foram feitos dois testes, desta forma, coletou-se 512 dados de número de entregas necessárias para conseguir comunicação. As figuras 4.2 e 4.3 esclarecem a mecânica para coleta das amostras.

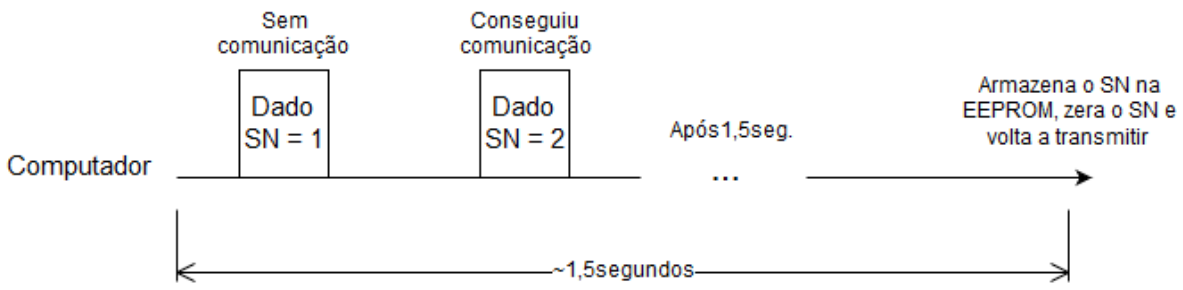


Figura 4.2 – Computador armazena quantos dados conseguiu enviar até ter sucesso.

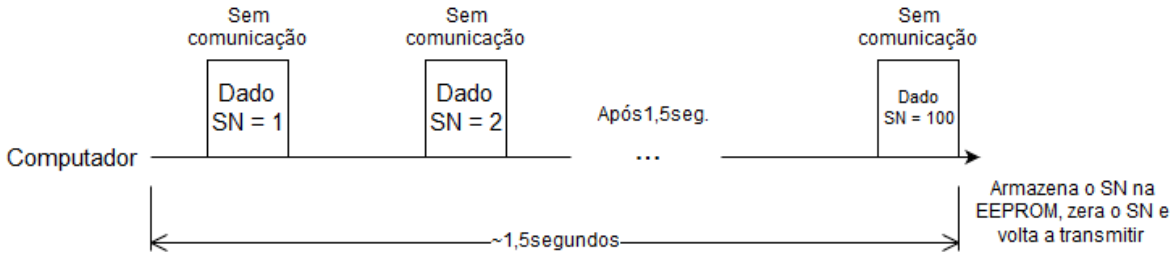


Figura 4.3 – Computador armazena quantidade máxima de tentativas, pois não conseguiu comunicação.

A seguir os testes de cada canal serão avaliados em diferentes tópicos e no final os resultados serão comparados. A escolha dos canais foi feita com base nos seguintes critérios: o canal 11 foi escolhido por possuir uma interferência mediana, o 17 por ser o mais vulnerável e optou-se pelo 26 por estar fora da faixa de frequência da rede Wi-Fi permitida pela Anatel (TELECO, 2005). Considerando que a análise do desempenho do rádio é focada exclusivamente para as técnicas *Duty Cycling* desenvolvidas, foi testado apenas um canal no modo sem *sleep*, somente para se ter uma ideia da diferença.

4.2.1 Canal 11

O canal 11, como pode-se observar na figura 4.1, sofre com a interferência do canal 1 da rede IEEE 802.11. Com o auxílio do software *Acrylic* foi possível identificar os roteadores próximos e quais faixas de frequência eles ocupavam. Na figura 4.4 visualiza-se que seis dispositivos estão utilizando o canal 1 e interferindo no desempenho do rádio, em especial a rede denominada TP-LINK_C902EC com RSSI médio de -57dB. Observa-se que este foi o roteador que deu mais prejuízo no desempenho da comunicação.

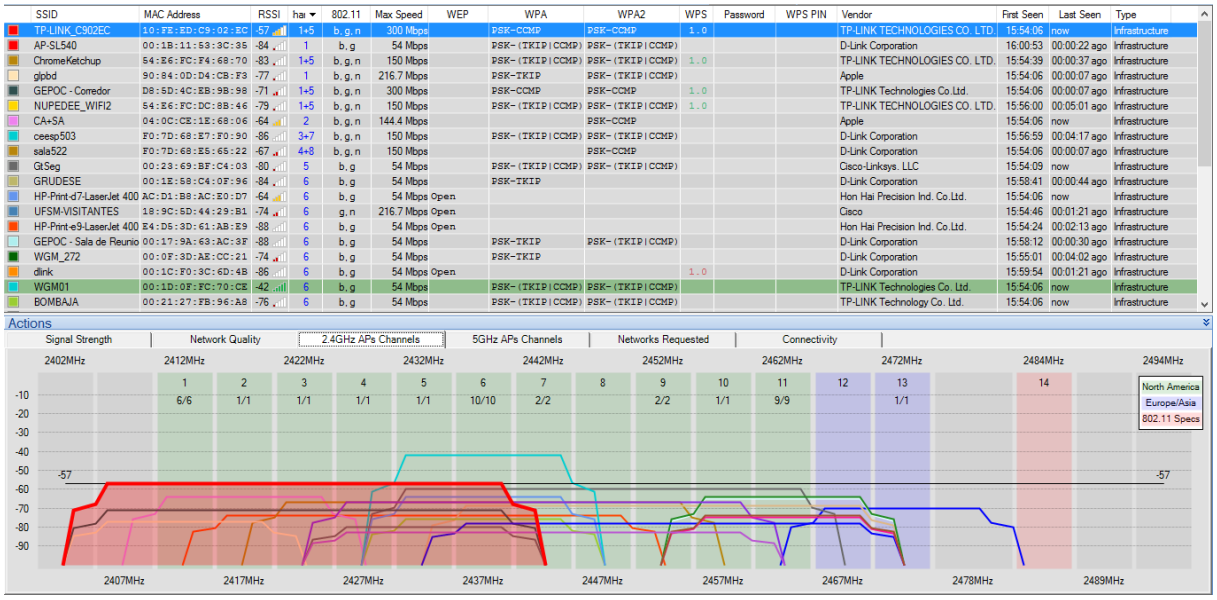


Figura 4.4 – Roteadores Wi-Fi que interferiram nos testes do canal 11.

Nos gráficos abaixo examina-se os resultados obtidos durante os testes, lembrando que o número de transmissão equivale a quantidade de dados enviados e necessários para realizar a comunicação. Como pode-se observar, muitas vezes o rádio precisou de apenas duas tentativas para conseguir transmitir, e este valor pode ser chamado de ponto estável, que ocorre devido a sincronia entre computador e sensor. Quando o ciclo de aproximadamente 1,5 segundos do computador terminar, ele volta a transmitir e geralmente envia o primeiro dado quando o sensor ainda está dormindo, e o segundo quando está acordado. Outras vezes o rádio precisou fazer mais de 10 tentativas, mas isto não quer dizer que ele falhou quando o sensor ligou, pois, esta variação é uma falha de sincronismo entre os rádios, causada principalmente por atraso no hardware. Por exemplo, em vez do computador voltar a enviar com o sensor no fim do *sleep*, ele o fez alguns milissegundos antes ou depois, deparando-se com o sensor na metade do tempo de dormir. Lembramos, então, que se o *sleep* do sensor tem 306ms e o computador leva 14,52ms em cada transmissão que falhou, teríamos um máximo de 21 tentativas enquanto o rádio estiver dormindo. Após estas 21 tentativas, na 22º encontraria o rádio ligado, ou seja, se o número de tentativas foi maior que 22, houve falha na transmissão durante o tempo de *wake-up* do sensor.

Recapitulando, se houver apenas duas tentativas, quer dizer que os rádios estão sincronizados; caso ocorra mais de duas e menos de 23 tentativas, os rádios perderam a sincronia, mas conseguiram se comunicar durante o tempo de *wake-up* do sensor. Porém, se houve mais de 22 tentativas, entende-se que aconteceram falhas na comunicação, e se chegou

a 100 tentativas, o computador não conseguiu se comunicar em momento algum durante aproximadamente 1,5 segundos.

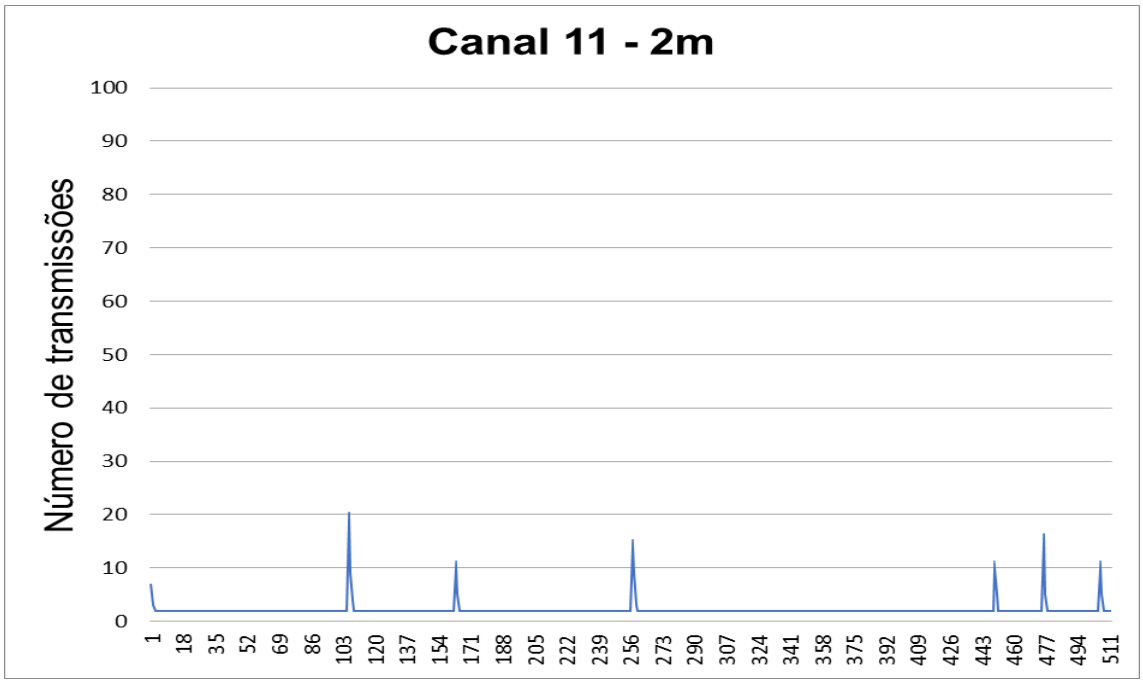


Figura 4.5 – Número de transmissões no canal 11 com 2m de distância entre os rádios.

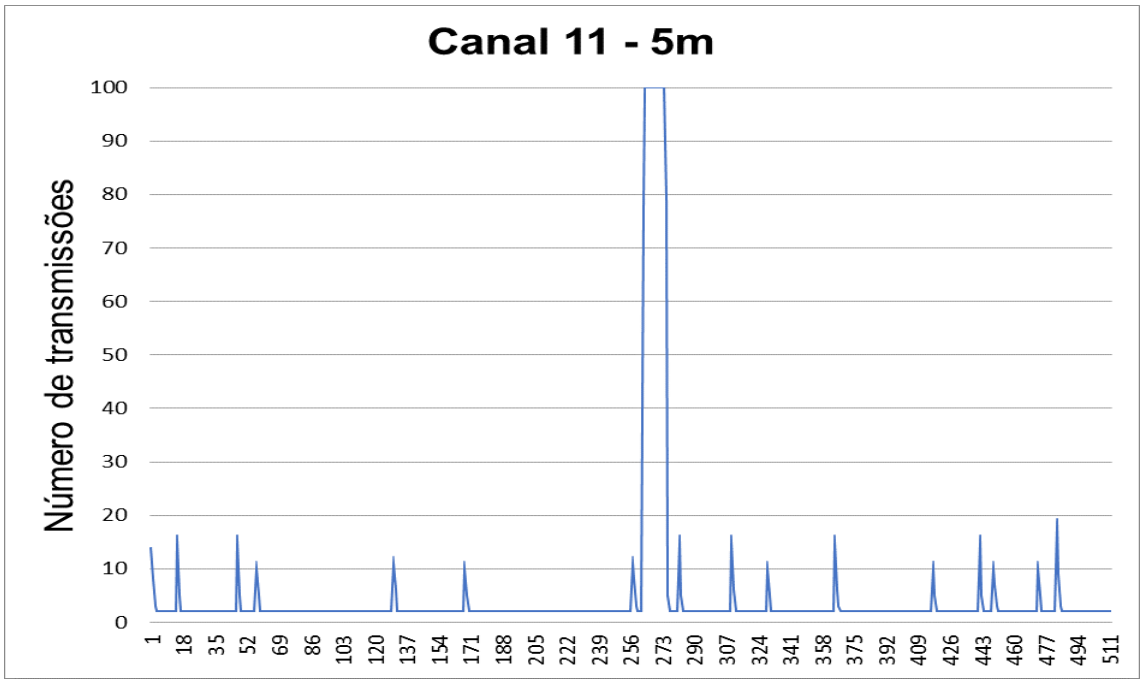


Figura 4.6 – Número de transmissões no canal 11 com 5m de distância entre os rádios.

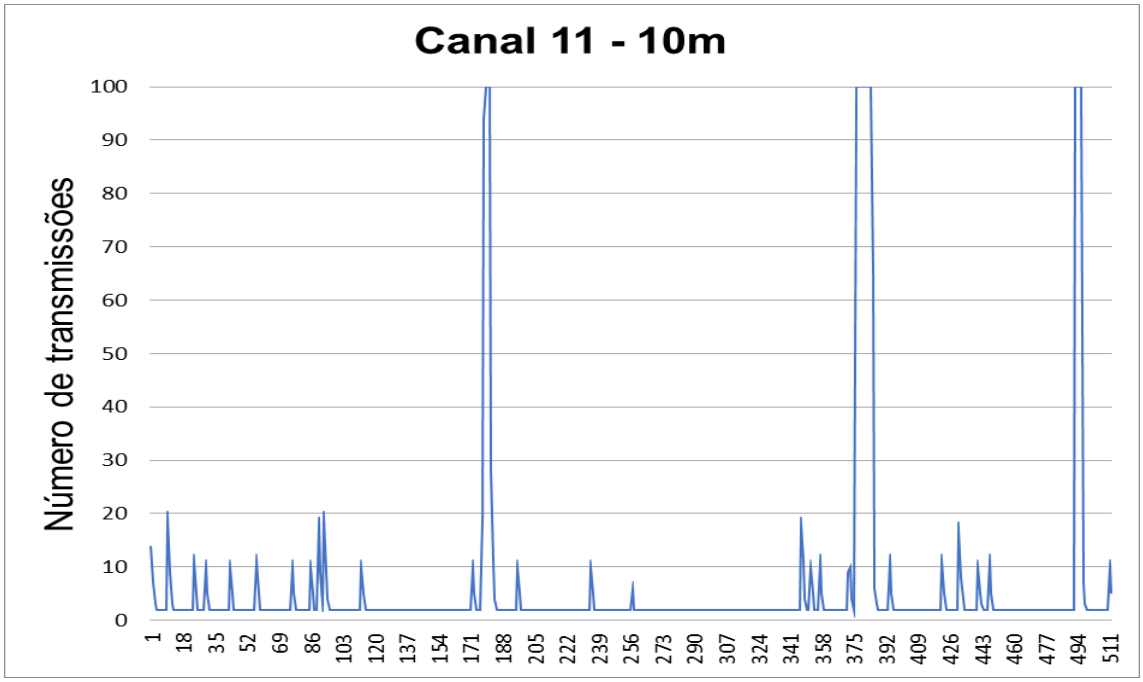


Figura 4.7 – Número de transmissões no canal 11 com 10m de distância entre os rádios.

