**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS**
**CENTRO DE CIÊNCIAS EXATAS, AMBIENTAIS E DE TECNOLOGIA**

**LEONARDO DELFORNO**

# BANCADA DE EMULAÇÃO DE CANAL DE RÁDIO PARA RSSF EM INTERNET DAS COISAS

**CAMPINAS**
**2020**

LEONARDO DELFORNO

# BANCADA DE EMULAÇÃO DE CANAL DE RÁDIO PARA RSSF EM INTERNET DAS COISAS

Dissertação apresentada como exigência para obtenção do Título de Mestre em Gestão de Redes de Telecomunicações, ao Programa de Pós-Graduação em Engenharia Elétrica, do Centro de Ciências Exatas, Ambientais e de Tecnologias, da Faculdade de Engenharia Elétrica, da Pontifícia Universidade Católica de Campinas.

Orientador: Prof. Dr. Frank Herman Behrens

**CAMPINAS**
**2020**

LEONARDO DELFORNO

# BANCADA DE EMULAÇÃO DE CANAL DE RÁDIO PARA RSSF EM INTERNET DAS COISAS

Dissertação apresentada como exigência para obtenção do título de Mestre em Gestão de Redes de Telecomunicações ao Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências Exatas, Ambientais e de Tecnologias.
Área de Concentração: Engenharia Elétrica.
Orientador: Prof. Dr. Frank Herman Behrens.

Dissertação defendida e aprovada em 05 de fevereiro de 2020 pela Comissão Examinadora constituída dos seguintes professores:

Prof. Dr. Frank Herman Behrens
Orientador da Dissertação e Presidente da Comissão Examinadora
Pontifícia Universidade Católica de Campinas

Profa. Dra. Cecilia de Freitas Morais
Pontifícia Universidade Católica de Campinas

Prof. Dr. Geraldo Peres Caixeta
Universidade São Francisco - USF

Ficha catalográfica elaborada por Vanessa da Silveira CRB 8/8423
Sistema de Bibliotecas e Informação - SBI - PUC-Campinas

006.22
D351b

Delforno, Leonardo

Bancada de emulação de canal de rádio para RSSF em internet das coisas / Leonardo Delforno. - Campinas: PUC-Campinas, 2020.

75 f.: il.

Orientador: Frank Herman Behrens.

Dissertação (Mestrado em Gestão de Redes de Telecomunicações) - Programa de Pós-Graduação em Engenharia Elétrica, Centro de Ciências Exatas, Ambientais e de Tecnologia, Pontifícia Universidade Católica de Campinas, Campinas, 2020.
Inclui bibliografia.

1. Internet das coisas. 2. Radiofrequência. 3. Redes de sensores sem fio. I. Behrens, Frank Herman. II. Pontifícia Universidade Católica de Campinas. Centro de Ciências Exatas, Ambientais e de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDD - 23. ed. 006.22

# AGRADECIMENTOS

Primeiramente, a Deus, por ter me dado saúde e forças até o presente momento.

À minha família, que esteve disposta a sempre me apoiar e incentivar.

Ao Prof. Dr. Frank Herman Behrens, por ser incentivador, guia e amigo nessa trajetória.

Aos amigos, em especial Carlos, Ernesto, Guilherme, Marcelo, Pedro, Raphael e Débora pelo apoio e incentivo nesses dois anos juntos.

Ao Prof. Dr. Omar Carvalho Branquinho, por todo apoio e conhecimento.

À Pontifícia Universidade Católica de Campinas, pela concessão da bolsa no Programa de Mestrado, sem a qual este trabalho não poderia ter sido realizado.

*“A menos que modifiquemos a nossa forma de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo.” (Albert Einstein)*

# RESUMO

DELFORNO, Leonardo. BANCADA DE EMULAÇÃO DE CANAL DE RÁDIO PARA RSSF EM INTERNET DAS COISAS. Dissertação, Programa de Pós-graduação em Engenharia Elétrica, Pontifícia Universidade Católica de Campinas, Campinas, 2020.

