Fernando Abrantes Vita

Estudo e Desenvolvimento de Software Para Medição de Desempenho de Redes Sub-1GHz

Campina Grande, PB

2018

Fernando Abrantes Vita

Estudo e Desenvolvimento de Software Para Medição de Desempenho de Redes Sub-1GHz

Trabalho monográfico acadêmico apresentado como nota final da disciplina Projeto de Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática

Departamento de Engenharia Elétrica

Orientador: Dr. Rafael Bezerra Correia Lima

Campina Grande, PB

2018

Fernando Abrantes Vita

Estudo e Desenvolvimento de Software Para Medição de Desempenho de Redes Sub-1GHz

Trabalho monográfico acadêmico apresentado como nota final da disciplina Projeto de Engenharia Elétrica.

Aprovado em 22 de março de 2018

Dr. Rafael Bezerra Correia Lima
Orientador

Dr. George Acioli Júnior
Convidado

Campina Grande, PB
2018

Dedico este trabalho a minha família, sem eles eu não estaria aqui.

Agradecimentos

Agradeço primeiramente a minha mãe, ela é a principal responsável por esta conquista.

Agradeço ao professor Rafael Lima por ter me orientado neste trabalho.

Agradeço aos irmãos Lucas e Iandê, apesar de não mais dividirmos a mesma casa sei que vocês sempre estão perto de mim.

Agradeço a Thayanne Lima pela paciência, pelos incentivos e pelas revisões de última hora.

Agradeço a Charles Dias pelo profissionalismo e solicitude.

Agradeço também aos professores Alexandre Cunha, Elmar Melcher e Joseana Fechine que em diferentes momentos me orientaram dentro da academia.

Agradeço a todos que caminharam junto comigo ao longo deste curso.

"O que separa o homem dos bichos
é que o homem sabe que é irracional."
(Ed Mort, personagem de Luis Fernando Verissimo)

Resumo

Internet das coisas (IoT) é um tema emergente e que promete mudar a forma como interagimos com o mundo ao nosso redor. Mas, apesar da relevância do tema, poucos são os profissionais qualificados para atuar na área. Dentro os muitos desafios enfrentados pelos novos desenvolvedores, um dos que se destaca é a parte de conectividade, pois, existem muitas opções e cada uma com suas peculiaridades. É fundamental para um projetista conhecer as diferenças de cada uma para poder escolher de forma efetiva qual tecnologia será utilizada em seu projeto. Este trabalho se propôs a desenvolver uma solução completa no contexto da IoT, buscando um sistema operacional de tempo real (RTOS) que ao mesmo tempo desse suporte a placa B-L475E-IOT01A e a soluções de conectividade. O objetivo era desenvolver uma rede entre algumas placas e conectar essa rede a internet. Por fim, uma aplicação seria desenvolvida para mensurar o desempenho da comunicação entre dois nós da rede por meio de um rádio Sub-1GHz. Depois de analisar vários RTOS, não foi encontrado nenhum que fornecesse o suporte desejado. Logo, foi abandonada a ideia de utilizar um RTOS. Sem suporte a conectividade, o objetivo de conectar os dispositivos a internet não foi realizado. Entretanto, a comunicação via Sub-1GHz entre as placas foi realizada e a aplicação e o teste da métrica de desempenho foram feitos.

Palavras-chave: Internet das coisas. Desempenho de rede. B-L475E-IOT01A. Wireless. Sub-1GHz.

Abstract

The Internet of Things (IoT) is an emerging topic that is expected to change the way we interact with the world around us. But despite the relevance of the topic, still there are few qualified people to work in this field. Among all the difficulties faced by new developers connectivity is one of the most prominent due to the number of options each one with its peculiarities. To effectively choose which technology will be incorporated in a project is essential to know all its differences. The paper was intended to develop a complete IoT solution, finding a Real Time Operating System (RTOS) with support to both connectivity and the B-L475E-IOT01A board. The goal was to develop a network between the boards and connect this network to the internet. Finally, an application would be developed to benchmark the communication link between two Sub-1GHz radios. After analyzing several RTOS, none were found that provided the desired support. Therefore, the idea of using an RTOS was dropped. With a lack of connectivity support, the goal of connecting the devices to the internet has not been accomplished. However, the link between the two Sub-1GHz radios was done, and the benchmark application was made.

Keywords: Internet of Things. network benchmark. B-L475E-IOT01A. Wireless. Sub-1GHz.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

6LoWPan	- IPv6 sobre uma rede de área pessoal de baixa potência (*IPv6 over Low Power Wireless Personal Area Networks*)
FSK	- Modulação por Chaveamento de Frequência (*Frequency-Shift Keying*)
IDE	- Ambiente de desenvolvimento Integrado (*Integrated Development Environment*)
IPv6	- Protocolo de internet versão 6 (*Internet Protocol version 6*)
IoT	- Internet das coisas (*Internet of Things*)
NFC	- Comunicação por Campo de Proximidade ()*Near-Field Communication*)
PDU	- Unidade de Dados de Protocolo (*Protocol Data Unit*)
RTOS	- Sistema Operacional de Tempo Real (*Real Time Operating System*)
RTT	- *Round Trip Time*

Sumário

1 Introdução

Imagine um mundo onde sensores estão espalhados por uma variedade de objetos comuns ao dia a dia, e que, através de uma rede sem fio estes objetos possam trocar informações entre si para a realização de alguma ação inteligente que facilite nossas atividades. Este é o cenário atual da internet das coisas (IoT), trata-se de uma tecnologia que está em amplo desenvolvimento, possui grande área de atuação, trará grandes benefícios econômicos em um futuro próximo (NING; HU, 2012) e promete revolucionar a forma como nós interagimos com o mundo ao nosso redor (STANKOVIC, 2014).

Apesar da relevância do tema, ainda há poucos profissionais com conhecimento no mercado, principalmente no que tange a parte de conectividade. Neste sentido, este trabalho busca desenvolver um cenário típico de IoT, conectando dispositivos inteligentes a uma rede sem fio e permitindo a comunicação desta com a internet por meio de um roteador. Desenvolvido este ambiente, o presente estudo objetiva desenvolver uma aplicação para avaliar parâmetros de desempenho de comunicação entre dois nós da rede criada.

1.1 Objetivos

Este trabalho de conclusão de curso tem como objetivos:

Objetivos Gerais

Criar uma rede de dispositivos inteligentes que podem ser acessados através da internet e extrair parâmetros que avaliem o desempenho e a qualidade da rede.

Específicos

- Conectar dispositivos inteligentes por meio de uma rede sem fio;
- Criar um roteador de borda e permitir que os dispositivos possam ser acessados por meio da internet;
- Desenvolver aplicação que permita extrair parâmetros de qualidade na comunicação entre dois nós da rede.

1.2 Metodologia

Inicialmente buscou-se verificar um cenário típico de uso da Internet das coisas, para que assim este pudesse ser implementado. Depois, fez-se uma pesquisa de sistemas

operacionais de tempo real (RTOS) que fossem compatíveis com a placa utilizada e que fornecessem as ferramentas necessárias para o desenvolvimento deste cenário. Uma vez que não foi encontrado nenhum sistema que desse suporte a placa e que fornecesse as ferramentas desejadas, e diante impossibilidade de fazer esse suporte manualmente antes da conclusão deste trabalho, optou-se por focar os esforços na avaliação de desempenho da rede Sub-1GHz. Em seguida, foi feita uma revisão bibliográfica sobre quais parâmetros devem ser medidos para avaliar o desempenho da rede e uma aplicação foi criada com essa finalidade. Por fim, a aplicação foi executada em dois dispositivos e os parâmetros da rede foram colhidos.