Na figura 4.5 observa-se que o rádio não teve grandes problemas na comunicação. Em nenhuma vez necessitou fazer mais de 22 tentativas, portanto conseguiu enviar em toda vez que o sensor ligou. Além disso, manteve-se estável a maior parte do tempo, apenas em algumas vezes ocorreu falha no sincronismo, fazendo o computador enviar mais dados enquanto o sensor ainda estava dormindo. Já na figura 4.6, começa-se a notar os efeitos da distância no desempenho do rádio, que passou a perder mais dados e, em um certo instante, não foi possível realizar a comunicação. Além disso, nota-se que aumentou a quantidade de vezes que a comunicação saiu do ponto estável. Para finalizar, na figura 4.7, percebe-se que em três momentos o rádio não conseguiu realizar a comunicação, provando a previsão da perda do desempenho relacionada com a distância entre os rádios. Desta forma, concluímos que quanto maior for a distância, pior será o desempenho do rádio, pois a força do seu sinal é prejudicada e acaba sofrendo mais com a interferência. Outro problema que pode ser notado é que a distância e, ocasionalmente, a distorção do sinal causam atrasos e extravios no envio dos pacotes, fazendo que os rádios demorem mais para executar certas funções e transmitir os dados, e por isso sai do seu ponto estável.

4.2.2 Canal 17

Por ser o canal que sofre interferência do roteador mais próximo da sala de testes, o canal 17 deverá ter a maior quantidade de tentativas com erro. Visualizando a figura 4.1, nota-se que ele possui alto índice de perturbação, pois ocupa a mesma faixa de frequência do pico do canal 6 da rede Wi-Fi. Utilizando o software *Acrylic* para fazer uma nova análise da rede, percebe-se que o dispositivo WGM01 detêm o sinal mais forte e faz uso do canal 6, e isto pode ser visto na figura 4.8. Os gráficos dos testes demonstraram o quanto este sinal de -38dB é influente no desempenho da rede.

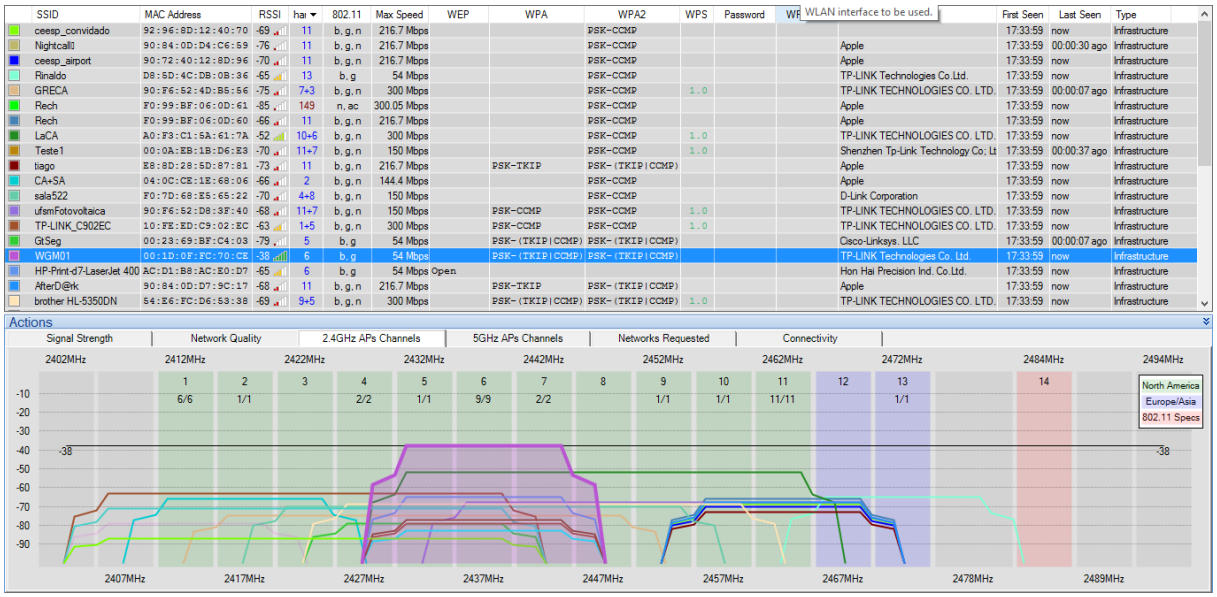


Figura 4.8 – Roteadores Wi-Fi que interferiram nos testes do canal 17.

No gráfico dos 2 metros, visto na figura 4.9, percebe-se a enorme instabilidade que há no canal. Apesar de ter conseguido se comunicar em todos os ciclos, em alguns pontos houve mais de 22 tentativas, caracterizando a falha da comunicação em pelo menos um ciclo de *wake-up* do sensor. Por exemplo, entre a 1ª e a 18ª amostra temos o pior caso, que necessitou de 27 tentativas para se comunicar. Então presumimos que o computador começou a enviar os dados quando faltava 58ms para o sensor acordar. Isto deu tempo para 4 tentativas, de 14,52ms cada, e ao acordar tentou-se mais uma vez, porém a interferência no canal impediu a comunicação e

o sensor voltou a dormir, fazendo o computador transmitir por mais 21 pacotes durante o *sleep* e, finalmente, conseguir comunicação no 27º dado enviado.

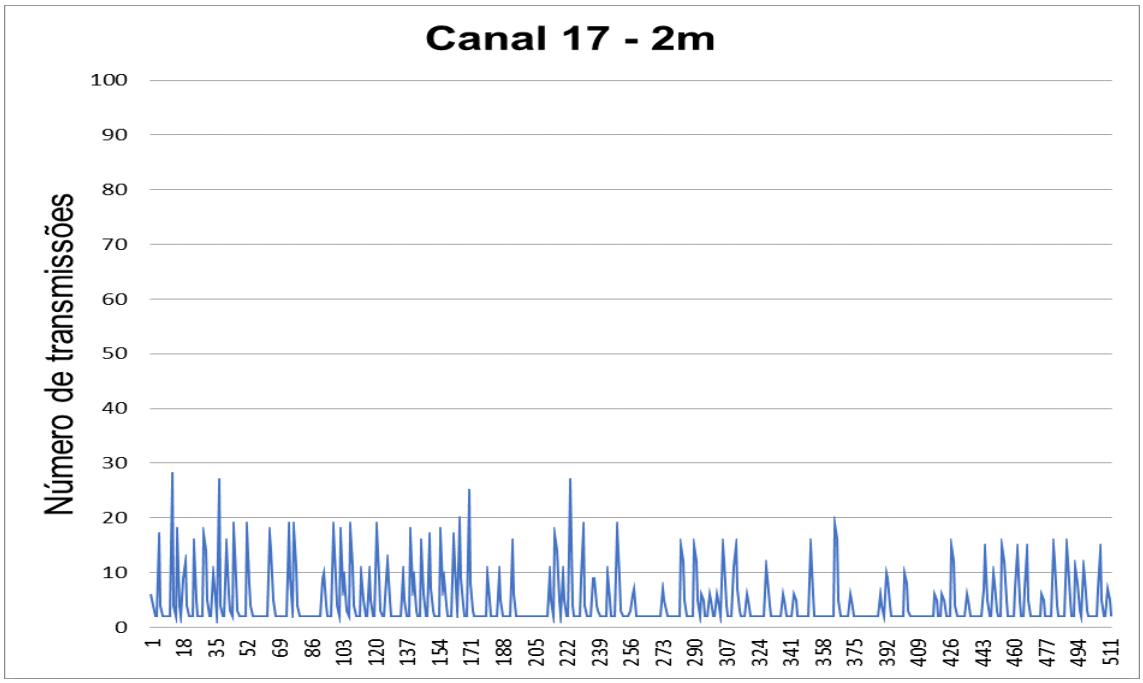


Figura 4.9 – Número de transmissões no canal 17 com 2m de distância entre os rádios.

A partir dos 2 metros, os gráficos de 5m e 10m (figuras 4.10 e 4.11, respectivamente) provam que o rádio fica impróprio para a comunicação quando opera na mesma banda de um roteador próximo. Nota-se que em cerca de 1/3 do tempo o computador não conseguiu transmissão na distância de 5 metros e em 10 metros este tempo dobrou. Provamos então o que foi explicado no capítulo 2 sobre a arquitetura *low-IF*, que por necessitar de um casamento perfeito entre amplitude e fase, faz com que a interferência de sinais com potências próximas ou maiores que a do rádio diminua o seu desempenho (SOARES, 2008).

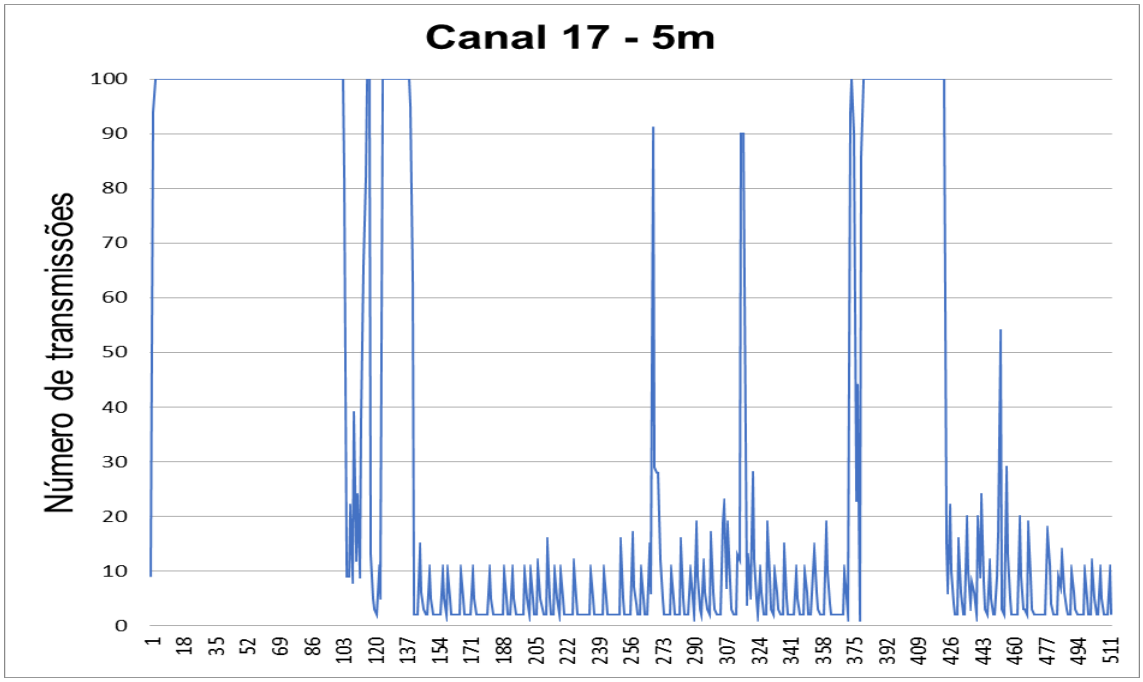


Figura 4.10 – Número de transmissões no canal 17 com 5m de distância entre os rádios.

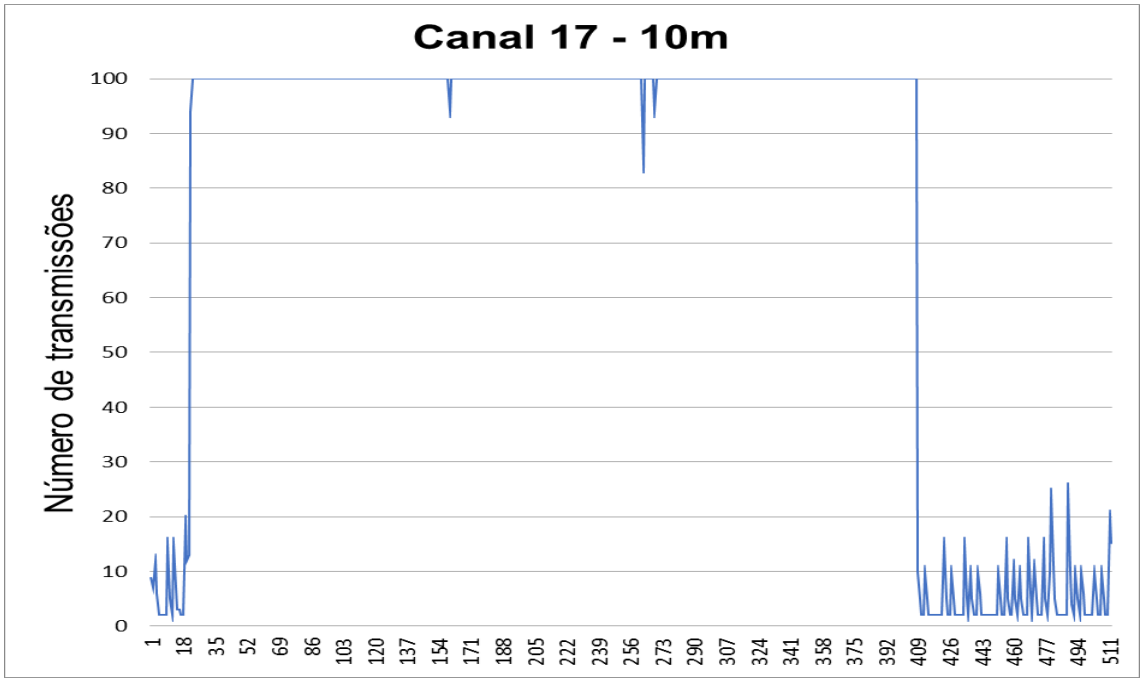


Figura 4.11 – Número de transmissões no canal 17 com 10m de distância entre os rádios.

Durante a realização dos testes de 10 metros do canal 17, percebeu-se que não é simplesmente a potência do sinal que interfere no seu desempenho. O tráfego na rede está diretamente relacionado a quantidade de falhas na comunicação, por exemplo, na figura 4.12

vemos o gráfico de um teste realizado ao meio dia, horário em que os usuários estão almoçando e a utilização da internet está abaixo do normal. Podemos perceber que o seu desempenho foi tão bom quanto a 2 metros, com poucas falhas de transmissão. Outro momento que proporcionou esta percepção foi durante o desenvolvimento do trabalho, quando tudo estava funcionando perfeitamente e ao começar a carregar um vídeo na web os rádios foram perdendo comunicação. Por isso citou-se na introdução do capítulo que os testes foram realizados durante os turnos com tráfego normal da rede, afim de obter justiça na comparação.

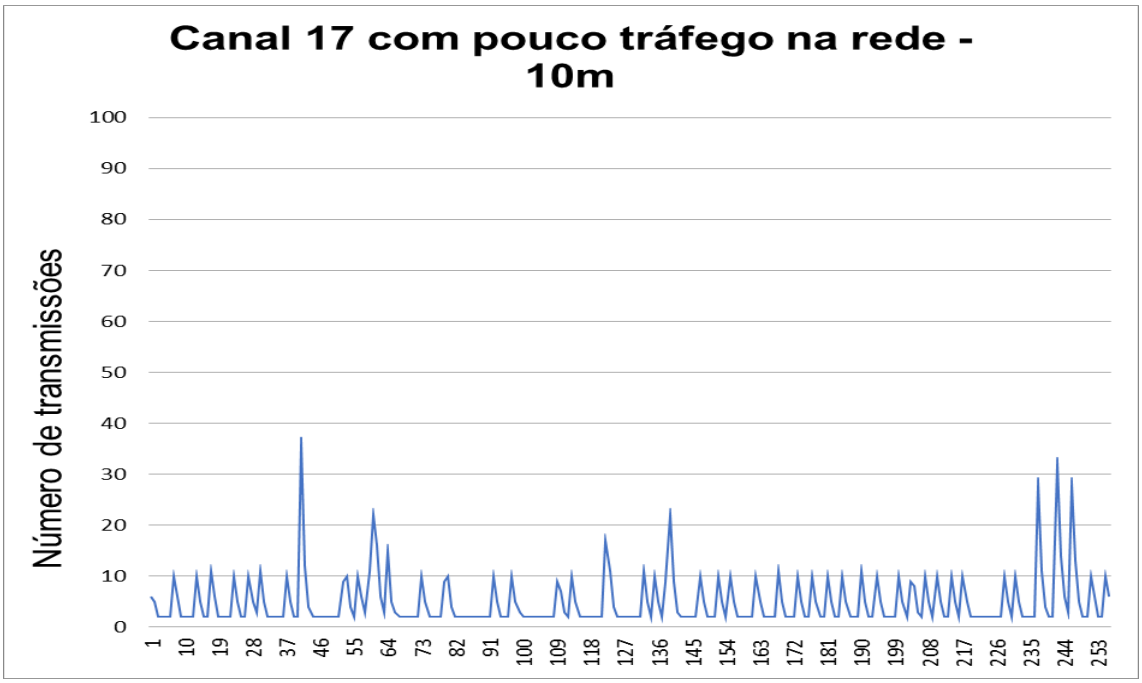


Figura 4.12 – Número de transmissões no canal 17 com pouco tráfego na rede.