Com o crescente uso da Internet das Coisas (IoT, do inglês *Internet Of Things*), o conhecimento do comportamento do canal de conexão sem fio de Redes de Área Ampla e Baixo Consumo (do inglês *Low Power Wide Area Networks,* LPWAN) é uma questão-chave para a implementação de redes de sensores sem fio (RSSF) em redes urbanas, industriais e residenciais. Para isso, os profissionais que irão implementar essas redes precisam entender o comportamento de um canal de rádio dinâmico, submetido a diversas condições de propagação. Assim, esse trabalho busca auxiliar no treinamento, ensaio e no ensino do comportamento do canal de comunicação entre os nós de uma Rede LPWAN por radiofrequência (RF). Assim, este trabalho apresenta uma bancada de emulação de canais de comunicação via rádio de baixo custo, aplicáveis a redes de sensores sem fio, a ser utilizada por instituições de ensino e pesquisa ou para treinamento na investigação do comportamento das propriedades de propagação e atenuação do canal no âmbito de LPWANs. Como resultado, é possível preparar profissionais para atuar na implementação e operação das tecnologias LPWAN aplicáveis à IoT.

**Palavras-chave:** emulação, radiofrequência, canal de comunicação, RSSF, LPWAN, Internet das Coisas.

# ABSTRACT

DELFORNO, Leonardo. RADIO CHANNEL EMULATION WORKBENCH IN INTERNET OF THINGS. Dissertation, Master in Telecommunication Network Management, Pontifícia Universidade Católica de Campinas, Campinas, 2020.

With the increasing use of the Internet of Things (IoT), knowledge of the behavior of the Low Power Wide Area Networks wireless channel is a key issue for the deployment of wireless sensors networks (WSN) in urban, industrial and residential networks. For this, the professionals who will implement these networks need to understand the behavior of a dynamic radio channel, subjected to various propagation conditions. Therefore, a solution is needed to assist in testing, training, or even learning the behavior of the communication channel between LPWAN Radio Frequency (RF) Network nodes. Thus, this work presents a low-cost radio communication channel emulation bench, applicable to wireless sensor networks, to be used by educational and research institutions or for training in the investigation of the propagation and attenuation properties of the system channel under LPWANs. As a result, professionals can be prepared to work on the implementation and operation of LPWAN technologies applicable to IoT.

**Keywords:** Emulation, radiofrequency, communication channel, WSN, LPWAN, Internet of Things.

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

BER = *Bit Error Rate* (Taxa de erro de bit).

ERB = Estação Rádio Base (em inglês *Radio Base Station*).

IDE = Integrated Development Environment (Ambiente de desenvolvimento integrado).

IoT = *Internet of things* (Internet das coisas).

IP = *Internet protocol* (Protocolo de Internet).

ISM = *Industrial Sientific and Medical bands* (Faixa Industrial, Científica e Médica).

LNA = *Low Noise Amplifier* (Amplificador de baixo ruído).

LPWAN = *Low Power Wide Area Network* (Rede de Área Ampla de Baixa Potência.).

LR-WPAN = *Low-Rate Wireless Personal Area Network* (Rede de área pessoal sem fio de baixa taxa).

LTE = *Long term evolution* (Evolução a Longo Prazo).

NS = Nó Sensor (no inglês *Sensor Node*).

PA = *Power Amplifier* (Amplificador de Potência).

PC = *Personal Computer* (Computador Pessoal).

PDF = *Probability Distribution Function* (Função de distribuição de probabilidade.).

QoS = *Quality of service (*Qualidade de serviço).

RF = *Radio Frequency* (Radiofrequência).

RSSI = *Received Signal Strength Indicator* (Indicador de intensidade do sinal recebido).

RSSF = Rede de sensores sem fio (do inglês *Wireless Sensor Network).*

SLA = *Service level agreement (*Suporte ao nível de serviço).

SNR = *Signal to Noise Ratio (*Relação Sinal Ruído).