Além deste capítulo introdutório, o presente trabalho se divide em mais 6 capítulos conforme a seguir.

- **Capítulo 2:** breve introdução sobre internet das coisas, sistemas operacionais de tempo real e métricas de desempenho de uma rede;
- **Capítulo 3:** aborda o *hardware* utilizado;
- **Capítulo 4:** detalha a pesquisa feita para a escolha de um RTOS;
- **Capítulo 5:** descreve detalhes da construção do teste;
- **Capítulo 6:** apresentação e análise dos resultados;
- **Capítulo 7:** conclusões acerca do trabalho desenvolvido.

2 Fundamentação Teórica

2.1 Internet das Coisas (IoT)

A internet das coisas é uma ideia diretamente ligado à computação pervasiva. Trata-se de um conceito que considera a presença de inúmeros objetos, ligados por meio de conexões com ou sem fio e que possuem uma forma de endereçamento próprio. Estas "coisas" podem atuar de forma cooperativa e criar novos serviços e aplicações que tornem o nosso ambiente mais inteligente (VERMESAN; FRIESS, 2013).

O desenvolvimento da internet das coisas foi impulsionado pelas novas soluções de comunicações e de dispositivos semicondutores de baixa potência, de tal modo que pequenos dispositivos inteligentes contendo sensores, um microcontrolador e um controlador *wireless* pudessem ser construídos (KOPETZ, 2011). Com isto, tornou-se possível construir uma rede de objetos físicos que são capazes de escutar, ouvir, se comunicar uns com os outros, fazer tarefas em conjunto e coordenar decisões (AL-FUQAHA et al., 2015). Assim, cada objeto passou a ser visto como uma unidade operacional de um sistema maior.

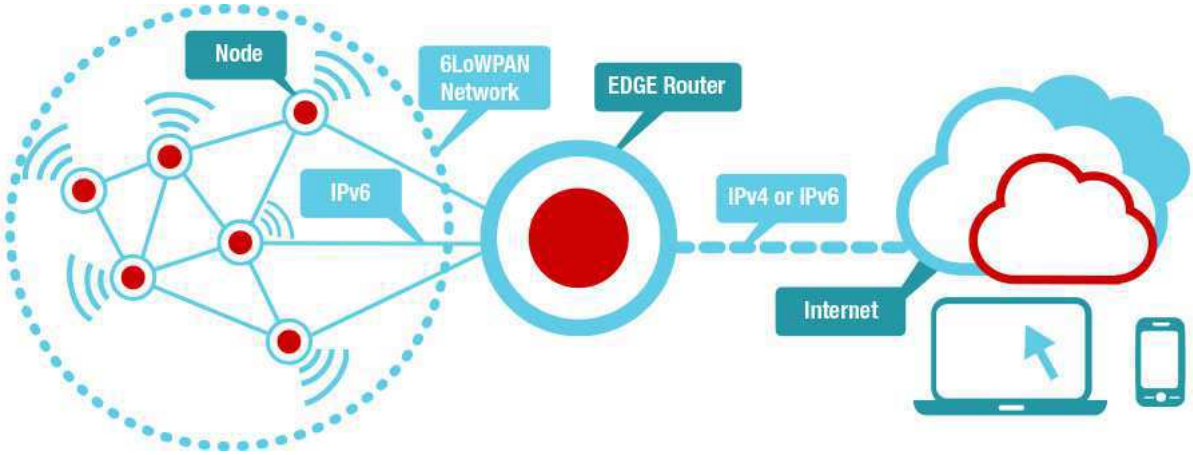
Na área industrial, por exemplo, livres das restrições dos cabos, os desenvolvedores estão descobrindo novas formas de melhorar a eficiência e produtividade na empresa e, aparentemente, a única limitação para as aplicações sem fio é a própria imaginação dos desenvolvedores (MONNIER; ZIGMAN; HAMMER, 2015). Além das aplicações industriais, IoT também está em forte desenvolvimento na domótica, cidades inteligentes, área médico hospitalar, militar, agricultura entre outras áreas.

Porém, uma solução completa em IoT pode demandar o conhecimento de várias tecnologias, desde diferentes protocolos de rede, roteamento, *hardware*, entre outras coisas. Na Figura 1, pode-se ver um cenário de implementação típica. Nela, os dispositivos inteligentes estão ligados entre si por meio de uma rede 6LoWPAN, e esta se conecta a um roteador de borda, que também pode ser um objeto inteligente, através do protocolo IPv6. Deste modo, o roteador conecta os dispositivos inteligentes a internet/nuvem e estes, por meio de aplicações, podem ser acessados ou controlados, individualmente ou em conjunto, de qualquer lugar.

Apesar da relevância do tema e do seu valor econômico, atualmente, há uma escassez de profissionais qualificados no mercado (NING; HU, 2012), talvez isso ocorra, porque, como se pode verificar, a implementação de um cenário de IoT completo, pode demandar o conhecimento de diferentes tecnologias.

Assim, como já visto, o ecossistema que a internet das coisas está inserido é complexo. Esta complexidade traz muitos desafios que não podem ser solucionados de

Figura 1 – Exemplo de um cenário IoT típico.



Fonte: (TI E2E Community, 2015).

forma eficiente sem fazer uso de um sistema operacional de tempo real (MILINKOVIĆ; MILINKOVIĆ; LAZIĆ, 2015).

2.2 Sistema Operacional de Tempo Real (RTOS)

Um sistema computacional de tempo real é um sistema cujo funcionamento depende não apenas que o processamento seja feito de forma correta, como também que o resultado seja produzido no momento certo (KOPETZ, 2011). Assim, um RTOS é um sistema operacional com funcionalidades que permitem o desenvolvedor trabalhar com funcionalidades de tempo real. Dentre os principais serviços oferecidos por um RTOS estão controle de tarefas, supervisão de entrada e saída, comunicação e sincronização entre tarefas (ABDELSAMEA; ZORKANY; ABDELKADER, 2016).

Um sistema operacional de tempo real pode garantir vantagem competitiva, diminuindo o tempo até o mercado e os custos de desenvolvimento de um produto. Entretanto, pesquisadores defendem que os RTOS precisam se atualizar para a era das aplicações IoT. Dentre essas atualizações estão abstração de *hardware* (ELVSTAM; NORDAHL, 2016) para suportar diferentes plataformas com poucas modificações de código; suporte a diferentes soluções de conectividade (ELVSTAM; NORDAHL, 2016); suportar os protocolos mais utilizados hoje pelas industrias como CAN, Bluetooth, Continua, ZigBee, Wi-Fi e internet (GRAHAM; WEINSTEIN, 2014), entre outros.

É importante salientar que os dispositivos comumente utilizados em internet das coisas apresentam consideráveis restrições de memória, processamento e consumo de energia. Sendo assim, um sistema operacional como o Linux, apesar de já possuir uma diversidade de protocolos e bibliotecas implementadas, não atende a estas restrições normalmente impostas aos dispositivos inteligentes (BACCELLI et al., 2013).

É vantajoso desenvolver qualquer sistema inteligente complexo utilizando um RTOS, não só pelo seu suporte multitarefa e pelo acurado controle de tempo, indispensáveis para trabalhar com conectividade de forma mais simplificada, como também para aproveitar a presença de várias bibliotecas integradas e maduras que simplifiquem o processo de desenvolvimento e diminuam as chances de eventuais *bugs* futuros.