4.2.3 Canal 26

O canal 26 do padrão IEEE 802.15.4 está fora da faixa de frequência da rede Wi-Fi norte-americana, a mesma adotada pela Anatel no Brasil. Desta forma, o esperado era obter resultados limpos e sem interferência de outros sinais, porém, o programa *Acrylic* captou uma rede utilizando o canal 13 do padrão IEEE 802.11, permitido apenas na Ásia e na Europa e, portanto, fora das leis impostas no Brasil. Apesar de ilegal, a rede deve ser considerada e pode ser vista na figura 4.13 com o nome Rinaldo e -68dB, pois possui menos força entre os sinais

que influenciaram nos testes do canal 11 e 17. Desta forma, como será demonstrado nos gráficos, este canal foi o que obteve melhores resultados.

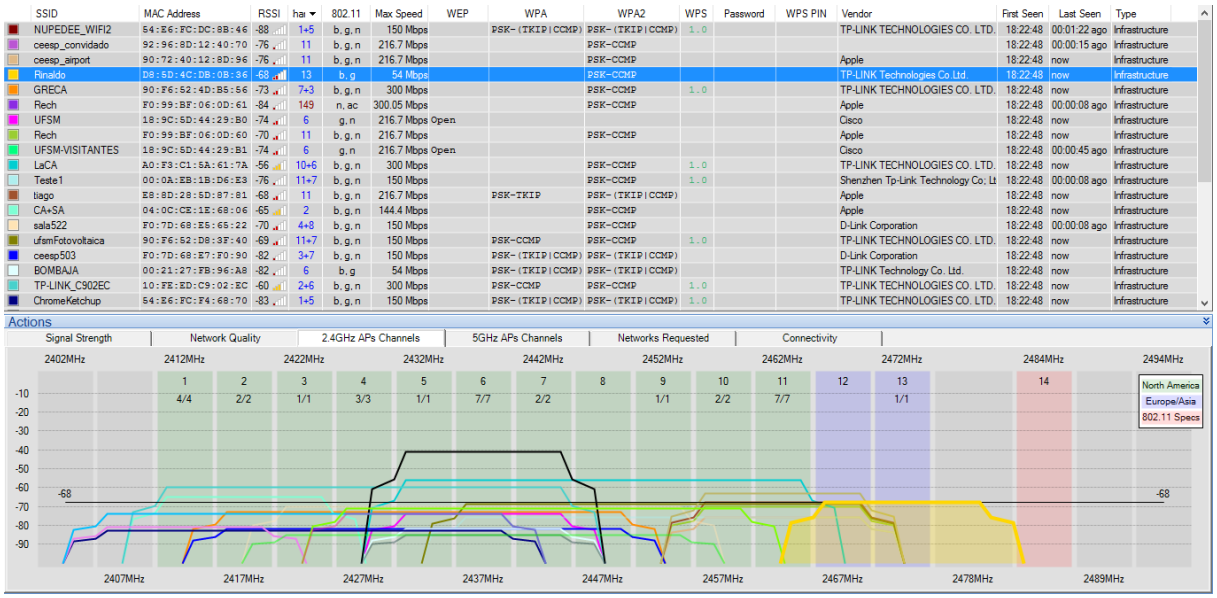


Figura 4.13 – Roteadores Wi-Fi que interferiram nos testes do canal 26.

No teste de 2 metros (figura 4.14) o rádio obteve um bom desempenho e manteve-se quase o tempo inteiro estável, conseguindo comunicação na segunda tentativa. O computador comunicou-se em todos os ciclos, apenas algumas vezes teve pequenos atrasos que motivaram a necessidade de mais tentativas, sendo 10 a quantidade máxima. No teste de 5 metros, visto na figura 4.15, nota-se que o desempenho do rádio quase não mudou. Continuou a maior parte do tempo estável, apenas perdeu o sincronismo mais vezes em relação a análise de 2 metros. E por último, no teste de 10 metros (figura 4.16), ocorreram perdas de dados em alguns ciclos de *wake-up* do sensor, porém em todos os intervalos do computador conseguiu-se comunicação. Necessitando de 70 tentativas no pior instante, considerando que cada envio de pacote custe 14,52ms, calcula-se um tempo máximo de 1016,4ms que o rádio ficou sem conseguir se comunicar.

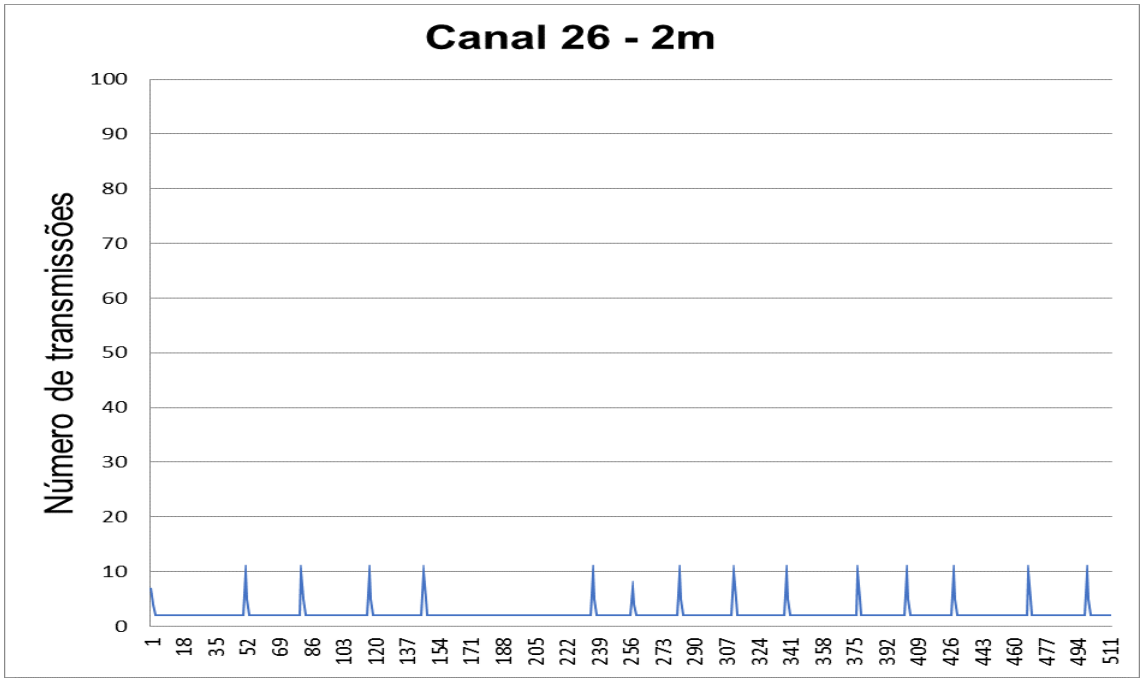


Figura 4.14 – Número de transmissões no canal 26 com 2m de distância entre os rádios.

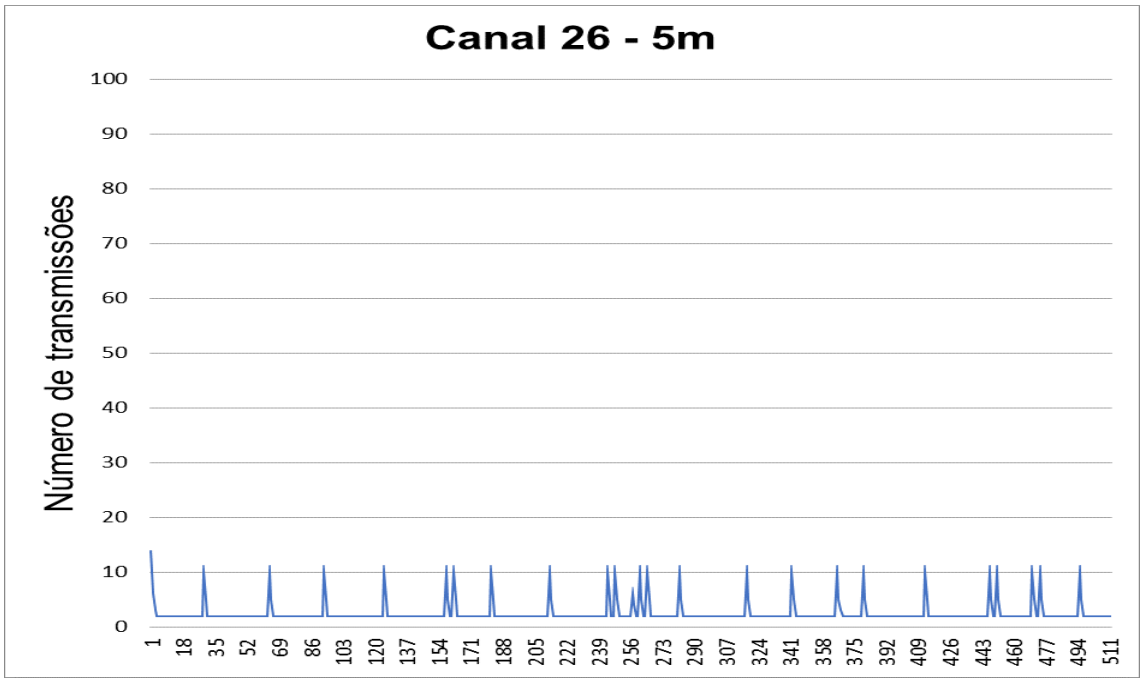


Figura 4.15 – Número de transmissões no canal 26 com 5m de distância entre os rádios.

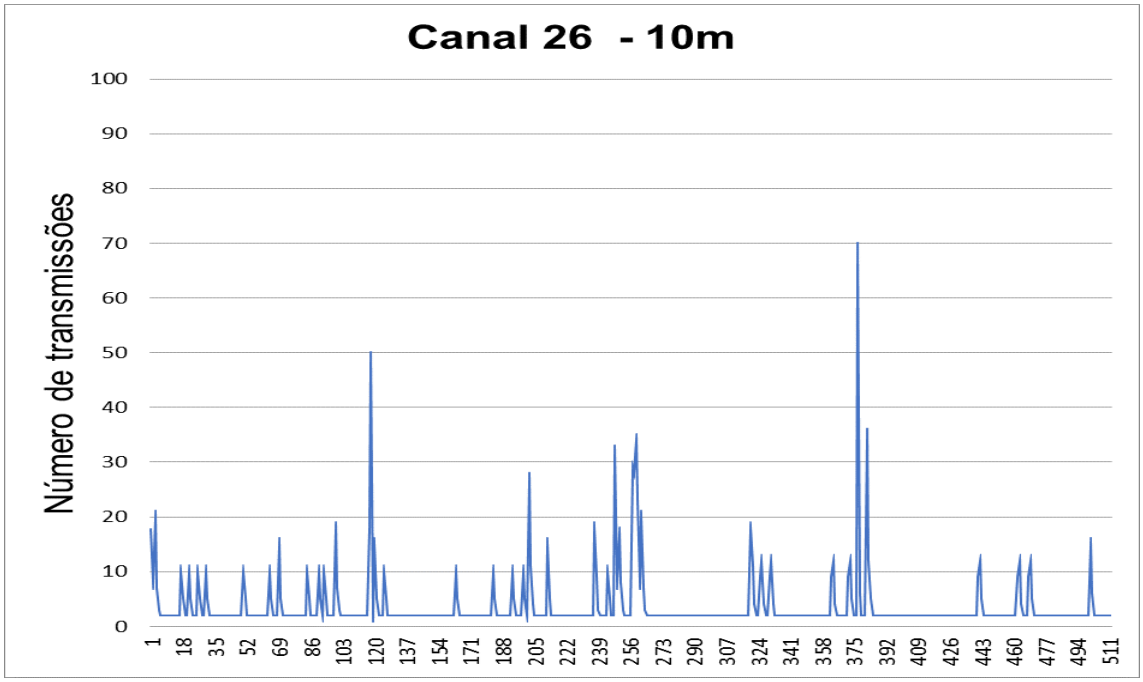


Figura 4.16 – Número de transmissões no canal 26 com 10m de distância entre os rádios.

4.2.4 Canal 11 sem *sleep*

Este é o único teste do rádio operando sem o uso do *Duty Cycling*, realizado apenas para ter uma base do quanto muda o seu desempenho. Demais experimentações não foram feitas, pois o foco principal do trabalho é analisar e comparar o desempenho da técnica implementada para economia de energia. Desta forma, tirou-se a condição que faz o sensor entrar em modo *sleep*, mantendo-o sempre ligado em modo de recepção, até identificar o pedido de comunicação e enviar os dados do paciente. Assim, o rádio tem o melhor desempenho possível, pois em todas as tentativas o computador acharia o sensor ligado, mudando o seu ponto estável para a primeira tentativa, fato que pode ser demonstrado pelos gráficos abaixo.

Tendo bom desempenho em todas as distâncias, com apenas alguns instantes de instabilidade, geralmente logo no segundo envio foi possível retomar a comunicação. Mesmo assim, ainda se percebe a perda de desempenho com a distância. Por exemplo, na figura 4.19 existem mais momentos com duas tentativas que na figura 4.17. É importante ressaltar que os testes foram realizados em dias diferentes dos testes do canal 11 com uso da técnica *Duty Cycling*, portanto deve-se descartar qualquer comparação direta entre eles em questão de

desempenho. Como foi dito anteriormente, estes testes servem apenas para ter uma ideia de como ficaria sem o *sleep*.

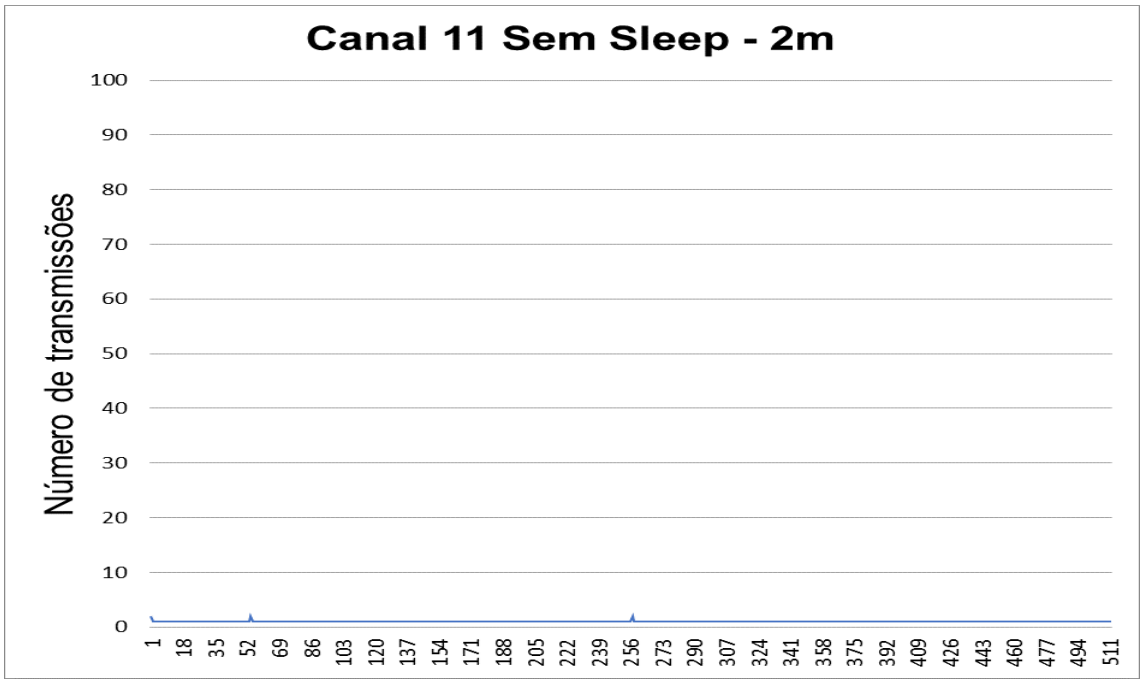


Figura 4.17 – Número de transmissões no canal 11 sem *sleep* com 2m de distância entre os rádios.