TCP = *Transmission control protocol (Protocolo de Controle de Transmissão).*

WLAN = Wireless Local Area Network (Rede local sem fio).

# SUMÁRIO

## INTRODUÇÃO

A Internet das Coisas (IoT) é uma realidade e cada vez mais aplicada em ações úteis que visam à melhora do monitoramento e do controle dos processos industriais, ambientes urbanos, ou, simplesmente, trazer mais conforto ou segurança, como em automação residencial e aplicações em cidades inteligentes.

Em contrapartida, as necessidades para implementar a IoT passam pelas carências locais das aplicações, pois, os problemas estão nas infraestruturas encontradas nas cidades, nas indústrias, nas casas, e demais outras. Para a concretização desta realidade, um elemento fundamental é a comunicação sem fio para a construção de uma rede de nós sensores ou atuadores (RSSF – rede de sensores sem fio), conectados a um elemento concentrador (*gateway*), de forma a permitir a conexão dos elementos da rede com a Internet. Várias tecnologias vêm sendo propostas para atender diferentes cenários, como LoRa e SigFox (RAZA; KULKARNI; SOORIYABANDARA, 2017).

Assim, o conhecimento de como funcionam as RSSFs é uma questão fundamental para a efetiva implementação de uma aplicação IoT. Contudo, os profissionais que irão criar essas redes necessitam de treinamento ou de um ambiente de ensaios para realização de projetos, para atender, principalmente, pequenas e médias empresas e pequenas cidades, uma vez que os empreendimentos de grande porte terão condições de adotar soluções comerciais fechadas, oriundas de fornecedores de tecnologia tradicionais do mercado.

Ademais, o ensino, treinamento ou a experimentação de aplicações de redes sem fio encontra dificuldades, muitas vezes, na complexidade da montagem de experimentos que permitam reproduzir os fenômenos encontrados nesse tipo de conexão, quando se utiliza comunicação através de ondas de rádio não confinadas. Nos ambientes de laboratório seria difícil, por exemplo, reproduzir cenários de atenuação em espaço livre ou efeitos de atenuação do sinal de radiofrequência (RF) causados pelo ambiente ou por obstáculos.

Este trabalho apresenta uma estratégia de emulação das condições de comunicação sem fio, considerando as perturbações sofridas pelo sinal de radiofrequência. Para criar as condições que permitam entender esses processos, é proposto um ambiente de emulação de um canal rádio, denominado de Bancada de Emulação. Com esta estratégia, é possível impor ao sinal de rádio as flutuações de

intensidade do sinal com base em modelos de propagação. O emulador reproduz os fenômenos de propagação, tais como atenuação no espaço livre, log-normal, *Rayleigh*, e outros.

Dessa forma, é possível avaliar conceitos de comunicação em condições reais, com experimentos passíveis de serem realizados em laboratório e que seriam difíceis de serem controlados no ambiente real.

A vantagem da Bancada de Emulação é a possibilidade de avaliar o efeito do canal nas comunicações, no que diz respeito à camada física do modelo TCP/IP. Além disso, são explorados parâmetros do enlace de comunicação como modulação, taxa de transmissão, potência, *offset* de frequência, dentre outros parâmetros. Em particular, existe a necessidade de se entender como o sistema de comunicação reage em relação às condições de propagação no canal. O trabalho propõe essa análise através da emulação do canal de RF, de forma que o sinal de radiofrequência seja confinado a um cabo coaxial de modo a simular uma propagação ideal sem interferências.

As condições de atenuação reais encontradas na prática são obtidas através de atenuadores programáveis controlados segundo um modelo específico definido no momento da emulação. Fazendo-se uso do sistema cabeado, é possível emular qualquer ambiente, desprezando as interferências externas.

## 1.1 Objetivo principal

O objetivo desta dissertação consiste no desenvolvimento de uma Bancada de Emulação de canal de rádio frequência de baixo custo, para utilização em instituições de ensino e pesquisa, ou, para ensaios em empresas de projeto. Esta permitirá reproduzir as atenuações sofridas pelo canal de radiofrequência em diferentes cenários.