2.3 Desempenho de uma rede

Com um grande número de dispositivos se espalhando pelo mercado utilizando diferentes bandas de frequência e diferentes protocolos, escolher a tecnologia que melhor se enquadra na sua aplicação sem fio não é tarefa fácil (LETHABY, 2017). Variadas aplicações IoT demandam diferentes configurações de velocidade, consumo de energia, confiabilidade. Igualmente, existem inúmeras opções de tecnologias e protocolos que podem ser escolhidas para integrar um projeto final. Deste modo, para uma escolha mais rápida e eficiente, é importante que o projetista conheça os requisitos de desempenho exigidos pelo projeto, para que assim ele possa escolher qual tecnologia será utilizada para atingir tais requisitos.

No geral, as métricas mais relevantes para medir o desempenho de uma rede são disponibilidade, perdas ou erro, latência e a largura de banda (HANEMANN et al., 2006).

A Huawei propôs a primeira métrica de rede voltada para IoT, este modelo é baseado em cinco dimensões. São estas: disponibilidade, vazão, alcance, *jitter* e eficiência energética (Huawei, 2016).

É importante atentar que podem haver divergências na forma como cada autor define algumas destas métricas. Sendo assim, será definido alguns termos importantes para este trabalho.

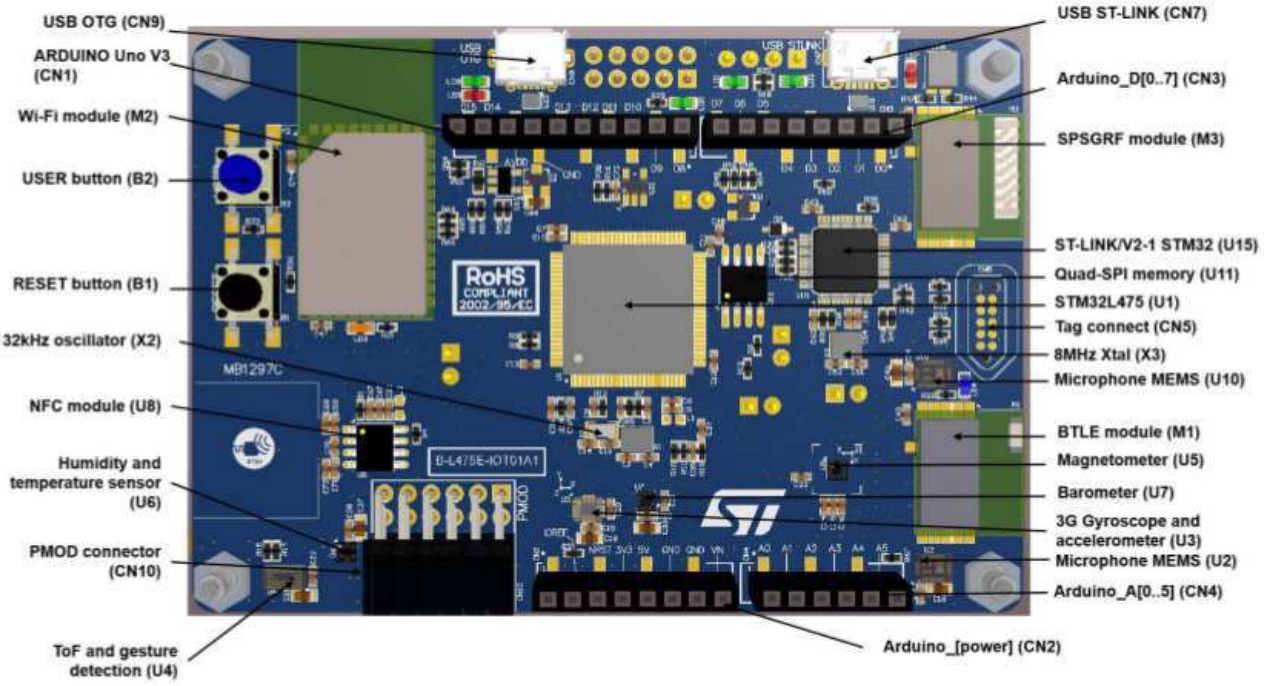
- Vazão - Número de *bits* por segundo que trafegam pelo canal de rede, incluindo o preâmbulo;
- RTT (*Round Trip Time*) - tempo que leva para enviar o pacote e obter a confirmação de que o sinal foi recebido (também conhecido como *ping time*);
- *Jitter* - o *jitter* neste trabalho está definido como o desvio padrão do RTT;
- Taxa de perda de pacotes - relação em termos percentuais entre pacotes sem confirmação de recebimento e todos os pacotes enviados.

3 Hardware Utilizado

3.1 Hardware

Neste trabalho foram utilizados dois exemplares da placa B-L475E-IOT01A. Esta placa é fabricada pela STMicroelectronics, contém um ARM Cortex-M4 com funções *ultra-low-power*, possui periféricos como Bluetooth, rádio Sub-1GHz, Wi-Fi, NFC e sensores como o de umidade, microfones omnidirecionais, magnetômetro de três eixos, giroscópio 3D, barômetro, *time of flight* e sensor de movimento (STMicroelectronics, 2017a). Isso faz dela uma placa com muitos recursos de conectividade para quem quer desenvolver projetos em IoT. O *layout* da placa pode ser visto na Figura 2.

Figura 2 – *Layout* da placa B-L475E-IOT01A.



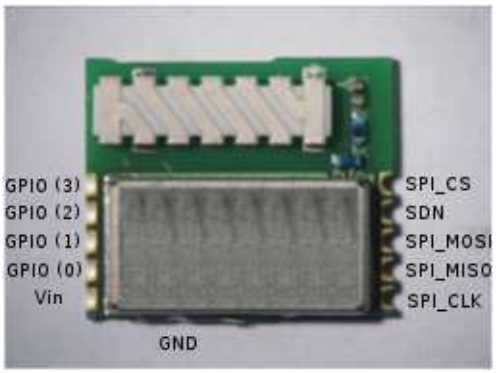
Fonte: (STMicroelectronics, 2017a).

3.1.1 Rádio SPSGRF

O rádio Sub-1GHz utilizado para os testes foi o SPSGRF-915. Ele trabalha na banda ISM de 915 MHz (ANATEL, 2015), possui esquemas de modulação 2-FSK, GFSK, MSK, GMSK, OOk, ASK, apresenta antena *on-board* e o *datasheet* afirma que sua velocidade pode variar de 1 até 500 kbps (STMicroelectronics, 2017b). Não foram encontradas muitas informações que fornecessem detalhes sobre o rádio utilizado, a melhor fonte encontrada

foi o arquivo de cabeçalho do *driver* que foi utilizado para controlar o rádio. Uma foto do rádio e a distribuição de seus terminais podem ser vistos na Figura 3.

Figura 3 – Imagem do rádio SPSGRF-915 e seus terminais.



Fonte: (STMicroelectronics, 2017b).

4 Escolha de um RTOS

Vários RTOS foram analisados em busca de encontrar características como: suporte a placa, aos rádios Sub-1GHz e Wi-Fi, a bibliotecas que facilitassem o uso de internet, e ao 6LoWPan. Infelizmente não foi encontrado nenhum RTOS que fornecesse o suporte de rede desejado que também suportasse a placa utilizada. Abaixo segue uma lista dos sistemas operacionais encontrados e as características que nos levaram a não adotá-los.

Contiki OS

Está entre os sistemas mais populares, especialmente em se tratando de redes de sensores sem fio (MILINKOVIĆ; MILINKOVIĆ; LAZIĆ, 2015). O Contiki OS não dá suporte ao microcontrolador STM32L4.