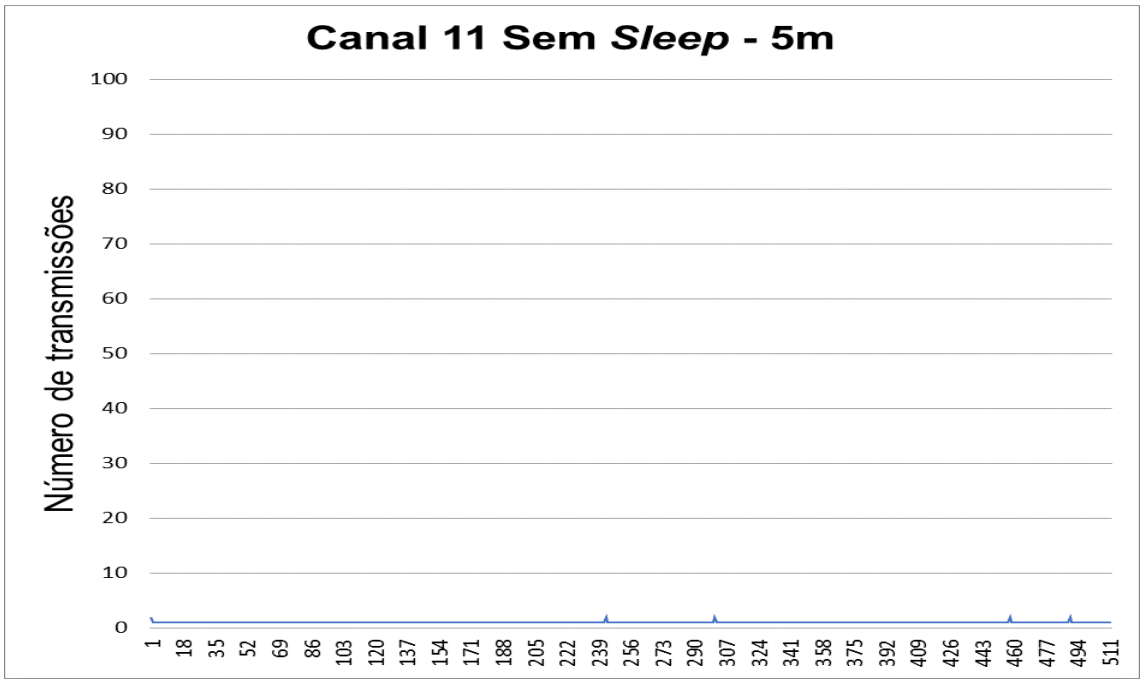


Figura 4.18 – Número de transmissões no canal 11 sem *sleep* com 5m de distância entre os rádios.

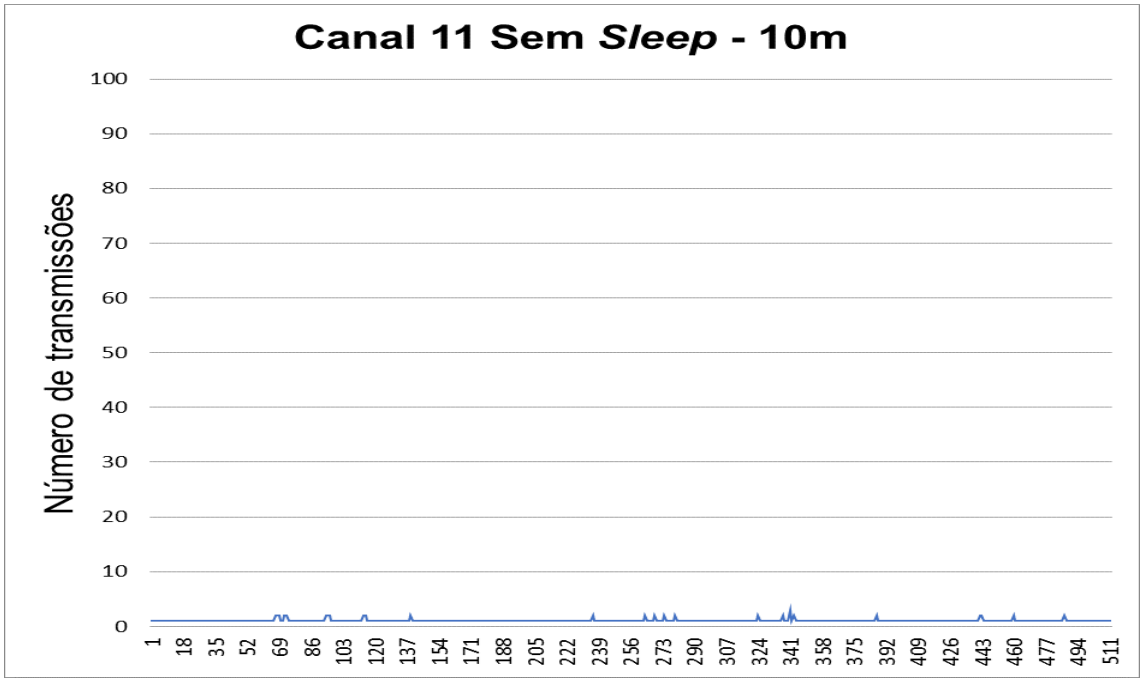


Figura 4.19 – Número de transmissões no canal 11 sem *sleep* com 10m de distância entre os rádios.

4.2.5 Comparações do tempo médio de comunicação

Para facilitar a comparação entre os testes foram realizados o cálculo da média e do desvio padrão do tempo médio de comunicação. Lembrando que os dados levam o tempo máximo de 14,52ms para comunicar, multiplicou-se todos os valores dos gráficos mostrados nos tópicos acima por 14,52 e calculou-se a sua média e o seu desvio padrão. Os resultados encontram-se nas tabelas abaixo e no gráfico da figura 4.20.

Tempo médio de comunicação - 2m		
Canal	Média(ms)	Desvio Padrão(ms)
11	31,90	19,68
17	68,69	68,02
26	33,49	19,89
11 Sem *Sleep*	14,61	1,11

Tabela 4.1 – Tempo médio de comunicação Monocanal 2m.

Tempo médio de comunicação - 5m		
Canal	Média(ms)	Desvio Padrão(ms)
11	70,84	218,16
17	552,98	645,20
26	36,90	26,57
11 Sem *Sleep*	14,66	1,43

Tabela 4.2 – Tempo médio de comunicação Monocanal 5m.

Tempo médio de comunicação - 10m		
Canal	Média(ms)	Desvio Padrão(ms)
11	89,90	258,08
17	1115,94	590,88
26	54,42	81,85
11 Sem *Sleep*	15,26	3,32

Tabela 4.3 – Tempo médio de comunicação Monocanal 10m.

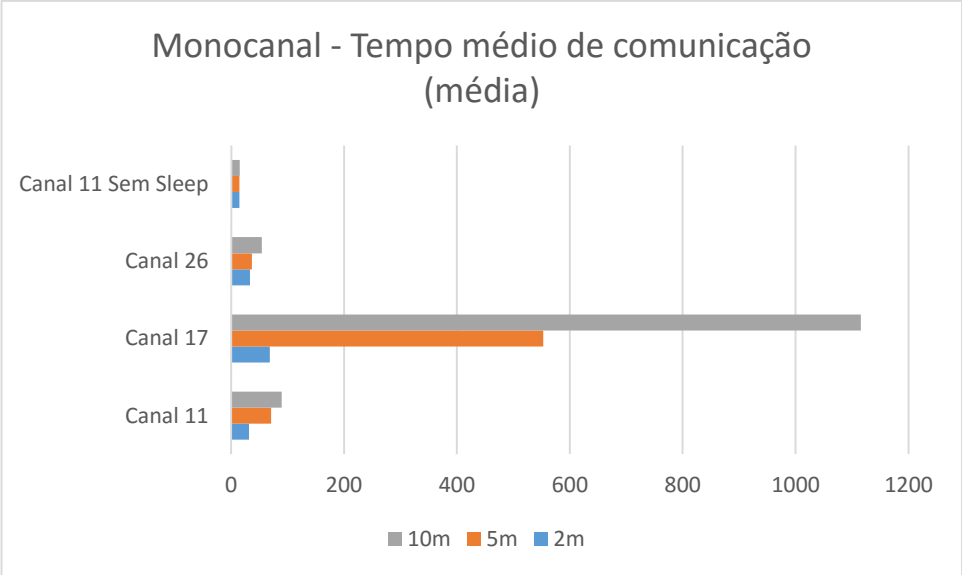


Figura 4.20 – Comparação do tempo médio de comunicação do Monocanal.

Com os valores do tempo médio de comunicação fica mais clara a comparação dos testes. Na tabela 4.1 nota-se que o desempenho do rádio utilizando o canal 11 e o 26 é praticamente igual, com uma pequena vantagem do 11. Já o canal 17 levou um pouco mais que o dobro do tempo para conseguir encontrar o sensor ligado. Para os testes de 5 metros, percebemos que o canal 26 conseguiu manter um bom desempenho, quase não mudou em

relação a 2 metros, enquanto o 11 já começou a sentir mais os efeitos da interferência e teve perdas significativas. O 17, como foi citado anteriormente, ficou 1/3 do tempo sem comunicação, refletindo, e muito, no seu tempo médio de comunicação, que chega a ter um desempenho 10 vezes pior que os outros canais. De 5 para 10 metros, os canais 11 e 26 tiveram um acréscimo de aproximadamente 20ms na sua comunicação, portanto, tiveram quase a mesma perda no desempenho. O canal 17 em relação ao teste de 5 metros, como foi dito no tópico 4.2.2, fica o dobro do tempo sem comunicação, chegando a ter um desempenho médio 15 vezes pior que os outros canais e 75 vezes pior que o teste sem *sleep*.

O rádio sem a técnica *Duty Cyling* teve a sua média praticamente igual ao tempo necessário para conseguir se comunicar na primeira tentativa. Quase invariável com a distância, mantendo um desempenho duas vezes melhor que o rádio operando no seu melhor canal (o 26) e chegando a ser quase 4 vezes melhor em 10 metros. A única variação mais brusca foi no desvio padrão em 10 metros, devido a maior instabilidade notada ao analisar o seu gráfico, e em um momento que ele necessitou de 3 tentativas.

4.3 Desempenho da técnica *Duty Cycling* Multicanal

O mecanismo dos testes da técnica Multicanal é bem parecido como o utilizado para o Monocanal, a grande diferença está no número máximo de tentativas que o computador faz por ciclo, pois em vez de 100 ele envia 85 dados diferentes. Esta mudança foi necessária para deixar o tempo de cada ciclo do computador igual ao utilizado na técnica Monocanal, pois, o código do Multicanal é mais extenso e cada tentativa de envio leva mais tempo tornado os ciclos mais demorados e mudando o seu ponto estável. Desta forma, coma a diminuição do número de tentativas, conseguimos ajustar cada intervalo para possui os mesmos 1,5 segundos e o seu ponto estável é na segunda tentativa.

A seguir, os testes com quantidades diferentes de canais serão avaliados em tópicos e ao final serão comparados. Afim de obter resultados mais limpos e testar os benefícios da técnica, em cada modo adotou-se os melhores canais, buscando um espaçamento igual entre eles dentro da faixa de frequência. Por exemplo, com 16 canais não há o que fazer, é necessário utilizar todos os disponíveis pelo rádio; já com 8 canais, exclui-se aqueles com muita interferência, como é o caso do 17, e entre os restantes escolheu-se os que possuíam maior

distância entre um e outro. Para os modos de 2 e 4 canais, seguiu-se a mesma lógica adotada com 8 canais.

4.3.1 Dezesseis canais

Ocupar os 16 canais garante que o rádio nunca ficará mais de uma transmissão no mesmo canal congestionado. Outra vantagem é no caso de uma rede, pois quanto mais canais forem empregados, menos dispositivos irão usufruir do mesmo canal. Porém, ao mesmo tempo que se utiliza todos os canais bons disponíveis, perde-se desempenho quando se conecta aos ruins. Desta forma, espera-se que o desempenho seja superior ao do Monocanal no pior caso, que neste trabalho é o canal 17, e inferior ao do melhor caso, o canal 26. Os gráficos abaixo ilustram a quantidade de transmissões necessárias para realizar comunicação.

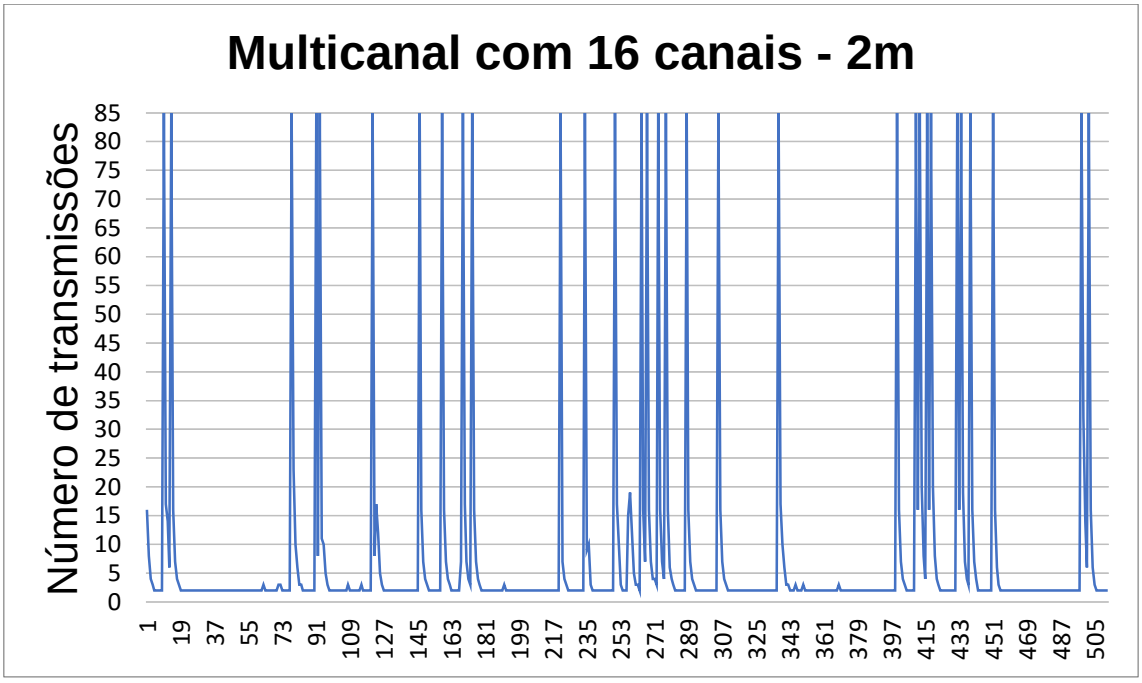


Figura 4.21 – Número de transmissões com 16 canais e 2m de distância entre os rádios.