## 1.2 Organização do trabalho

O trabalho está organizado da seguinte forma:

O **capítulo 2** apresenta os conceitos relacionados a uma RSSF – discutindo um Modelo de Referência para implementação de soluções de IoT – características de um link de rádio, modelos de propagação e protocolos utilizados em RSSFs para IoT.

O **capítulo 3** apresenta a proposta da Bancada de Emulação, mostrando sua funcionalidade e aplicação.

O **capítulo 4** apresenta a implementação da proposta desse trabalho, discutindo os materiais e métodos utilizados.

O **capítulo 5** apresenta os resultados obtidos.

O **capítulo 6** apresenta as conclusões, considerações finais e propostas para trabalhos futuros.

# REVISÃO BIBLIOGRÁFICA E EMBASAMENTO TEÓRICO

O universo de *Internet of things* (Internet das coisas ou IoT), é composto por uma série de elementos característicos, como sensores, atuadores, processadores, comunicação, aplicações, tratamento e visualização dos dados – para citar alguns. Conhecer as características de cada componente e como são integrados para buscar uma solução IoT é de suma importância. Neste capítulo, são abordados temas relacionados a esse assunto e pertinentes ao trabalho, com amparo na literatura científica.

## 2.1 Modelo de Referência

A necessidade de sistematizar e organizar as muitas disciplinas e tecnologias envolvidas nos vários níveis de qualquer solução IoT levou à criação de um Modelo de Referência do *Open* IoT, proposto por Déo (2018).

Este modelo facilita a identificação de possíveis dificuldades de implementação, especialmente para soluções de IoT desenvolvidas para pequenas e médias empresas.

A atenção com esse tipo de aplicação surge do alto custo das soluções oferecidas por algumas grandes empresas. Dada essa situação, o modelo visa guiar o desenvolvedor até a implementação completa de soluções que operam em plataformas *Open Source*.