ChibiOS

Trata-se de um projeto com mais de 13 anos de idade desenvolvido para aplicações de tempo real que precisam de um código enxuto e eficiente. (MILINKOVIĆ; MILINKOVIĆ; LAZIĆ, 2015). O ChibiOS não suporta o microcontrolador STM32L4.

μC/OS

O Micro-Controller Operating Systems é um sistema de código fonte fechado desenvolvido pela Micriμm. O μC/OS possui certificações que asseguram a qualidade e a segurança contra falhas. Infelizmente, o microcontrolador STM32L4 também não é suportado por este sistema.

freeRTOS

Trata-se de um RTOS facilmente portável para diferentes microcontroladores. Entretanto, a plataforma não dá suporte nativo aos periféricos como UART, SPI, sensores, rádio SPSGRF e Wi-Fi.

Zephyr

Trata-se de um sistema operacional de tempo real relativamente novo, apenas dois anos de existência, capitaneado pela Linux Foundation (The Linux Foundation, 2016) e voltado para IoT. O suporte deste sistema a placa utilizada é limitado ao NVIC, UART, PINMUX, GPIO, I2C e PWM (The Zephyr Project, 2018).

Amazon FreeRTOS

Sistema operacional lançado no fim de novembro pela Amazon. Baseado no *kernel* do freeRTOS, este apresenta bibliotecas que facilitam a conexão com os serviços da Amazon Web Services (AWS) (Amazon, 2018). Este RTOS suporta grande parte dos periféricos da placa, dentre eles o Wi-Fi, porém não suporta o rádio Sub-1GHz. Também não foi encontrado referência ao suporte do 6LoWPAN.

RIOT OS (BACCELLI et al., 2013)

O RIOT OS foi testado quando o suporte parcial da placa ainda estava em um *pull request* em processo de revisão. Este RTOS tem um ótimo suporte a conectividade, porém, atualmente não há suporte a grande parte dos periféricos da placa, dentre eles, os rádios.

Na Tabela 1 é possível ver o suporte oferecido pelos RTOS a cada uma das característica desejadas. Uma vez que nenhum RTOS tinha o que era necessário, e diante da limitação em realizar o suporte da placa a um destes RTOS dentro do limite de tempo requerido para este trabalho, optou-se por uma solução *bare-metal*.

Tabela 1 – Suporte integrado oferecido pelos RTOS ao microcontrolador STM32L4, ao módulo Sub-1GHz SPSGRF, ao módulo Wi-Fi Inventek ISM43362-M3G-L44, a bibliotecas comuns na internet (como IPv4/IPv6, TCP, UDP) e ao 6LoWPAN.

RTOS	Microcontrolador	Sub-1GHz	Wi-Fi	Internet	6LoWPAN
Contiki OS	não	não	não	sim	sim
ChibiOS	não	não	não	sim	não
μC/OS	não	não	não	sim	não
freeRTOS	não	não	não	sim	não
Zephyr	sim	não	não	sim	sim
Amazon FreeRTOS	sim	não	sim	sim	não
RIOT OS	sim	não	não	sim	sim

Fonte: o autor.

5 Teste

Para iniciar a construção do ambiente de testes foi escolhido um código exemplo fornecido pela própria STMicroelectronics (STMicroelectronics, 2018) que demonstra a comunicação ponto a ponto entre duas placas B-L475E-IOT01A utilizando o rádio SPSGRF. A partir desse código, foram feitas as modificações necessárias na camada de aplicação, removendo a demonstração, criando um protocolo simples e adicionando a parte do teste.

O preâmbulo criado para os testes possui 5 Bytes de tamanho e tem o seguinte formato:

- Palavra de comando (1 *byte*): indica o comando que está sendo enviado;
- Numero de sequência (2 *bytes*): número de sequência do pacote sob teste;
- Tamanho dos dados (1 *byte*): quantidade de dados enviada no pacote descontando o tamanho do preâmbulo;
- Checksum (1 *byte*): valor para checar se o pacote chegou de forma integra.

O formato completo do pacote pode ser visto na Figura 4.

Uma vez elaborado o protocolo e as funções de enviar e receber pacotes, dividiu-se a aplicação de teste em duas, uma cliente, que sempre inicia as transações e o teste que será feito, e um servidor, que apenas responde as requisições do cliente. Nas seções abaixo será abordado como cada uma delas funciona.

Figura 4 – PDU do protocolo criado para os testes.

1 bytes	2 bytes	1 bytes	1 bytes	Variável
Comando	Nº de sequência	Tamanho dos dados	Checksum	Dados

Fonte: o autor.

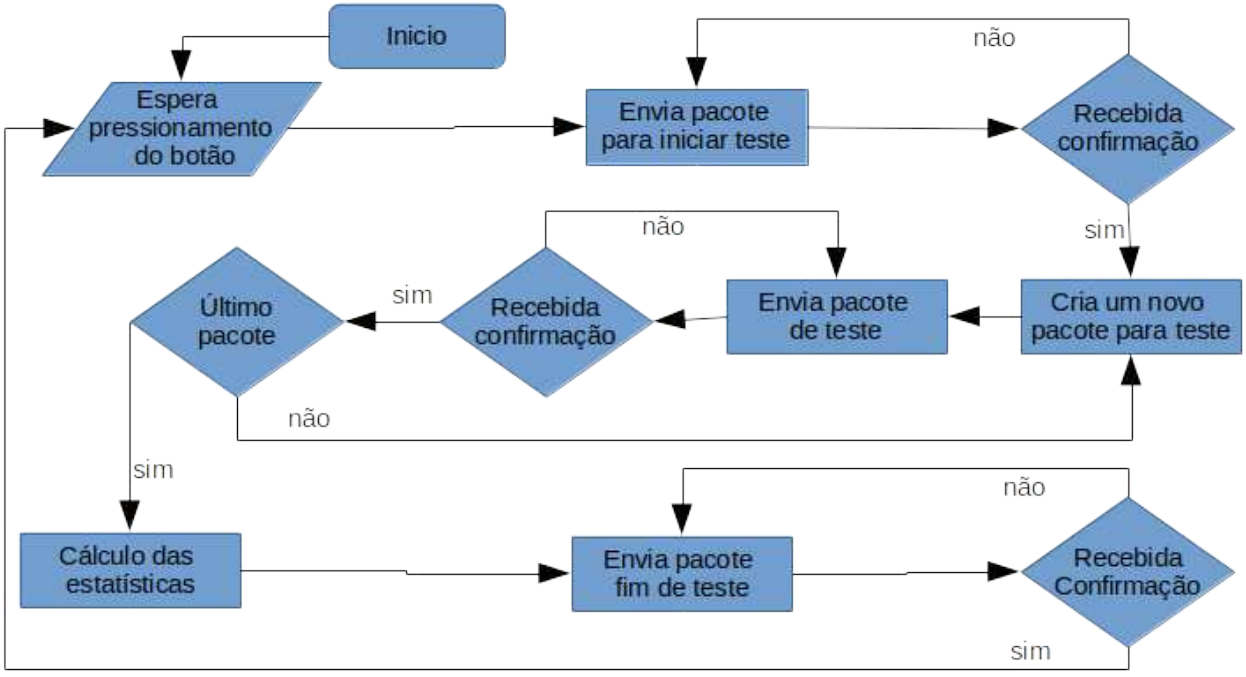
5.1 Aplicação cliente

Como supracitado, a aplicação cliente é que dá início aos testes. Assim que o operador aperta o botão para acionar o início dos testes, a cliente manda um pacote contendo o comando de início, a informação de quantos pacotes serão enviados durante o teste e fica aguardando a resposta do servidor. Caso o servidor não mande uma resposta de volta, o cliente reenvia o pacote de comando, e assim fica até receber uma resposta válida do servidor.