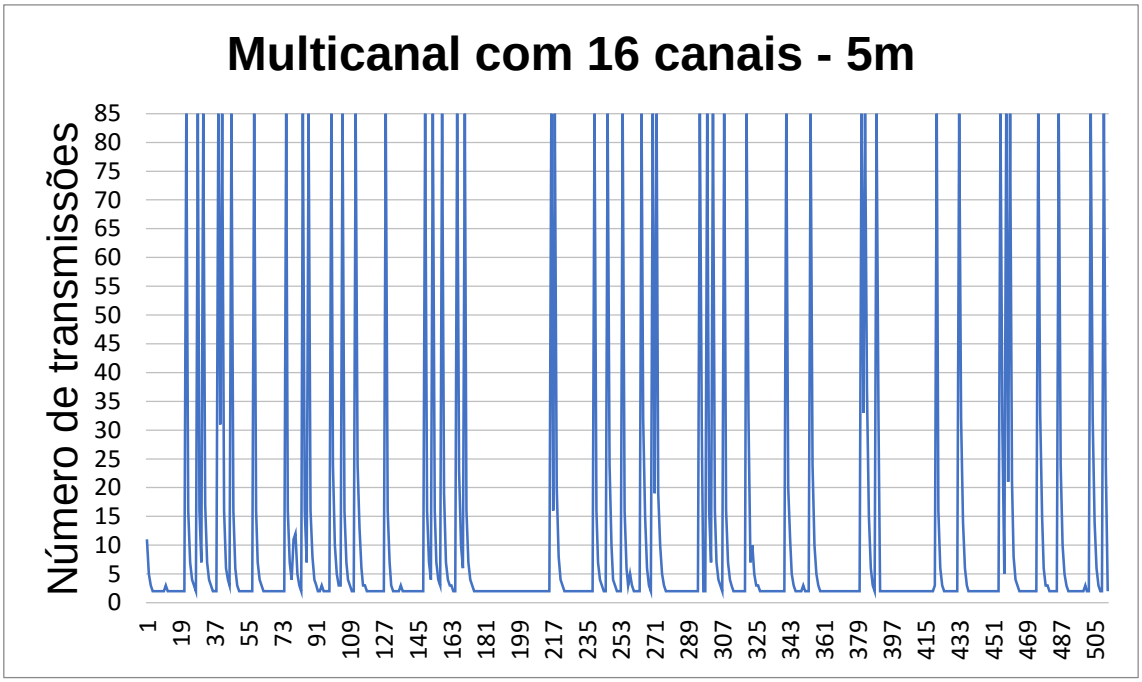


Figura 4.22 – Número de transmissões com 16 canais e 5m de distância entre os rádios.

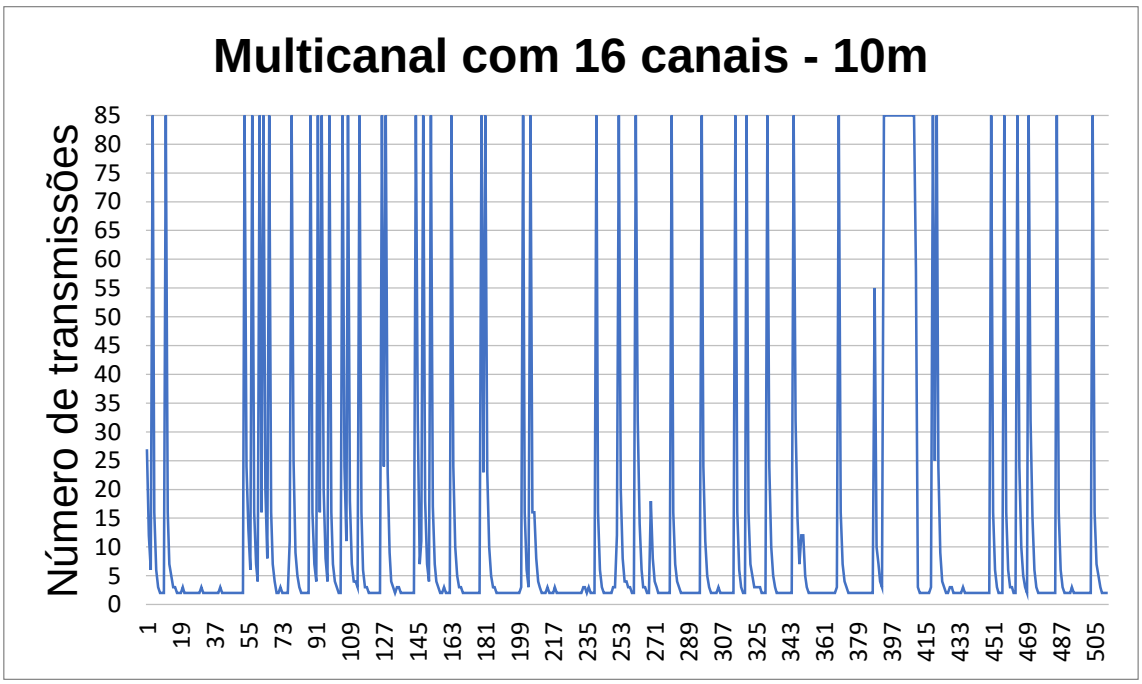


Figura 4.23 – Número de transmissões com 16 canais e 10m de distância entre os rádios.

Nota-se que em muitos ciclos o computador não conseguiu comunicação com o sensor, situação justificada por dois motivos em especial. O mais comum deles é o mesmo tipo de problema ocorrido no Monocanal, no qual a interferência impede os dados de chegarem ao

sensor quando acordado, fazendo com que o computador tente se comunicar 85 vezes sem sucesso. O segundo inconveniente é o frame do sensor não chegar ao computador, pois deste modo o sensor mudaria para o próximo canal e o computador ficaria no mesmo, provocando 100% de erro na comunicação no período seguinte. Este tipo de problema só é corrigido quando o computador percebe que ficou o tempo inteiro de um ciclo sem comunicação, então começa a varrer os canais até encontrar o correto. Geralmente a sincronia volta no próximo intervalo, a não ser que haja muito ruído naquele canal, pois como já foi dito, após 200 ciclos de *wake-up* do sensor ele sintonizará no canal de menor RSSI, buscando facilitar o encontro com o computador.

Nos testes coletados não houve grande problema para retomar o sincronismo, apenas em um instante, no gráfico dos 10 metros, analisa-se que o computador ficou alguns ciclos sem comunicação; foram precisamente 17 ciclos varrendo até voltar a encontrar o canal correto. Isto prova que a varredura funcionou corretamente, pois, por exemplo, se supormos que o computador havia parado no canal 11 e o sensor no 12, então ele deixaria de comunicar neste intervalo e no seguinte mudaria para o 12, porém a interferência impede-o de transmitir e então ele vai continuar varrendo, até voltar novamente ao 12, e por fim, conseguir comunicação. A figura 4.24 esclarece este problema e nela conta-se 17 mudanças de canal.

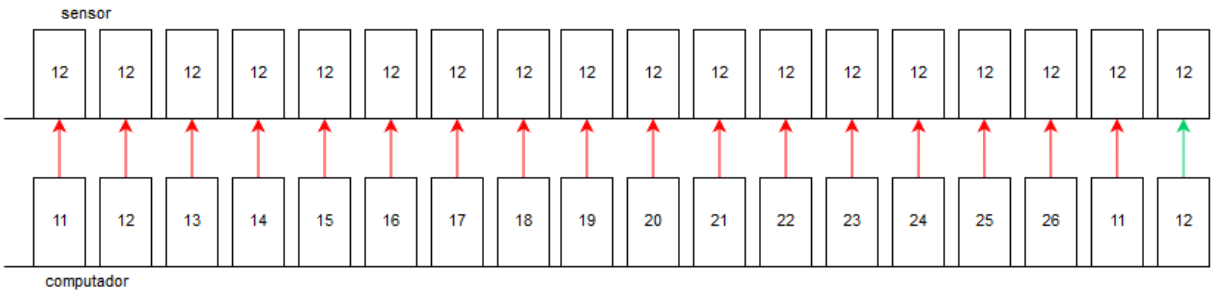


Figura 4.24 – Varredura dos canais pelo computador.

4.3.2 Oito canais

Como citado no início do tópico, para a realização dos testes de oito canais excluiu-se os canais com muita interferência para gerar resultados mais limpos e avaliar o melhor desempenho da técnica. Com o auxílio do software *Acrylic* foi realizada a análise das redes

locais e, da mesma forma que nos tópicos anteriores, diagnosticou-se maior interferência no canal 6, portanto, os canais desta faixa de frequência não foram empregados. A figura 4.25 possui o resultado da análise dos roteadores próximos no momento da realização dos testes.

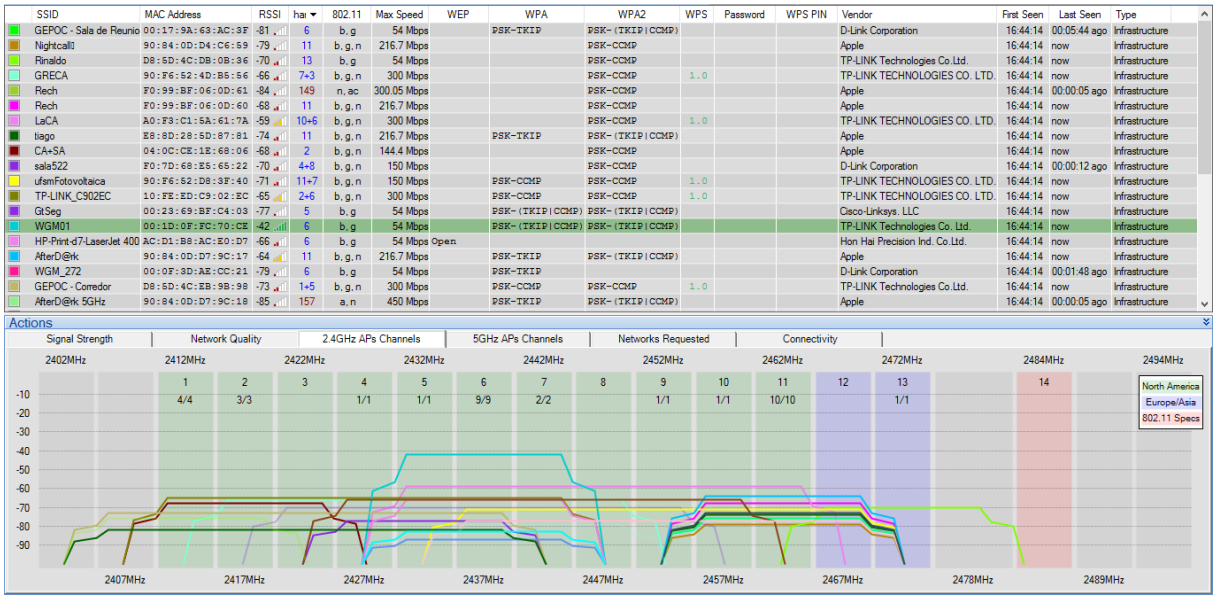


Figura 4.25 – Roteadores Wi-Fi que interferiram nos testes com 8 canais.

Os gráficos abaixo mostram os resultados do número de transmissões necessárias para realizar a comunicação. Apesar de continuar ocorrendo muitas falhas de envio, em nenhum momento o computador foi forçado a realizar uma varredura completa, e mesmo que ela fosse necessária, a redução na quantidade de canais torna-a mais rápida. Apenas no gráfico dos 10 metros analisa-se que, por um curto período de tempo, entre os ciclos 379 e 397 o rádio ficou mais tempo sem comunicação por precisamente 5 envios do computador. Isto não caracteriza uma varredura completa, já que estamos trabalhando com 8 canais, então deduzimos que o computador conseguia enviar pacotes para o sensor, fazendo-o mudar de canal, porém não recebia o byte do próximo canal de volta. Desta forma, ele também mudava o canal, pois ficava 85 tentativas sem receber nada e iniciava uma nova varredura. Isto foi realizado até que os dois se estabeleceram em um canal bom e retomaram a transmissão.

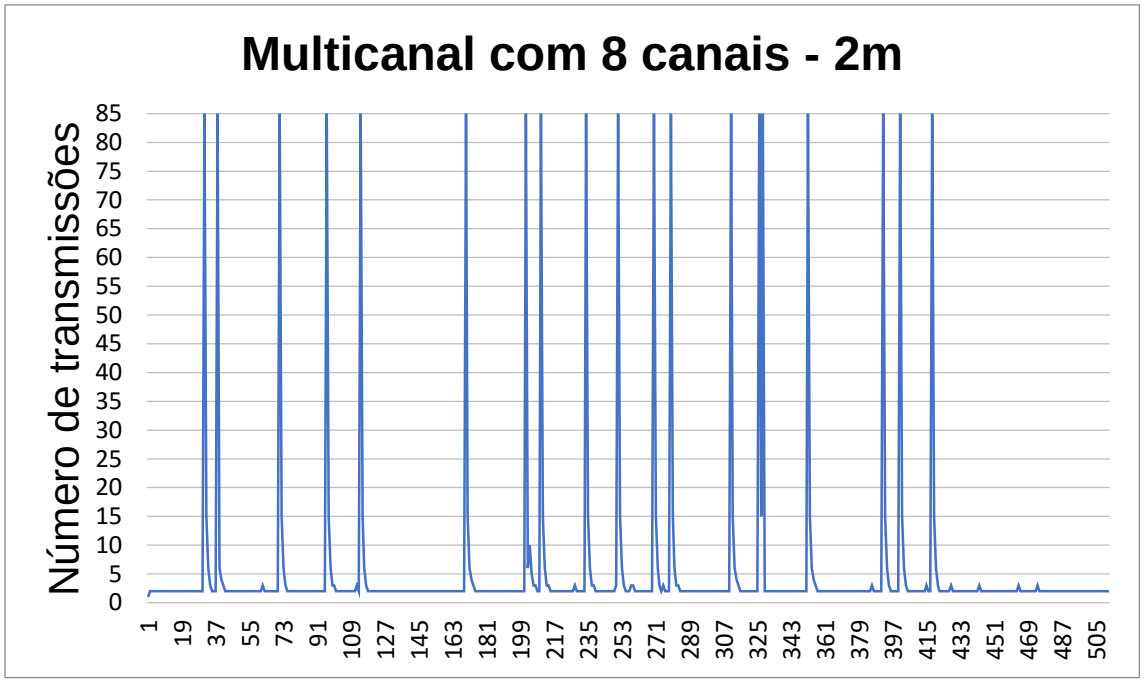


Figura 4.26 – Número de transmissões com 8 canais e 2m de distância entre os rádios.

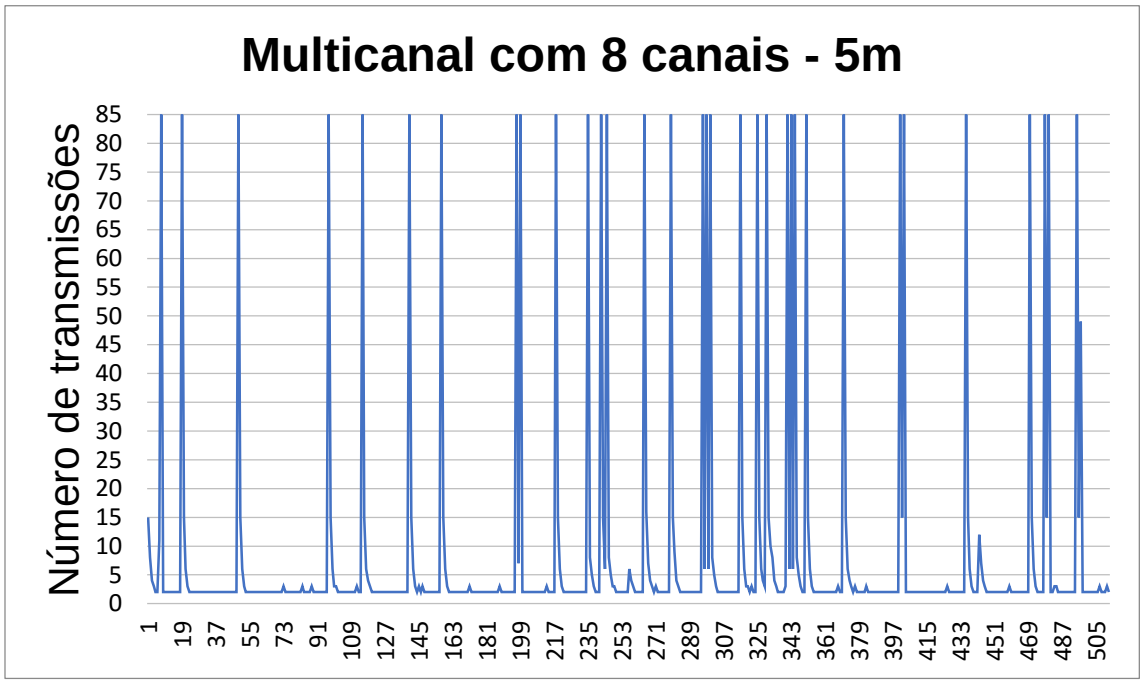


Figura 4.27 – Número de transmissões com 8 canais e 5m de distância entre os rádios.