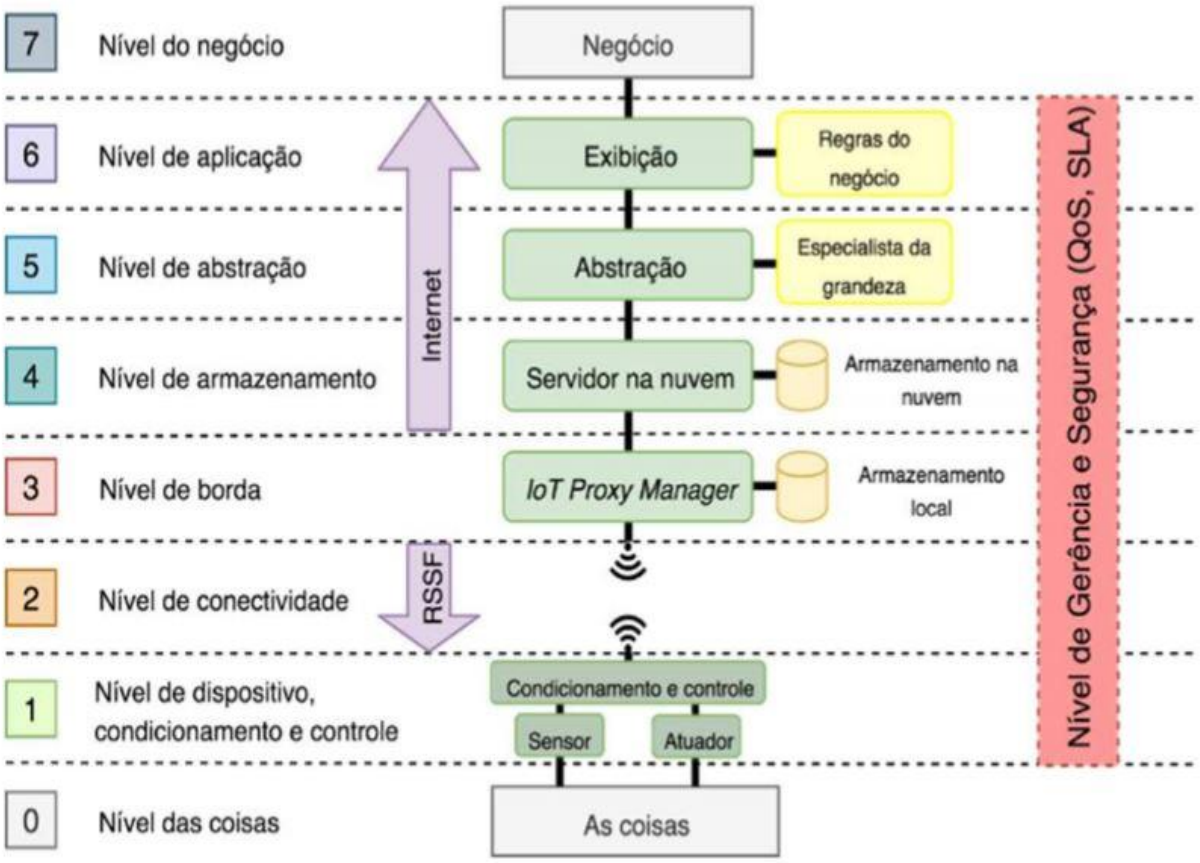
### 2.1.1 Arquitetura

Primeiramente, uma solução de IoT origina da necessidade de se coletar alguma medida ou controlar algo, para atender algum objetivo necessário a um negócio. Quando se parte de uma necessidade real, existe um interessado e um objetivo claro a ser atendido, que refletirá a agregação de valor em algum ponto do negócio. Portanto, necessita-se de estabilidade e garantia de funcionamento, ou seja, a solução a ser desenvolvida precisa ser confiável, gerenciável e segura (DÉO, 2018).

A Figura 1 apresenta a proposta de um Modelo de Referência para desenvolvimento de aplicações de IoT, proposto por Déo (2018), que cobre todas as etapas a serem realizadas, da coleta do dado à disponibilização da informação, auxiliando o negócio na tomada de decisão. É possível observar os níveis, suas

correlações e seus elementos. Cada nível, elemento e necessidade será detalhado nas seções subsequentes.

Figura 1 – Modelo de Referência de IoT.




Fonte: Adaptado de Déo (2018, p. 59).

### 2.1.2 Visão Conceitual de um Modelo de Referência de IoT

Como já discutido anteriormente, toda solução de IoT parte da necessidade de monitorar algo para que determinados objetivos do negócio sejam atendidos. Para que essa ligação das “coisas” com o “negócio” aconteça, algumas etapas precisam ser realizadas. Dessa forma, é essencial:

– A coleta, o tratamento e a transmissão dos dados;

– Um elemento de borda, que permita o controle do processo local e envie os dados para a Internet;

– Uma solução na nuvem que receba e armazene esses dados tratados pelo elemento de borda;

– A abstração dos dados, transformando-os em informações. Nesse ponto, é necessário um especialista que seja capaz de identificar as necessidades para coleta e tratamento dos dados;

– Uma interface que permita a visualização e monitoramento dos dados;

– O sistema levar em consideração questões de gerência e segurança, para garantir que todas as etapas sejam cumpridas de forma satisfatória, garantindo a integridade, confidencialidade e disponibilidade.

Esse é o fluxo que os dados percorrem a pilha do modelo de referência dos dados à exibição, sendo transformados ao longo do processo em informação (DÉO, 2018).

### 2.1.3 Nível 0 – Nível das Coisas

Nesse nível, estão presentes os elementos físicos que possuem a necessidade de monitoramento. Por exemplo, para uma solução IoT em um sistema de controle de uma plantação, a coleta de grandezas físicas associadas à aplicação (tais como temperatura, umidade do ar, umidade do solo) são imprescindíveis para o negócio (DÉO, 2018).