Uma vez que o servidor responde a solicitação do cliente, o cliente envia um pacote com número de sequência 0, contendo dados aleatórios, e novamente fica aguardando a confirmação de recebimento do servidor. Caso o servidor não responda dentro de um tempo limite, o cliente volta a reenviar o pacote até que a confirmação seja recebida. Só depois de recebida a confirmação do primeiro pacote é que o cliente manda o segundo pacote contendo novos dados aleatórios e com o número de sequência acrescido de um, e assim segue até que o último pacote tenha seu envio confirmado.

Só após o recebimento da confirmação do último pacote os cálculos das métricas de rede são feitos. Por fim, o cliente envia um comando ao servidor confirmando o fim do teste. Um fluxograma simplificado da aplicação cliente pode ser vista na Figura 5.

Figura 5 – Fluxograma simplificado do funcionamento da placa cliente.



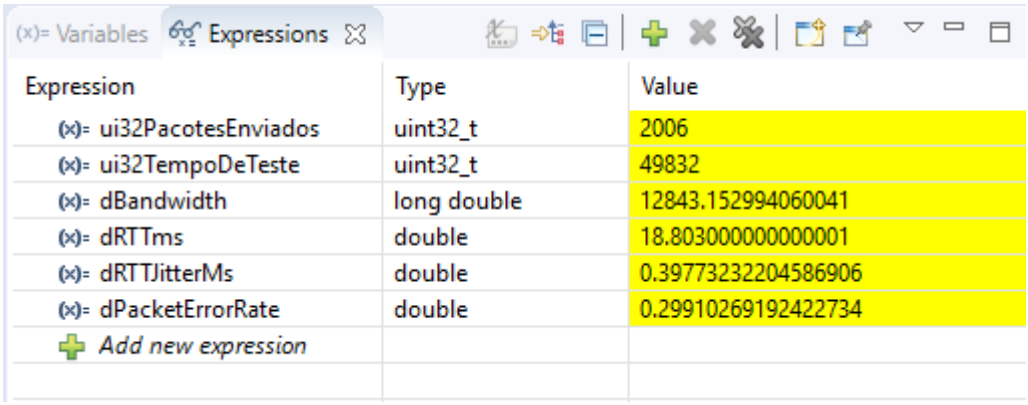
Fonte: o autor.

A coleta dos dados dos destes é feita dentro do debug dá própria IDE. Sempre ao fim de cada teste, o código para em um *breakpoint* e assim torna-se possível fazer a leitura das variáveis desejadas a partir de uma janela como a da Figura 6.

5.2 Aplicação servidor

A aplicação servidor fica sempre a escuta de um novo pacote. Ao receber um pacote com o comando indicando qual teste será iniciado, o servidor envia um pacote de confirmação e guarda o número de pacotes que o cliente enviará. Feito isso, volta a esperar por novos pacotes.

Figura 6 – Leitura dos dados por meio do debug.

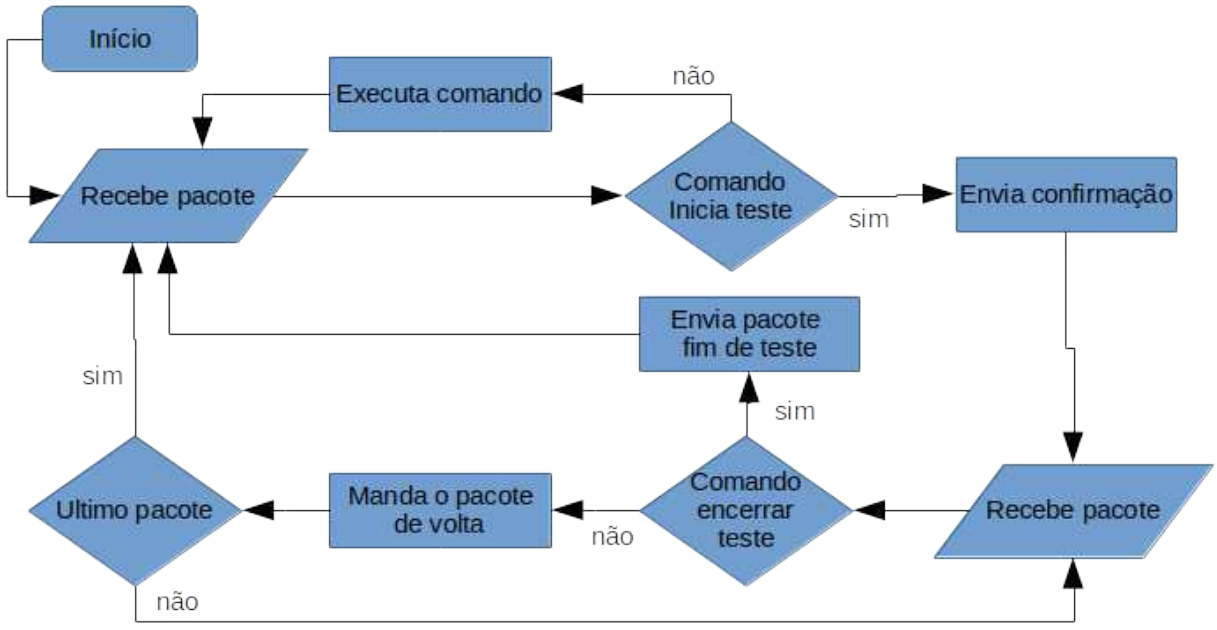


Fonte: o autor.

Para o teste realizado neste trabalho, sempre que um pacote com *checksum* correto chegar ao servidor, este o reencaminha de volta para o cliente para comprovar que o pacote foi recebido.

O fim dos testes para o servidor pode se dar de duas formas: 1. Quando o servidor receber último pacote, lembrando que o número de pacotes foi informado pelo cliente no pacote de iniciar testes; ou 2. ao receber um comando de fim de teste do cliente. O fluxograma simplificado do servidor pode ser visto na Figura 7.

Figura 7 – Fluxograma simplificado do funcionamento do servidor.



Fonte: o autor.

5.3 Cálculos das métricas da rede

O *software* desenvolvido tem a capacidade de mensurar os seguintes parâmetros:

- Vazão
- RTT
- Jitter
- Taxa de perda de pacotes

A aplicação cliente é a responsável por medir e calcular todos os parâmetros da rede. Como referência de tempo, algo fundamental para as medições, foi configurado uma interrupção que incrementa uma variável a cada milissegundo. Sendo assim, basta ler esta variável em um primeiro momento, e lê-la novamente em um outro instante para obter esta referência de tempo. O módulo da diferença das duas leituras será o tempo decorrido entre os dois momentos dado em milissegundos.

Antes de demonstrar como foram feitos os cálculos, é necessário definir algumas variáveis:

- PES - Pacote Enviados com Sucesso;
- T - Tamanho do pacote;
- t - tempo;
- RTT - *Round Trip Time médio*;
- RTT_i - *Round Trip Time* do pacote i;
- $TPP_{\%}$ - Taxa de perda de pacotes.

Concluídas as definições, agora é possível explicar como foi calculado cada parâmetro.

5.3.1 Cálculo da vazão

A vazão calculada representa o número de *bits* trafegado pelo canal, incluindo o preâmbulo e a resposta do servidor. Para encontrar o número de *bits* total multiplicou-se o PES pelo tamanho de cada pacote em *bytes* e depois multiplicou-se por 8 para o resultado ficar em *bits*. Por fim, multiplicou-se por 2 para levar em conta os dados enviados pelo servidor e obter o número total de *bits* trafegado pelo canal.