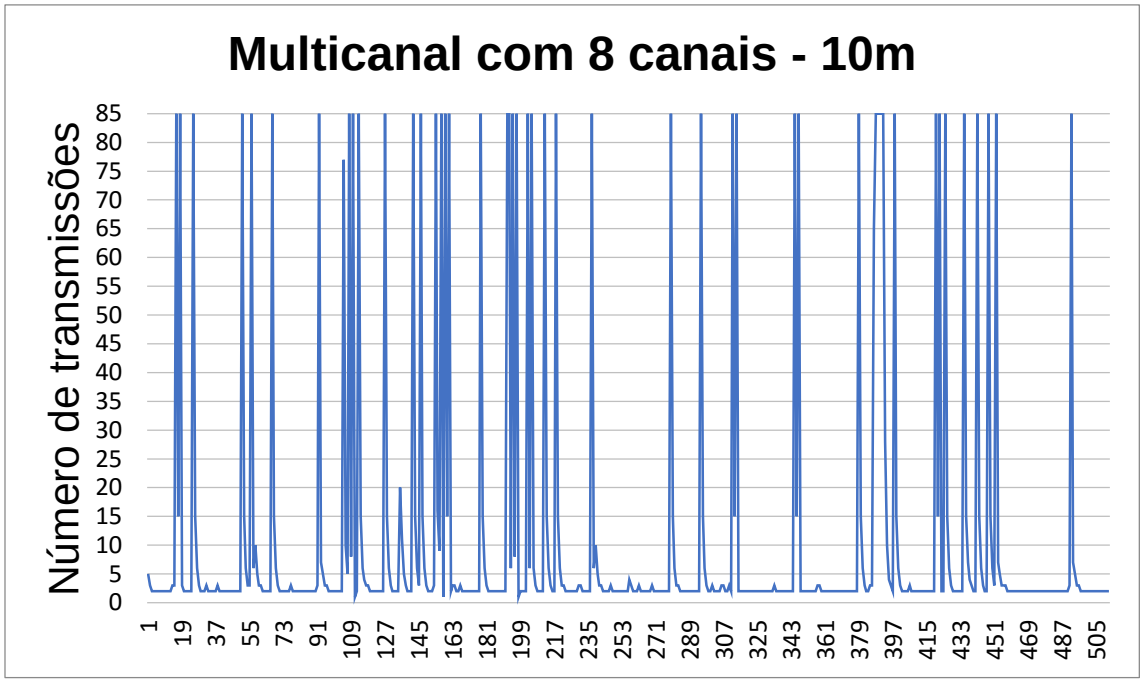


Figura 4.28 – Número de transmissões com 8 canais e 10m de distância entre os rádios.

4.3.3 Quatro canais

Da mesma forma feita na implementação dos testes de oito canais, excluiu-se os piores e utilizou-se os quatro melhores dentro dos selecionados anteriormente. Novamente foi feita a análise da rede com o software *Acrylic*, cuja figura 4.29 ilustra a ocupação dos canais.

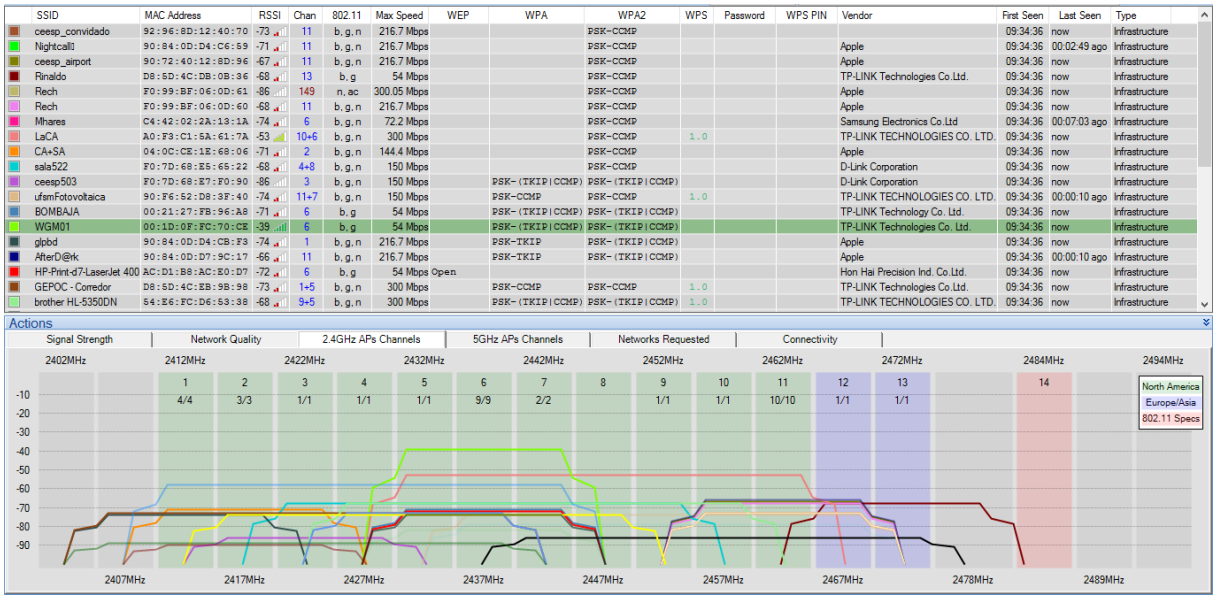


Figura 4.29 – Roteadores Wi-Fi que interferiram nos testes com 4 canais.

Analisando os gráficos abaixo, nota-se que com o aumento da distância não houve uma queda significativa no desempenho do rádio utilizando 4 canais. Apenas, novamente no caso mais distante, em alguns instantes foi necessária a varredura do computador. Visualizando na figura 4.32 o intervalo entre o ciclo 109 e 163, percebe-se a ocorrência de 4 varreduras com as mesmas características da realizada no rádio de 16 canais, no qual foi necessário passar por todos os canais até voltar ao correto. Comparando aquele caso com este, fica evidente que a recuperação é muito mais rápida quando se está trabalhando com uma quantidade menor de canais.

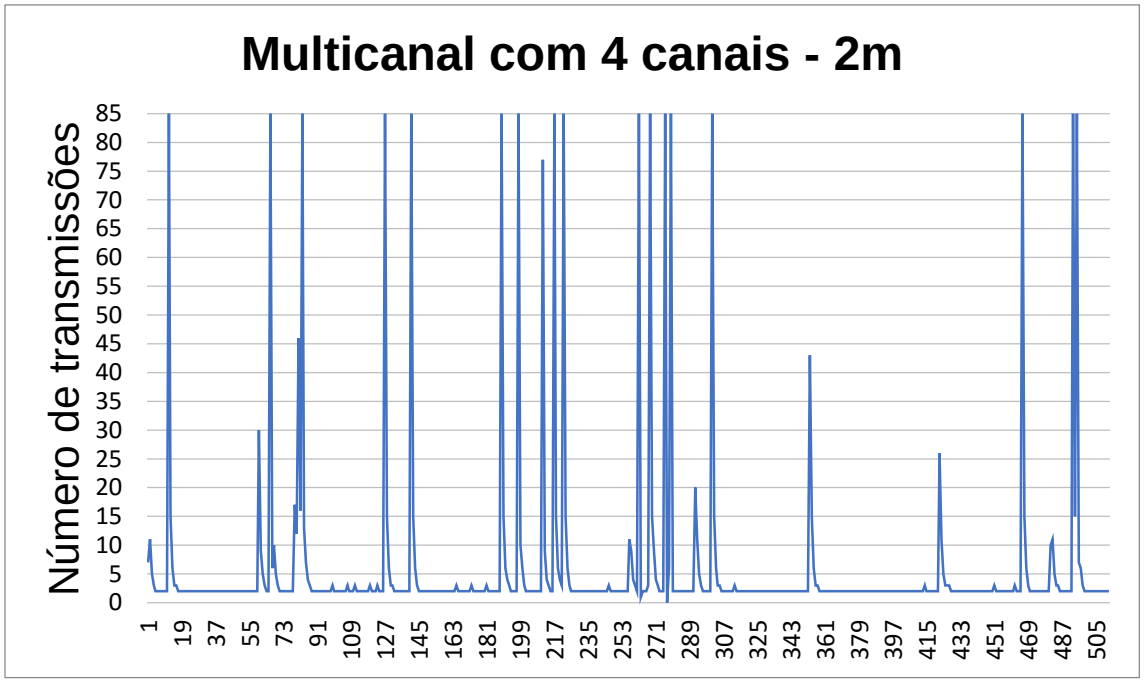


Figura 4.30 – Número de transmissões com 4 canais e 2m de distância entre os rádios.

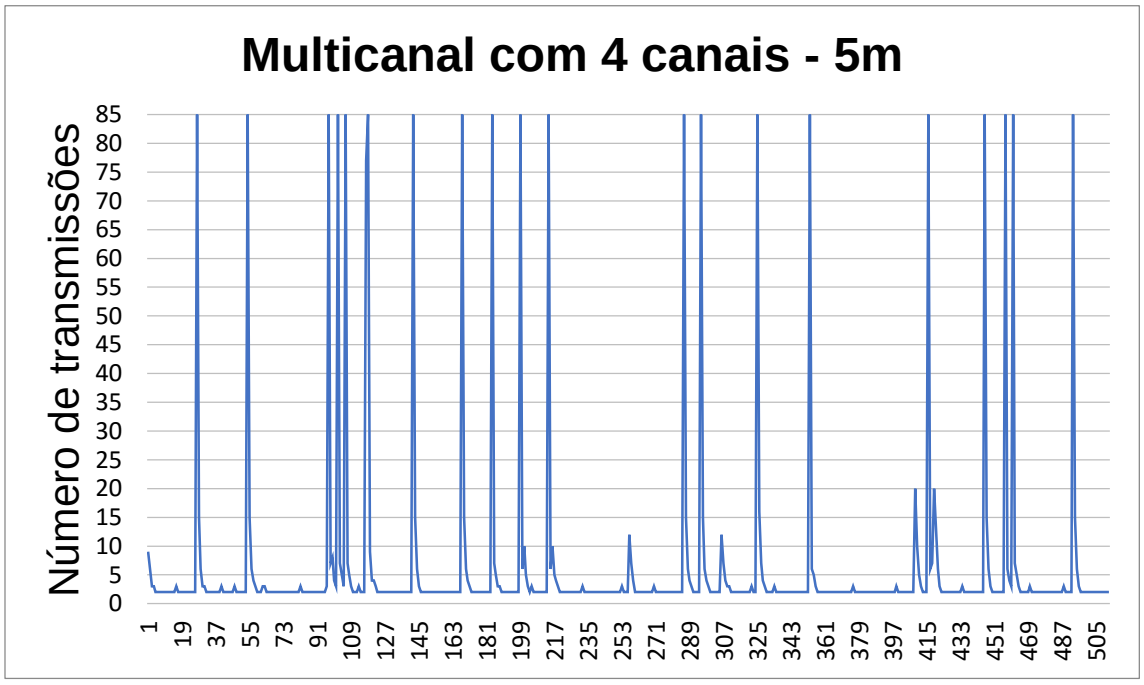


Figura 4.31 – Número de transmissões com 4 canais e 5m de distância entre os rádios.

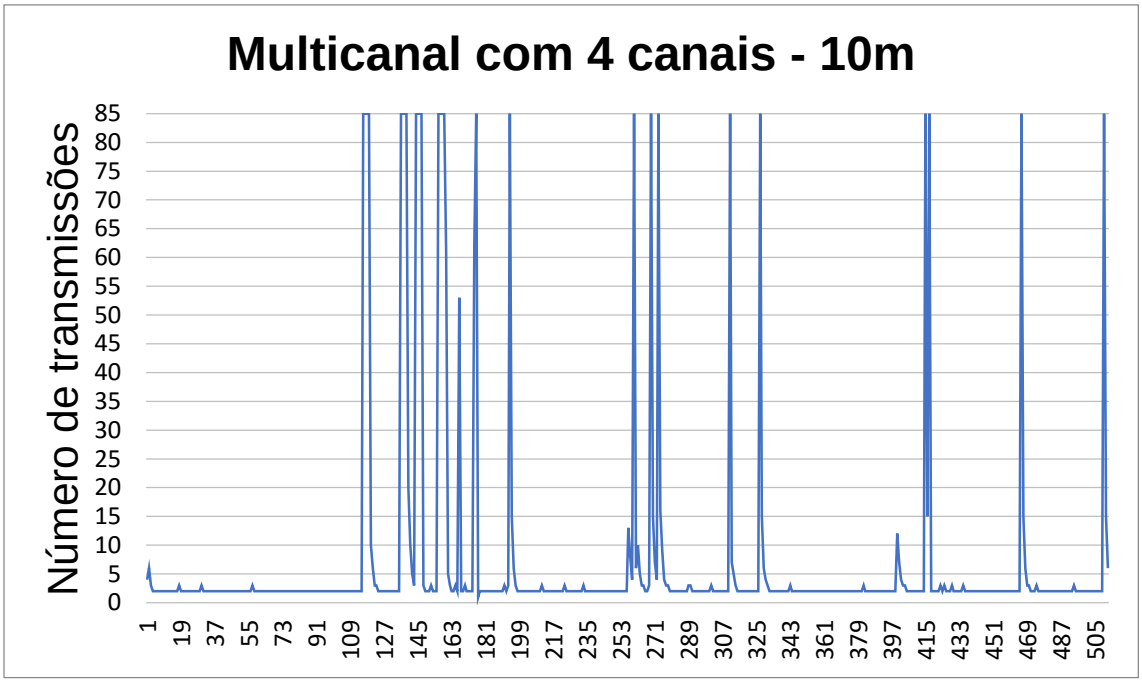


Figura 4.32 – Número de transmissões com 4 canais e 10m de distância entre os rádios.

4.3.4 Dois canais

Para coletar dados mais justos e coerentes em relação aos outros testes, foram usados dois canais que estiveram presentes em todos anteriores, o 11 e o 26. Seguindo a lógica, conforme baixa-se a quantidade de canais utilizados, menor será a probabilidade de ocupar um canal com muita interferência. Desta forma, os testes com dois canais devem ter desempenho próximo ao do Monocanal com pouca perturbação, como pode-se ver nos gráficos abaixo e comparando-os com os resultados do Monocanal já apresentados.

Temos com 2 metros alguns instantes sem comunicação, provavelmente causados por perda do byte de canal, e que não aconteceu nos testes com apenas o canal 11 ou 26, isoladamente. Fora estes momentos de falta de sincronia, o rádio conseguiu manter-se no ponto estável, de duas tentativas, pela maior parte do tempo. Nos testes de 5 e 10 metros fica evidente que o rádio conseguiu manter o seu desempenho, se comparado ao do rádio com o canal 11 e com o canal 26, provando a previsão de possuir um comportamento próximo ao do rádio com um canal.