### 2.1.4 Nível 1 – Nível de dispositivo, condicionamento e controle

Uma vez que existe um interessado e uma necessidade definida, é preciso descobrir como coletar os dados de interesse por meio de sensores e o que eles representam ao negócio. Em certas situações, torna-se necessário também intervir no processo monitorado através de atuadores, como consequência dos resultados da análise de dados segundo as regras do negócio. Neste nível, entram, principalmente, os elementos sensores e atuadores (DÉO, 2018).

Os sensores são os dispositivos que transformam as grandezas físicas em sinais elétricos, que podem ser digitais ou analógicos. Exemplificando, um sistema de controle de uma plantação citado acima, seriam definidos, neste nível, quais sensores devem ser utilizados para coleta de temperatura, umidade do ar e umidade do solo. Da mesma forma, seria definido ainda como intervir sobre a plantação por meio dos atuadores, a partir das medições coletadas, por exemplo, para ligar a irrigação.

### 2.1.5 Nível 2 – Nível de conectividade

Os dados coletados precisam chegar à nuvem como uma informação útil ao negócio através de um elemento de borda. Para isso, existem várias opções para conectividade, sejam elas sistemas proprietários ou de código fonte aberto. Pode-se classificá-las em três categorias principais:

– Soluções de Internet: Protocolos proprietários de RSSF com disponibilização dos dados na Internet por meio do protocolo TCP/IP. Como, por exemplo, SigFox, LoraWan, LTE NoB e *Wi-Fi*;

– Soluções de Conectividade: Protocolos proprietários que utilizam protocolos específicos, como, por exemplo, LoRa, ZigBee, Z-WAVE e *BLUETOOTH*;

– Soluções Open Source: Protocolos de código-fonte aberto que permitem a construção de redes especialistas para atender problemas específicos, como, por exemplo, o RADIUINO (2020). Soluções como essa permitem a adequação às necessidades, ao contrário de outras propostas, em que as necessidades precisam se adaptar à solução.

Neste nível, é importante que a solução adotada possua uma capacidade de escoamento dos dados compatível com o serviço que está sendo prestado. Em geral, as tecnologias possuem limitação de taxa de comunicação em função de questões como:

- Baixa velocidade de transferência;
- Perda de pacote;
- Compartilhamento de grande quantidade de sensores.

Um equívoco comum é imaginar que todos os dados produzidos localmente serão transferidos para o armazenamento na nuvem. Os dados a serem transmitidos – bem como sua frequência e possíveis tratamentos – são definidos de acordo com a necessidade do negócio e das tecnologias escolhidas. Deve ser possível, nessa etapa, transmitir um dado coletado, como, por exemplo, uma temperatura. Também, deve ser permissível receber a parametrização da temperatura para definição de novo comportamento local no Nível 1 e, ainda, transmitir as ações de controle enviadas pelo cliente, no caso de necessidades específicas, como desligar um atuador (DÉO, 2018).

A conectividade é, portanto, a forma de conexão entre o sensoriamento e o processamento/armazenamento das informações coletadas. A falta de conectividade rompe o elo entre o mundo físico e o mundo digital, impossibilitando que as

informações coletadas pelos sensores possam ser utilizadas para atendimento do negócio. Entender o comportamento do canal de RF para diferentes cenários significa permitir o correto funcionamento da solução IoT, considerando a opção de conectividade sem fio.

O trabalho proposto está focado nessa fase do Modelo de Referência IoT, permitindo a emulação do nível de conectividade para diferentes situações, prevendo o comportamento do canal em diferentes cenários.

### 2.1.6 Nível 3 – Nível de Borda

Como os dados coletados são provenientes de uma RSSF, mas têm como objetivo final a Internet, precisa-se de um elemento que realize a ligação desses dois universos. Este elemento normalmente é chamado Elemento de Borda ou *Gateway*. A borda consiste na interface entre a RSSF e a Internet (DÉO, 2018).