Já para obter o tempo total do teste, o tempo inicial é lido logo antes de enviar o primeiro pacote e depois relido logo depois de receber a confirmação do último pacote. Fazendo a diferença, obtém-se o tempo total. Deste modo, ao dividir o número de *bits* trafegados durante o teste pelo tempo total do teste, encontra-se a vazão. A fórmula utilizada para o cálculo da vazão está descrita na equação 5.1.

$$Vazão = \frac{2 \times PES \times T \times 8}{t} \tag{5.1}$$

5.3.2 Cálculo do RTT

O RTT_i é o módulo da subtração entre dois tempos, o tempo imediatamente antes do envio do pacote i pela aplicação cliente e o tempo imediatamente após o recebimento da confirmação. Caso haja perdas de pacote, o tempo de envio será atualizado a cada nova tentativa.

Assim sendo, o RTT calculado, conforme a equação 5.2, é a média do RTT_i de todos os i pacotes.

$$RTT = \frac{\sum_{i=1}^{PES} RTT_i}{PES} \tag{5.2}$$

5.3.3 Cálculo do jitter

O cálculo do *jitter*, conforme a equação 5.3, representa o desvio padrão do RTT_i.

$$jitter = \sqrt{\frac{\sum_{i=1}^{PES}(RTT_i - RTT)^2}{PES}} \tag{5.3}$$

5.3.4 Cálculo de perda de pacotes

O cálculo de perda de pacotes, equação 5.4, é realizado fazendo-se a subtração entre o número de tentativas de envios de pacotes e o número de pacotes confirmadamente recebidos, e dividindo-se o resultado por PE. O valor final é multiplicado por 100 para que a taxa se dê em termos percentuais.

$$TPP_{\%} = \frac{PE - PES}{PE} \times 100 \tag{5.4}$$

5.4 Realização do teste

Durante todos os experimentos alguns parâmetros foram mantidos fixos: 1- o número de pacotes enviados corretamente foi de 2000; o tamanho de cada pacote ficou

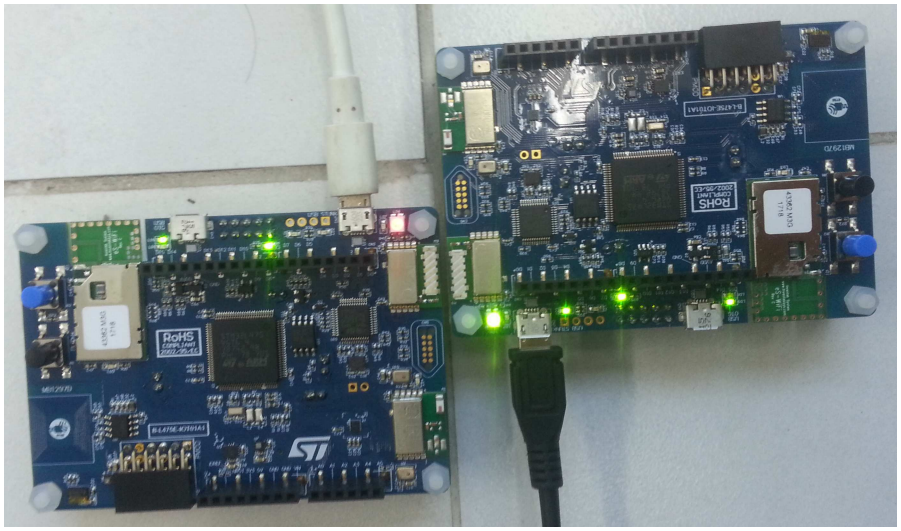
fixo em 20 *bytes*, sendo 5 de preâmbulo e 15 *bytes* de dados; 3- a modulação utilizada foi a 2-FSK.

Os testes foram realizados no primeiro pavimento do LIEC - UFCG, onde o piso, laje e parede são todos de concreto. Em relação a localidade, os experimentos podem ser divididos em dois ambientes: 1- com as placas dentro da mesma sala; e 2- em sala distintas. A planta baixo do primeiro pavimento do LIEC e os pontos de medição podem ser vistos da Figura 10.

5.4.1 Placas na mesma sala

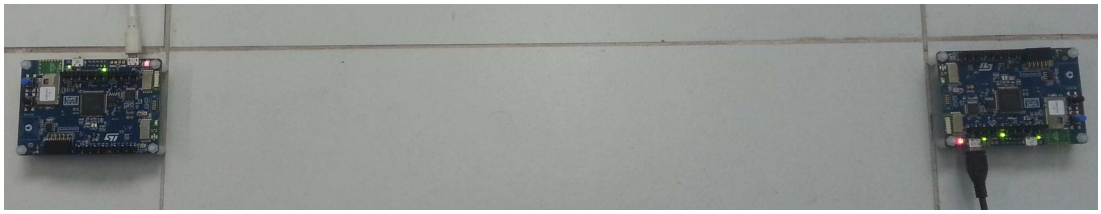
As placas foram colocadas no chão e foram feitas três medições em 3 diferentes posições: 1- os rádios quase colados um ao outro; 2- placas 50 cm de distância; e 3- placas a 8 m de distância. As fotos da disposição das placas com os rádios quase colados e a 50 cm podem ser vistas nas Figuras 8 e 9 respectivamente. As posições dos pontos de medição estão na Figura 10 onde "o" representa a posição da placa cliente, que não se moveu, e "a", "b" e "c", os locais onde foram posicionadas a placa servidor.

Figura 8 – Teste com os rádios a milímetros um do outro.



Fonte: o autor.

Figura 9 – Teste com rádios a 50 cm um do outro.