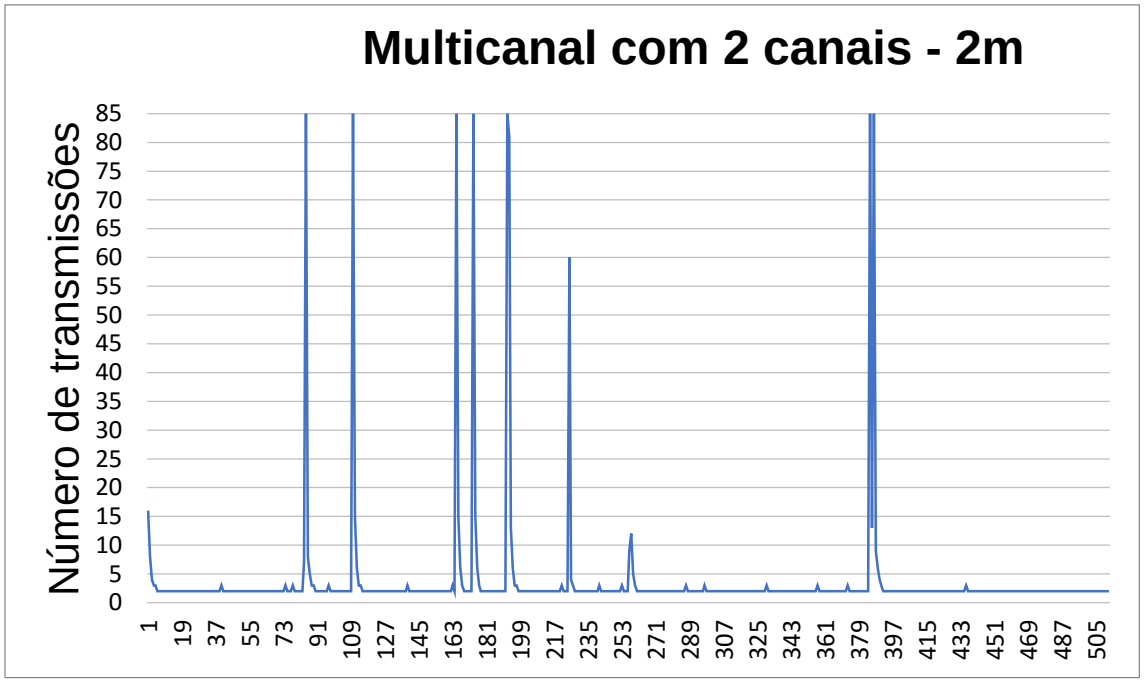


Figura 4.33 – Número de transmissões com 2 canais e 2m de distância entre os rádios.

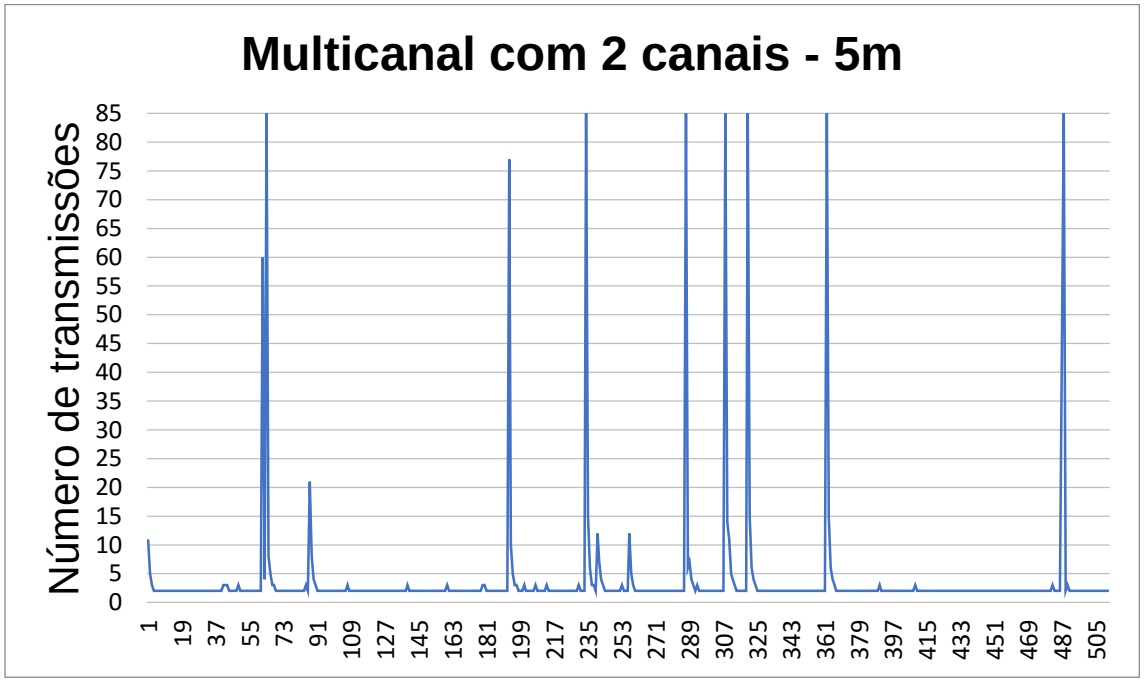


Figura 4.34 – Número de transmissões com 2 canais e 5m de distância entre os rádios.

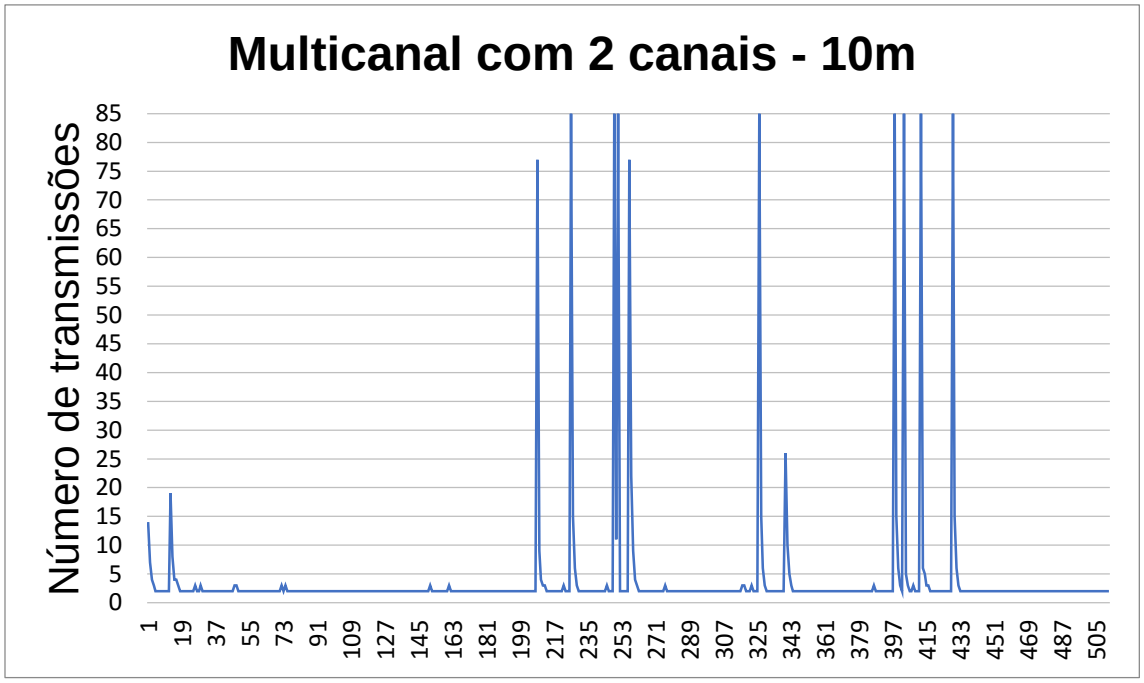


Figura 4.35 – Número de transmissões com 2 canais e 10m de distância entre os rádios.

4.3.5 Comparações do tempo médio de comunicação

Os cálculos do tempo médio de comunicação desta seção são feitos da mesma forma que os realizados no tópico 4.2.5. Multiplicando todos os valores dos gráficos mostrados nos testes acima por 14,52, calculou-se a sua média e o seu desvio padrão. Os resultados encontram-se nas tabelas abaixo e no gráfico da figura 4.36.

Tempo médio de comunicação - 2m		
Multicanal	Média(ms)	Desvio Padrão(ms)
16 canais	123,96	287,84
8 canais	83,096	228,81
4 canais	87,83	225,82
2 canais	54,39	154,22

Tabela 4.4 – Tempo médio de comunicação Multicanal 2m.

Tempo médio de comunicação - 5m		
Multicanal	Média(ms)	Desvio Padrão(ms)
16 canais	175,01	344,48
8 canais	122,57	296,37
4 canais	90,38	238,39
2 canais	56,61	155,73

Tabela 4.5 – Tempo médio de comunicação Multicanal 5m.

Tempo médio de comunicação - 10m		
Multicanal	Média(ms)	Desvio Padrão(ms)
16 canais	210,51	384,67
8 canais	163,49	353,29
4 canais	105,18	275,47
2 canais	58,79	165,96

Tabela 4.6 – Tempo médio de comunicação Multicanal 10m.

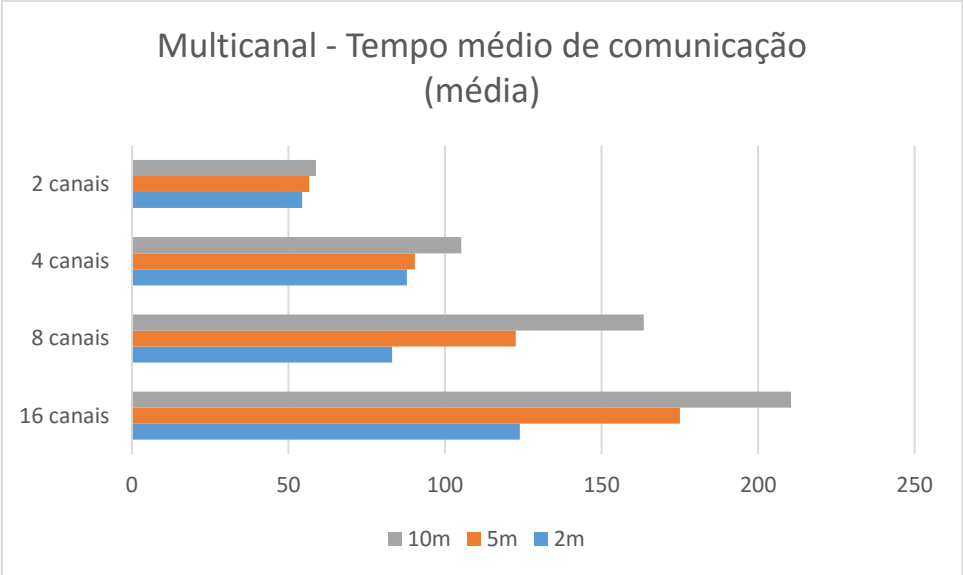


Figura 4.36 – Comparação do tempo médio de comunicação do Multicanal.

Analisando as tabelas acima, nota-se que em quase em todos os casos o tempo médio de comunicação diminuiu conforme empregou-se menos canais, salvo os testes de 2 metros, nos quais de 8 para 4 houve aumento da média, porém, o desvio padrão decresceu. Com isso conclui-se que obteve-se mais valores máximos no teste de 8 canais, porém menor instabilidade, ou seja, com 8 canais o computador ficou mais ciclos sem conseguir comunicação e no ponto

estável do que o computador com 4 canais. Além disso, ganhou-se em média 40ms utilizando 8 ou 4 canais no lugar de 16, e mais 35ms se operado com apenas 2 canais.

Com 5 metros de distância as diferenças começam a ficar mais claras, principalmente se compararmos a ocupação dos 16 canais com os outros modos. Só de cortar o número de canais pela metade já se ganha 50ms de tempo médio de comunicação. Conforme diminui-se a quantidade de canais, esta variação encurta, por exemplo, de 4 para 2 canais houve ganho de apenas 34ms. Já com 10 metros de distância, o desempenho melhorou de forma um pouco mais gradual, ganhando cerca de 40 a 50ms cada vez que se diminuiu a quantidade de canais. Apenas de 8 para 4 canais que ficou um pouco fora deste valor.

Outro fator importante que pode-se notar é a grande queda de desempenho do rádio Multicanal conforme utiliza mais canais em distâncias maiores. Por exemplo, se compararmos as tabelas, repara-se que há um aumento entre 40ms e 60ms no tempo médio de comunicação empregando 16 e 8 canais, enquanto que para 4 canais perde-se em torno de 10ms e para 2 canais apenas 2ms.

4.4 Consumo de energia

Relembramos que a corrente no modo de recepção é de 19mA, no modo de transmissão é de 23mA e no modo *sleep* é de 2µA. Sabe-se que o sensor, quando consegue comunicação, fica cerca de 44,73ms acordado, 33,89ms em modo de recepção e 10,84ms em modo de transmissão. Com *sleep* de 306ms o sensor permanece 87,25% do tempo dormindo, 9,66% recebendo e 3,09% transmitindo. Conhecendo as porcentagens é possível determinar quanto gasta de bateria em uma hora, através do seguinte cálculo:

$$\begin{aligned} &Consumo\ de\ bateria\ em\ uma\ hora \\ &\quad = 19mA \cdot 0{,}0966h + 23mA \cdot 0{,}0309h + 2\mu A \cdot 0{,}8725h \\ &\quad = 1{,}8354mAh + 0{,}7107mAh + 0{,}001745mAh \\ &Consumo\ de\ bateria\ em\ uma\ hora = \mathbf{2{,}547845mAh} \end{aligned}$$

Também podemos calcular o gasto em uma hora, caso o sensor não conseguisse comunicação, ligando por apenas 14,56ms. Desta forma, 95,46% do tempo fica dormindo e 4,54% do tempo em modo de recepção. Com as porcentagens calculamos:

$$Consumo\ de\ bateria\ em\ uma\ hora = 19mA \cdot 0{,}0454h + 2\mu A \cdot 0{,}9546h$$
$$= 0{,}8626mAh + 0{,}0019092mAh$$
$$Consumo\ de\ bateria\ em\ uma\ hora = \mathbf{0{,}8645092mAh}$$

Por último, é feito o cálculo do modo sem economia de energia, e neste caso, utilizamos as mesmas quantidades de tempo do primeiro cálculo, considerando que o rádio fique 33,89ms recebendo e depois 10,84ms transmitindo, ou seja, 75,77% em modo de recepção e 24,23% em modo de transmissão:

$$Consumo\ de\ bateria\ em\ uma\ hora = 19mA \cdot 0{,}7577 + 23mA \cdot 0{,}2423h$$
$$= 14{,}3963mAh + 5{,}5729mAh$$
$$Consumo\ de\ bateria\ em\ uma\ hora = \mathbf{19{,}9692mAh}$$

Sabendo, em cada caso, o quanto se consome de bateria em uma hora, é possível fazer o cálculo de quantos dias levaria para esgotar a bateria de um celular, que em média tem capacidade de 2000mAh, dividindo-se o valor da bateria pelos valores calculados e inserindo-os na tabela 4.7.

Casos	**Consumo em uma hora**	**Dias para esgotar bateria de celular**
Comunicando em todo ciclo de *wake-up*	2,548mAh	32,7 Dias
Nunca comunicando	0,865mAh	96,4 Dias
Sem *sleep*	19,969mAh	4,2 Dias

Tabela 4.7 – Consumo de bateria do rádio.

Com os dados da tabela acima, pressupõe-se que o rádio em trabalho normal, algumas vezes comunicando e outras não, esgotaria a bateria entre 32,7 e 96,4 dias. Desta forma, o uso da técnica economiza entre 88,62% e 95,86% o consumo em relação ao caso sem *sleep*, porcentagens bem próximas das previsões citadas no tópico 3.5.1 Tempo de *Sleep*. Por último, é importante ressaltar que o gasto do microcontrolador não está sendo considerado, tornando a sua escolha mais genérica, pois ao possibilitar a utilização de qualquer modelo de

microcontrolador, será necessário apenas acrescentar o seu consumo para saber quanto o aparelho inteiro irá demorar para esgotar a bateria.

4.5 Comparações do Monocanal e Multicanal

Fazendo uma comparação geral dos resultados, observa-se que para pequenas distâncias o uso de vários canais é praticamente inviável, pois para 16 canais obteve-se um desempenho duas vezes inferior ao pior caso do Monocanal. Com 8 e 4 canais o seu desempenho também foi pior, e com 2 canais houve um pequeno ganho, mas quase insignificante.

Com distâncias maiores as vantagens do Multicanal começam a aparecer. Comparando os dois piores casos de cada modelo, de 2 para 5 metros os testes com 16 canais pioraram o seu desempenho em 41% e o rádio operando no canal 17 piorou 700%, assim, o Multicanal começa a fazer mais sentido em questão de robustez, pois conseguiu um desempenho 216% melhor que o Monocanal operando com muita interferência. Desta forma, prova-se que é mais funcional tentar escapar das distorções e grandes tráfegos em determinadas faixas de frequência com o emprego de diferentes canais, em cada nova comunicação. Porém, no caso dos 5 metros, se compararmos o desempenho dos 16 canais com o uso de um canal que sofre de pouca interferência, como é o caso do canal 26, percebemos uma perda de 370%, e mesmo se compararmos com o uso de 4 canais, continua-se a ter perdas de 140%. Portanto, ao mesmo tempo em que o Multicanal ganha em relação a utilização de um canal congestionado do mono, perde-se mais quando opera-se em canais bons.