Está presente também nesse elemento de borda um *software* que faz o papel de *middleware*. Nesse *software*, serão configuradas as estratégias de armazenamento local, pré-tratamento dos dados e encaminhamento para a nuvem.

### 2.1.7 Nível 4 – Nível de Armazenamento

Uma vez que os dados atingem a nuvem, não basta exibi-los em tempo real. É preciso armazená-los e, para isso, são necessários dois elementos:

– Servidor na nuvem: permite que os usuários processem altas cargas de trabalho e armazenem grandes volumes de informação. Em geral, é contratado para que não exista a dificuldade de gerenciar o funcionamento deste armazenamento, além de existir a facilidade de alterar a capacidade do mesmo de maneira simples, geralmente alterando o plano de contratação;

– Capacidade de armazenamento: o principal impedimento nesse quesito são os custos envolvidos, posto que, em uma solução usando um armazenamento em nuvem, a cobrança é feita de acordo com o armazenamento e, ao se adotar um armazenamento próprio, surgem questões como *backup*, capacidade de ampliação do armazenamento e consumo de energia.

### 2.1.8 Nível 5 – Nível de Abstração

Nesse nível, os dados são transformados em informações. Para isso, são necessários dois elementos:

– Abstração: implementação de *software* com a capacidade de tratar os dados e deles extrair informações dos processos;

– Especialista: elemento humano externo que determina a estratégia de transformação dos dados em informação (DÉO, 2018).

### 2.1.9 Nível 6 – Nível de apresentação

No nível 6, é estabelecida a interface com o cliente final no Nível 7 (Negócio). Define-se como as informações serão apresentadas, se existem alertas e como eles se comportam e se esses dados serão utilizados para interação com outras ferramentas (DÉO, 2018).

### 2.1.10 Gerência e Segurança

De forma vertical ao longo dos diversos níveis, deve-se ter a preocupação com as questões de gerência e segurança. A gerência é dividida em duas partes:

– Gerência da Rede: é responsável pelos aspectos relativos à operação da rede nos seus diversos níveis. Ela garante o bom funcionamento por meio de avaliação de métricas para determinar a qualidade de serviço (QoS, *Quality of Service*) e dá suporte ao nível de serviço (SLA, *Service Level Agreement*);

– Gerência dos Dados: é responsável pela integridade dos dados nos diversos níveis. Essa interage com cada um dos níveis para garantir que os dados coletados, processados, abstraídos e exibidos estejam em conformidade com as necessidades dos clientes (DÉO, 2018).

## 2.2 Link de Rádio

Em aplicações envolvendo comunicação sem fio, é indispensável entender os efeitos sofridos pelo sinal de rádio, no processo de propagação, principalmente por fornecer a modelagem necessária para a estimativa da potência de transmissão requerida em uma comunicação confiável.

Enquanto o meio de transmissão que usa cabos é praticamente determinístico, na comunicação sem fio sempre existirá um nível de incerteza que deve ser considerado: as redes cabeadas possuem taxas de erro de *bit* (BER, *Bit Error Rate*) na ordem de $10^{-3}$ a $10^{-6}$. Isso traduz em uma taxa de perda de segmento de aproximadamente 1,2% para segmentos de 1500 *bytes*. BER são muito maiores em um domínio sem fio, geralmente na ordem de $10^{-3}$, e às vezes até $10^{-1}$. Com BER na ordem de $10^{-3}$, a taxa de perda de pacotes é de uma magnitude maior em um ambiente sem fio (cerca de 12%) (LEE, 1993).

Tendo em vista uma rede bastante simples, composta por um nó base ligado a um computador, um nó sensor (que monitora um parâmetro qualquer) e o canal de propagação (pelo qual os dados trafegam), é necessário pensar nos efeitos do canal de RF sobre o sistema para garantir a qualidade de comunicação, de forma adequada entre dois nós. Não é conveniente pensar em aplicações e utilização dos dados, se não houver garantia que esses dados serão enviados.