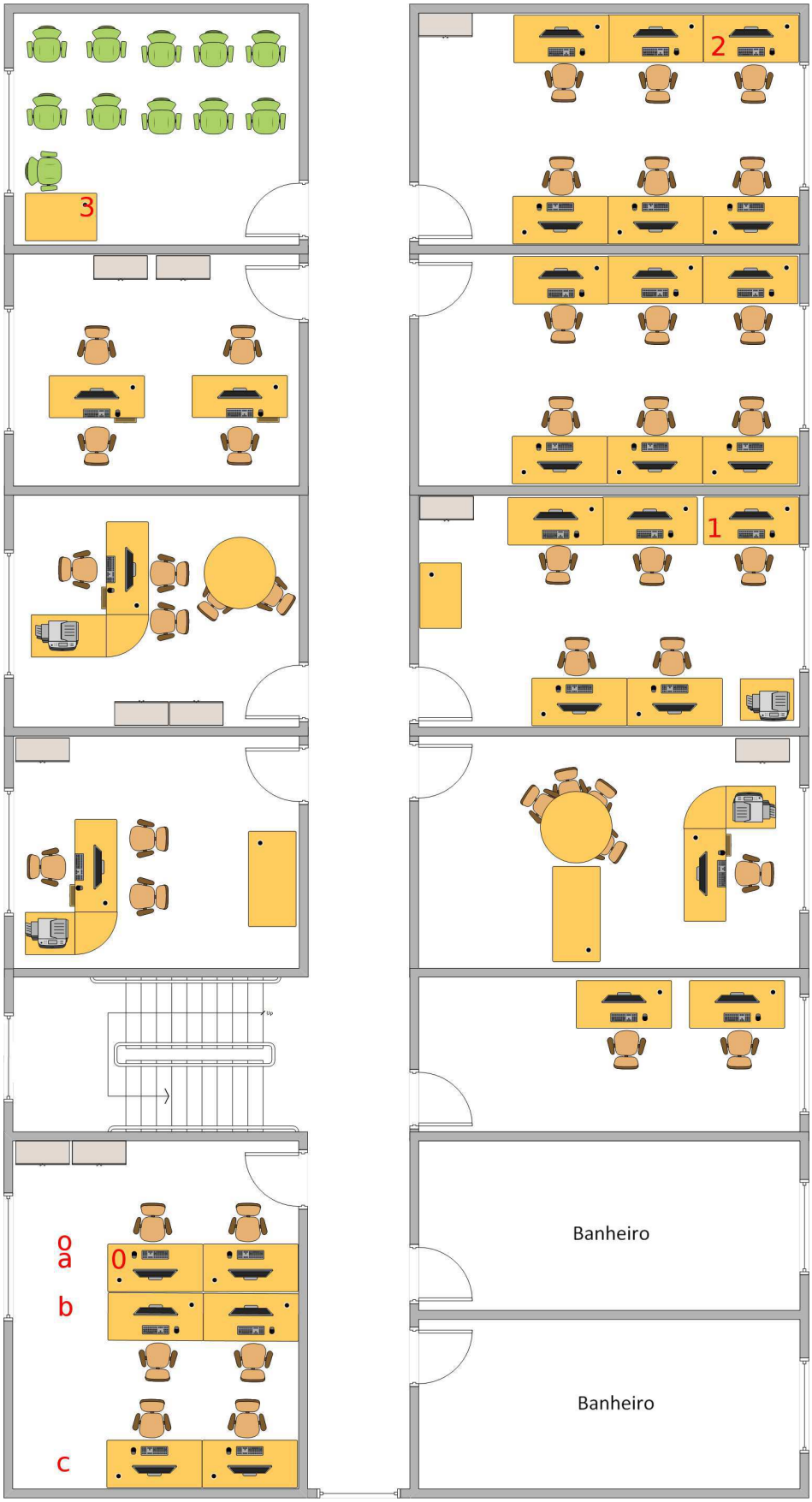


Fonte: o autor.

5.4.2 Placas em salas distintas

A placa cliente, durante todos os experimentos, ficou fixa em cima da mesa na posição "0" conforme a Figura 10. Já a placa servidor foi movida e os testes foram realizados com esta nas posições "1", "2" e "3". Em todas as posições foram feitos testes com as placas na horizontal. Nas posições "1" e "2" fez-se a medição com a placa na vertical e o rádio virado para cima. Na posição "1", o teste com a placa na vertical com o rádio virado para baixo também foi feito.

Figura 10 – Planta baixa do pavimento e posição das medições.



Fonte: o autor.

6 Resultados e análise

6.1 Placas na mesma sala

Os resultados das medições dentro da sala em diferentes distâncias podem ser vistos na Tabela 2.

Tabela 2 – Resultados obtidos para os experimentos feitos dentro de uma sala em diferentes distâncias.

Posições	Número do teste	Distância (m)	Tempo (s)	Vazão kbits/s	RTT (s)	Jitter (s)	TPP %
$\overline{oa}$	1	≈0	37,62	17,01	18,78	0,414	0,00
$\overline{oa}$	2	≈0	39,66	16,14	18,79	0,406	0,05
$\overline{oa}$	3	≈0	41,66	15,36	18,78	0,412	0,10
$\overline{ob}$	1	0,50	37,63	17,00	18,80	0,405	0,00
$\overline{ob}$	2	0,50	37,61	17,02	18,78	0,412	0,00
$\overline{ob}$	3	0,50	37,62	17,01	18,78	0,413	0,00
$\overline{oc}$	1	4,00	37,63	17,01	18,79	0,406	0,00
$\overline{oc}$	2	4,00	37,62	17,01	18,79	0,409	0,00
$\overline{oc}$	3	4,00	37,63	17,00	18,79	0,403	0,00

Fonte: dados do autor.

6.2 Placas em salas distintas

Os resultados das medições em diferentes distâncias podem ser vistos na Tabela 3.

Tabela 3 – Resultados obtidos para os experimentos feitos em diferentes salas do LIEC; os resultados com "-" indicam que devido a alta taxa de perda de pacotes, os testes não concluíram dentro do tempo de 210 segundos.

Posições	Orientação do rádio	Distância (m)	Tempo (s)	Vazão kbits/s	RTT (s)	Jitter (s)	TPP %
$\overline{01}$	horizontal	16,0	37,64	17,00	18,80	0,400	0,00
$\overline{01}$	horizontal	16,0	37,63	17,01	18,79	0,407	0,00
$\overline{01}$	cima	16,0	37,64	17,00	18,79	0,405	0,00
$\overline{01}$	baixo	16,0	80,14	7,99	18,80	0,399	1,04
$\overline{02}$	horizontal	23,0	82,27	7,78	18,80	0,397	1,09
$\overline{02}$	horizontal	23,0	155,2	4,12	18,79	0,406	2,82
$\overline{02}$	cima	23,0	-	-	-	-	-
$\overline{03}$	horizontal	18,5	-	-	-	-	-

Fonte: dados do autor.

6.3 Análise dos Resultados

Observando os resultados, o RTT e o *Jitter* se mantiveram praticamente constantes, com médias de 18,79 ms e 0,406 ms respectivamente. Neste caso, podemos concluir que, para as configurações de rádio utilizadas, a distância entre as placas e as barreiras entre elas não influenciam no RTT e no *Jitter*.

A informação do RTT e o seu respectivo desvio, neste trabalho o *Jitter*, é relevante, pois pode ajudar o projetista da rede a decidir quanto tempo esperar pela confirmação ou não de um pacote. Um RTT muito curto faz com o que vários pacotes sejam reenviados antes de receber a confirmação. Já um RTT muito longo pode fazer que se espere muito por um pacote que não irá chegar. Ambos os casos prejudicam o desempenho da rede.

Curiosamente, no caso em que os dois rádios estavam a milímetros de distância entre si, houveram algumas perdas de pacotes. Apesar das perdas terem sido poucas, é importante o desenvolvedor está atento a esta possibilidade.

Percebeu-se que as perdas de pacote afetou muito o desempenho da rede, apenas 1% de perdas fez com que o teste dobrasse em tempo de execução. Neste sentido, conclui-se que as perdas de pacotes tem forte influência sob a vazão e sobre o desempenho da rede. Para entender o motivo de uma queda tão brusca no desempenho é necessário entender melhor o funcionamento do *hardware*.

A partir do resultados, pode-se verificar que a distância não é o único fator que influência o desempenho da rede. Enquanto que na posição $\overline{02}$, a 23 m de distância houve comunicação e teste, nos locais $\overline{03}$, a 18,5 m os testes não foram concluídos devido a alta taxa de perda de pacotes. Pela Figura 10 é possível ver a quantidade de armários situados entre as posições “0” e “3”. O primeiro deles, próximo a posição “0”, é metálico. Estes armários, em especial o metálico, podem ter contribuindo para que o sinal não trafegasse, entre os dois pontos, de forma adequada.

Também pode-se concluir que não apenas o ambiente, como também a posição dos rádio podem influenciar nos resultados. Isto pode ser visto por exemplo na quantidade de perda de pacotes na posição $\overline{01}$ com o rádio virado para baixo quando comparado a outras posições e na posição $\overline{02}$ com o rádio virado para cima relativamente a posição horizontal.

Os testes, realizados em diferentes salas, não foram feitos em ambientes controlados, portas ocasionalmente eram abertas, pessoas passavam próximas e por vezes permaneciam na frente das placas sob teste. Uma vez que as condições ambientais influenciam no resultado, acredita-se que as condições citadas podem ter contribuindo com variabilidade no resultado dos testes, como aconteceu na posição $\overline{02}$ com o rádio na horizontal.