Por último, comparando os testes realizados com 10 metros de distância entre os rádios, as perdas do Multicanal 16 reduziram de 41% para 21% em relação aos testes de 5 metros, e para o Monocanal 17 reduziram de 700% para 100%. Porém, em relação a esses dois casos, os ganhos com a utilização de vários canais passaram de 216% para 429%, e as perdas na utilização de 16 canais com o uso de um canal com pouca interferência (canal 26) caíram de 370% para 289%. Desta forma, o ganho de desempenho comparando a utilização de um canal congestionado é muito mais compensador que a perda de desempenho do Monocanal operando com pouca interferência. Assim, finalmente prova-se a robustez do Multicanal, o qual sofre menos com a distância e mantém um desempenho aceitável.

5. CONCLUSÃO

Neste trabalho coletou-se uma série de dados para analisar o desempenho das técnicas desenvolvidas, diagnosticando vários fatores básicos que influenciaram na sua performance. O primeiro deles é a interferência de outros sinais, pois notou-se que quanto maior a sua potência, pior será o desempenho do rádio. Este problema está diretamente relacionado a arquitetura *low-IF*, que é uma das estruturas mais vulneráveis quando se trata da ocupação do canal. Outro fator bastante influente foi a distância dos rádios, pois quanto mais longe um fica do outro, menor será a força do seu sinal quando chega ao destino, fazendo com que o rádio sofra mais com os ruídos no canal.

Para evitar estes sinais de alta potência desenvolveu-se um modo que troca o canal do rádio a cada nova comunicação, buscando escapar de canais muito perturbados. Este modo perdeu em desempenho com relação ao rádio operando em apenas um canal de pouca interferência, porém, este tipo de perda era previsto, pois ao utilizar vários canais, é inevitável que alguns deles tenham maior intromissão que outros. Porém, a medida que se distanciou um rádio do outro, o Multicanal se mostrou mais robusto, tendo ganhos compensadores quando comparado ao Monocanal exposto a grandes interferências.

A comparação do desempenho destes modos com um rádio que opere o tempo inteiro sem entrar em *sleep* e, portanto, sem perder dados por estar dormindo, não foi levada a fundo. Ainda assim, um teste foi o suficiente para verificar que com a utilização das técnicas perde-se de duas a quatro vezes o desempenho. No entanto, o grande trunfo do trabalho está na econômica de energia, que torna o rádio entre 88,62% e 95,86% mais econômico.

Considerando que esta comunicação seria empregada em um dispositivo para monitoramento de sinais vitais de pacientes, o pedido destes dados pelo computador central aconteceria em média a cada 2 segundos ou até mais. Com os dados do tempo médio de comunicação do Multicanal utilizando 16 canais, temos uma latência de 210,5ms no envio dos dados, atraso praticamente insignificante dentro destes 2 segundos. Desta forma, pode-se concluir que é viável a aplicação do Multicanal por ser mais robusto à distância e não insistir nos mesmos canais.

Concluímos então que a utilização da técnica deu um excelente ganho em questão de economia de energia. Comparando o Monocanal e o Multicanal, pode-se observar que a utilização de apenas um canal que sofre com pouca interferência é vantajoso na questão de

tempo de entrega dos pacotes, porém, este canal pode sofrer mais em outro local, tornando inviável a utilização do mesmo. Isto não acontece no Multicanal, pois não ficará o tempo inteiro insistindo no mesmo canal e ainda mantém uma média no seu tempo médio de comunicação, independente das condições de operação. Desta forma, pode-se dizer que ao utilizar mais canais ganha-se em adaptação, robustez e confiabilidade na entrega.

5.1 Trabalhos futuros

Como trabalhos futuros podemos pensar na mescla entre as duas técnicas, caso seja necessária uma comunicação tão boa quanto o Monocanal no melhor caso e que não fique presa sempre ao mesmo canal. Assim, poderia ser feita uma técnica que periodicamente faça a varredura da rede e verifique o canal de menor RSSI, passando a utilizar apenas este canal até a nova varredura. Ou então, selecionando os melhores canais e fazer um Multicanal com eles.

Outro trabalho interessante seria a realização dos testes em uma rede para verificar se o desempenho do mono ainda é melhor. E talvez neste caso, implementar a escolha randômica de canais para evitar que muitos rádios utilizem o mesmo canal. Também seria interessante a utilização de outro microcontrolador para tentar fazer uso do *Contiki-OS* e compará-lo com o rádio deste trabalho.

Por último, sugere-se a implementação de alguma técnica que guarde os tempos de comunicação com cada rádio, como é feito no *ContikiMAC*, para poupar o tempo que o computador fica transmitindo antes de conseguir comunicação e sincronizar melhor os rádios. Para isso, talvez seja interessante a troca do modo *nonbeacon* para o *beacon*, modo que faz uso de um frame para a sincronização dos rádios.

REFERÊNCIAS

DUNKELS, A. **The ContikiMAC Radio Duty Cycling Protocol**. [Suécia]: SICS Technical Report, 2011.

AL NAHAS, B.; DUQUENNOY, S.; IYER, V.; VOIGT, T. **Low-Power Listening Goes Multi-Channel.** Suécia: SICS Swedish ICT AB; Suécia: Uppsala University.

AL NAHAS, B. **Multichannel Communicationin Contiki's Low-power IPv6 Stack**. Suécia: Uppsala University, 2013.

SOARES, V. F. **Projeto de Módulos de RF para Sistema em Chip CMOS**. Brasília: Universidade de Brasília, 2008.

MICROCHIP. **PIC18F2420/2520/4420/4520 Data Sheet**. Estados Unidos da América, 2004.

MICROCHIP. **MRF24J40 Data Sheet.** Estados Unidos da América, 2008.

CONTIKI-OS. **Contiki Hardware**. Disponível em: < http://www.contiki-os.org/hardware.html >. Acessado no dia 02 de dezembro de 2015.

MICROCHIP. **MWI Redes sem fio com os novos protocolos MiWi e ZigBee**. 2010. Disponível em: < http://pt.slideshare.net/andrerasminio/zigbee-andre >. Acessado no dia 02 de dezembro de 2015.

TELECO. **LAN / MAN Wireless I: Padrões 802.11 a, b e g.** Disponível em: < http://www.teleco.com.br/tutoriais/tutorialrwlanman1/pagina_3.asp >. Acessado no dia 02 de dezembro de 2015.

AZEVEDO, A. M. **Resumo do artigo "antena ativa ômega 3"**. Minas Gerais, 2002. Disponível em: < http://www.sarmento.eng.br/Tecnica10.htm >. Acessado no dia 02 de dezembro de 2015.

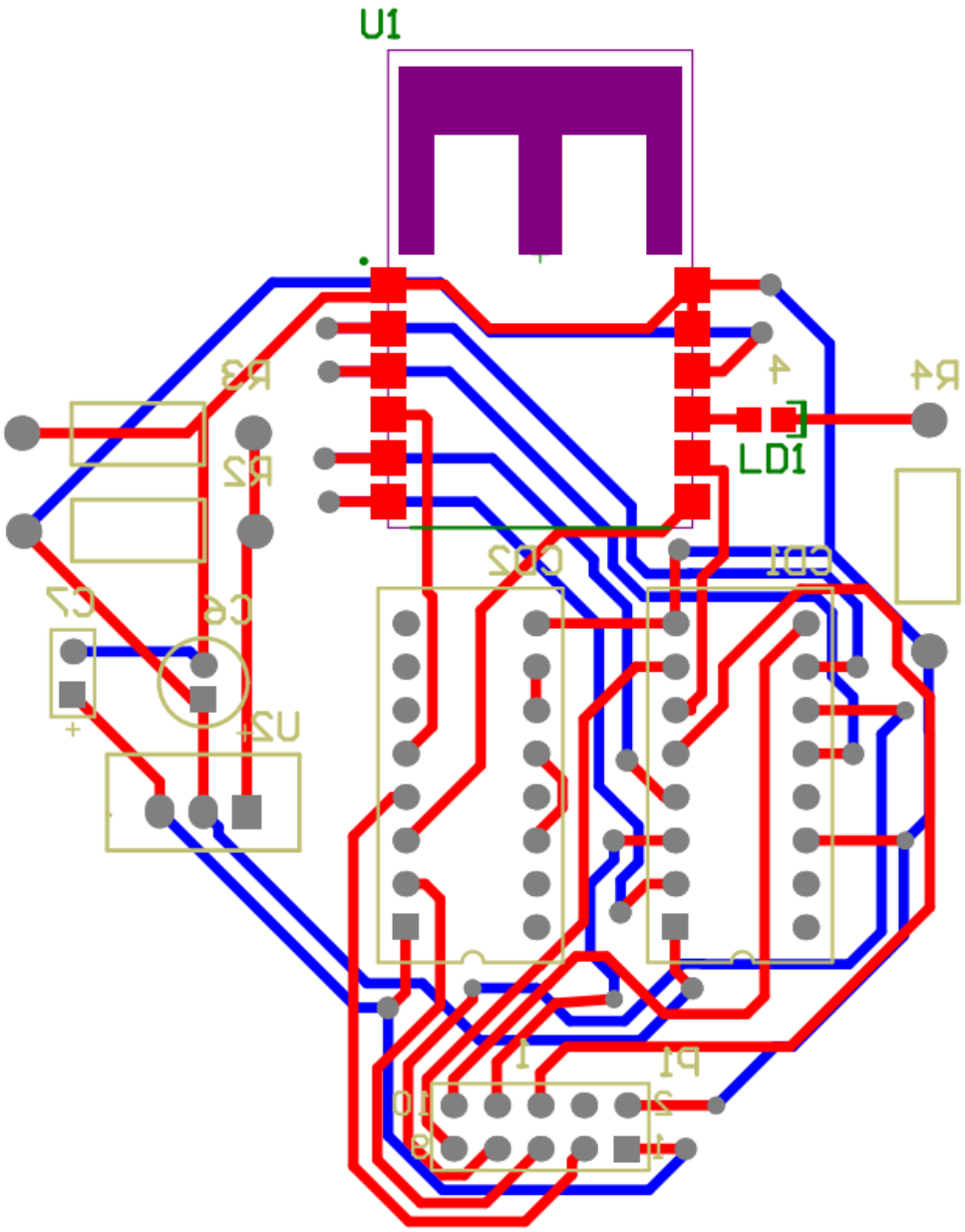
IEEE. **802.15.4** – IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan área networks – Specific requirements. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY)

Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). Nova York, 2003.

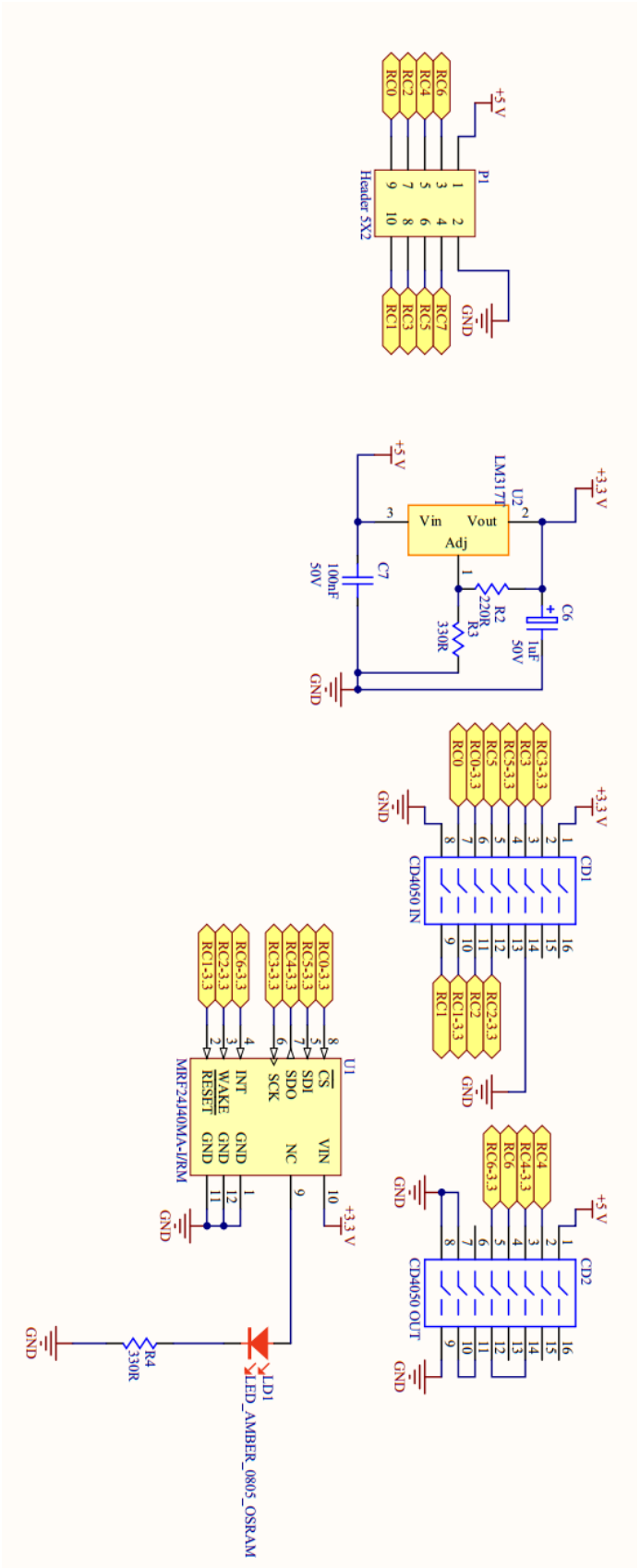
ROCHA, J. S. **Avaliação e Aplicações de Sistemas de Redes de Sensores Sem Fio.** 2011. Disponível em: < http://petroleo.dca.ufrn.br/redic/IV_Workshop_REDIC/Apresentacoes/IV%20Workshop%20REDIC_Instrumentacao.pdf >. Acessado no dia 03 de dezembro de 2015.

ANEXOS

ANEXO A – Esquemático da placa reguladora de tensão.



ANEXO B – Esquemático dos componentes da placa reguladora de tensão.



ANEXO C – Lista de materiais da placa reguladora de tensão.

Bill of Materials

<Parameter Title not found>

Source Data From: radio.PrjPCB
Project: radio.PrjPCB
Variant: None

Creation Date: 23/09/2015 14:51:32
Print Date: 42270 42270,61923

Footprint	Comment	LibRef	Designator	Description	Quantity
CAP_A	100uF/10V	CAP_0.1uF_50V_ELECTROLYTIC_RADIAL	C6	0.1uF 50V ELECTROLYTIC RADIAL	1
BAT-2	100nF_50V_10%	CAP_100nF_Disc_50V_CERAMICO	C7	Capacitor Cerâmico não polarizado Disc - 50V - Tolerância 10%	1
DIP-16-KEY	CD4050 IN	SW-DIP8	CD1	4009 Series DIP Switch, Raised actuator	1
DIP-16-KEY	CD4050 OUT	SW-DIP8	CD2	4009 Series DIP Switch, Raised actuator	1
LED_A_0805	LED_AMBER_0805_OSRAM	LED_AMBER_0805_OSRAM	LD1	Amber Hyper Mini TOPLED Hyper Bright LED OSRAM	1
HDR2X5	Header 5X2	Header 5X2	P1	Header, 5-Pin, Dual row	1
RES	220R	RES_1K5_DIP	R2	1K5 OHM 5% DIP Resistor - 1/4W	1
RES	330R	RES_1K5_DIP	R3, R4	1K5 OHM 5% DIP Resistor - 1/4W	2
MRF24J40MA	MRF24J40MA-I/RM	MRF24J40MA-I/RM	U1	2.4 GHz IEEE Std. 802.15.4, RF Transceiver 12-Pin Module, Industrial Temperature	1
TO-220	LM317T	LM317T	U2	DIP 3-Terminal Positive Voltage Regulators	1
					11

Approved **Notes**