Uma vez que para a IoT o consumo de energia, em muitos casos, deve ser o mínimo possível, então, para uma rede onde se sabe que não haverá perda de pacotes o projetista

da rede pode optar por tentar diminuir a potência do sinal e verificar se a qualidade da rede continua a mesma. Já no caso de uma alta taxa de perda de pacotes, uma possibilidade é fazer uso de repetidores conforme a Figura 1.

7 Considerações Finais

IoT é um tópico em alta, emergente e promete continuar assim por muito tempo. O mercado ainda está crescendo, as opções são inúmeras e muitas novidades, tecnologias e ideias continuarão a aparecer. Porém, por se tratar de um tópico novo, ainda mais neste ecossistema tão heterogêneo quanto o da internet das coisas, existem muitos desafios a serem solucionados. Atualmente, há uma demanda de novos desenvolvedores e ao mesmo tempo existe uma grande barreira técnica a ser vencida por estes profissionais e pelas empresas que estão investindo neste mercado.

Neste sentido, o mais importante no desenvolvimento deste trabalho não foi a elaboração da aplicação nem o resultado final obtido, mas sim o caminho percorrido e a experiência adquirida. Foi feito um caminho exploratório dentro da realidade atual do mundo IoT. Realizar testes com o Amazon FreeRTOS com menos de uma semana de lançamento, como também a execução testes com um suporte parcial do RIOT OS a placa utilizada, quando este ainda era *pull request* em processo de revisão, comprovam que este trabalho esteve lidando com o que era de mais novo na época. Desta forma, o autor enfrentou as mesmos obstáculos e barreiras que um desenvolvedor tem ao lidar com algo novo. Dentre estes obstáculos, pode-se mencionar, a dificuldade de encontrar suporte ao *hardware*, a periféricos e a parte de conectividade.

O objetivo de realizar o cenário completo não foi atingido, entretanto, mesmo sem o auxílio de um RTOS, o objetivo de criar uma aplicação para testar o desempenho da rede foi realizado. Os resultados encontrados são valiosos no sentido de ajudar um projetista a saber se esta tecnologia satisfaz os requisitos desejados. E caso seja escolhida, ajudar no processo de planejamento da infraestrutura da rede e no conhecimento de suas limitações.

Para finalizar, é reiterado os benefícios da utilização de um RTOS para aplicações envolvendo internet das coisas, como também a relevância em se conhecer previamente as características de uma tecnologia de rede de modo que o desenvolvedor possa escolher de forma mais rápida e eficiente qual tecnologia irá entrar em seu protótipo ou produto final.

No mais, como sugestões de melhoria, novos trabalhos podem adicionar mais parâmetros, por exemplo, a potência de recepção do sinal e a demanda de energia. Sugiro também que novos testes possam ser feitos em diferentes condições, como com distintos tamanhos de pacote e com outras modulações.

Referências

ABDELSAMEA, M. H. A.; ZORKANY, M.; ABDELKADER, N. Real time operating systems for the internet of things, vision, architecture and research directions. In: IEEE. *Computer Applications & Research (WSCAR), 2016 World Symposium on*. [S.l.], 2016. p. 72–77. Citado na página 26.

AL-FUQAHA, A. et al. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, IEEE, v. 17, n. 4, p. 2347–2376, 2015. Citado na página 25.

Amazon. *Amazon FreeRTOS*: Sistema operacional da iot para microcontroladores. 2018. Disponível em: <hhttps://aws.amazon.com/pt/freertos/>. Acesso em: 2 dez. 2018. Citado na página 32.

ANATEL. *Atribuição de Faixas de Frequências no Brasil*. 2015. Disponível em: <http://www.anatel.gov.br/Portal/verificaDocumentos/documento.asp?numeroPublicacao=325100&pub=principal&filtro=1&documentoPath=325100.pdf>. Citado na página 29.

BACCELLI, E. et al. Riot os: Towards an os for the internet of things. In: IEEE. *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*. [S.l.], 2013. p. 79–80. Citado 2 vezes nas páginas 26 e 32.

ELVSTAM, A.; NORDAHL, D. Operating systems for resource constraint internet of things devices: An evaluation. Malmö högskola/Teknik och samhälle, 2016. Citado na página 26.

GRAHAM, B.; WEINSTEIN, M. The rtos as the engine powering the internet of things. *white paper*, 2014. Citado na página 26.

HANEMANN, A. et al. A study on network performance metrics and their composition. *Campus-Wide Information Systems*, Emerald Group Publishing Limited, v. 23, n. 4, p. 268–282, 2006. Citado na página 27.

Huawei. *A Things-Oriented Network Planning Methodology*. 2016. White Paper. Citado na página 27.

KOPETZ, H. *Real-time systems: design principles for distributed embedded applications*. [S.l.]: Springer Science & Business Media, 2011. Citado 2 vezes nas páginas 25 e 26.

LETHABY, N. *Wireless connectivity for the IoT: One size does not fit all*. [S.l.], 2017. Citado na página 27.

MILINKOVIĆ, A.; MILINKOVIĆ, S.; LAZIĆ, L. Choosing the right rtos for iot platform. *INFOTEH-JAHORINA*, v. 14, p. 504–509, 2015. Citado 2 vezes nas páginas 26 e 31.

MONNIER, O.; ZIGMAN, E.; HAMMER, A. *Understanding wireless connectivity in the industrial iot*. [S.l.], 2015. Citado na página 25.

NING, H.; HU, S. Technology classification, industry, and education for future internet of things. *International Journal of Communication Systems*, Wiley Online Library, v. 25, n. 9, p. 1230–1241, 2012. Citado 2 vezes nas páginas 23 e 25.

STANKOVIC, J. A. Research directions for the internet of things. *IEEE Internet of Things Journal*, IEEE, v. 1, n. 1, p. 3–9, 2014. Citado na página 23.

STMicroelectronics. *STM32L4 Discovery kit IoT node, multi-channel communication with STM32L4*. [S.l.], 2017. Rev. 3. Citado na página 29.

STMicroelectronics. *Sub-GHz (868 or 915 MHz) low power programmable RF transceiver modules*. [S.l.], 2017. Rev. 6. Citado 2 vezes nas páginas 29 e 30.

STMicroelectronics. *STM32CubeL4*. 2018. Disponível em: <https://my.st.com/content/my_st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-mcu-packages/stm32cubel4.html#getsoftware-scroll>. Acesso em: 01 fev. 2018. Citado na página 33.

The Linux Foundation. *The Linux Foundation Announces Project to Build Real-Time Operating System for Internet of Things Devices*. 2016. Disponível em: <www.linuxfoundation.org/press-release/the-linux-foundation-announces-project-to-build-real-time-operating-system-for-internet-of-things-devices> Acesso em: 14 mar. 2018. Citado na página 31.

The Zephyr Project. *ST Disco L475 IOT01 (B-L475E-IOT01A)*. 2018. Disponível em: <http://docs.zephyrproject.org/boards/arm/disco_l475_iot1/doc/disco_l475_iot1.html>. Acesso em: 14 mar. 2018. Citado na página 31.

TI E2E Community. *Contiki-6LoWPAN*. 2015. Disponível em: <http://processors.wiki.ti.com/index.php/Contiki-6LOWPAN>. Acesso em: 15 fev. 2018. Citado na página 26.

VERMESAN, O.; FRIESS, P. *Internet of things: converging technologies for smart environments and integrated ecosystems*. [S.l.]: River Publishers, 2013. Citado na página 25.