A intensidade recebida do sinal cai com o aumento da distância em qualquer meio de transmissão. Entretanto, em sistemas sem fio, essa atenuação é bastante forte em função da distância, e também do tipo de ambiente em que esteja funcionando o sistema. Assim, deve ser feito um cuidadoso planejamento para que a atenuação sofrida pelo sinal não afete o desempenho da rede.

Ademais, a qualidade do *link* também pode ser prejudicada por fatores como interferências, efeitos de propagação de vários caminhos (*multi-path*) ou até mesmo problemas de *hardware. Links* aéreos apresentam características específicas que serão discutidas a seguir.

### 2.2.1 Características Espaciais

O alcance do sinal não é uniforme e nem isotrópico, sendo definido por três regiões de formato irregular (BACCOUR et al., 2012):

- Conectadas: geralmente onde os *links* são estáveis e de boa qualidade;

- Transicional: onde os *links* podem ser assimétricos, instáveis e de qualidade intermediária;

- Desconectadas: onde os *links* são inadequados para comunicações.

### 2.2.2 Características Temporais

A variação da qualidade do *link* ao longo do tempo está relacionada a mudanças no ambiente, fatores variáveis no tempo (como temperatura e umidade, presença humana, interferências e obstáculos fixos ou móveis), geram um impacto na qualidade do *link*. Para Cavalcanti (2018), existem três padrões para variações temporais de qualidade de *link*:

- Pequenas flutuações: causadas pelo desvanecimento de sinais em vários caminhos;

- Grandes flutuações/perturbações: causadas pelo efeito de sombreamento de pessoas e outros objetos;

- Grandes flutuações contínuas: causadas por interferências de ondas eletromagnéticas perenes.

### 2.2.3 Assimetria de Links

A assimetria de *links* pode afetar o desempenho da comunicação. *Links* assimétricos apresentam diferenças nas características de conectividade entre *uplinks* e *downlinks*. Descobriu-se que a assimetria de *hardware* é a principal causa de assimetria de *link*, causada por discrepâncias na calibração de *hardware*, ou seja, diferentes potências de transmissão e ruído entre os nós.

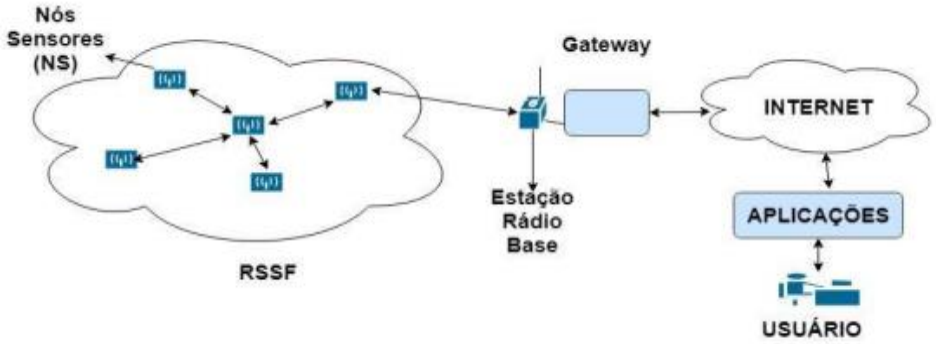
### 2.2.4 Interferências

Interferências podem ocorrer em transmissões pelo ar, uma vez que o meio é compartilhado por diversos nós transmissores. Essas interferências podem ser internas ou externas. Interferências internas podem ocorrer quando os nós pertencentes à RSSF transmitem ao mesmo tempo. Já a interferência externa pode ser causada por outras redes ou dispositivos operando na mesma banda de frequência que a RSSF.

## 2.3 Componentes de uma RSSF

As Redes de Sensores Sem Fio (RSSF), tradicionalmente, são compostas de alguns componentes como: Nó Sensor (NS), Estação Rádio Base (ERB) e *Gateway*. Na Figura 2, são mostrados os principais componentes de uma RSSF.

Figura 2 – Componentes básicos de uma RSSF.



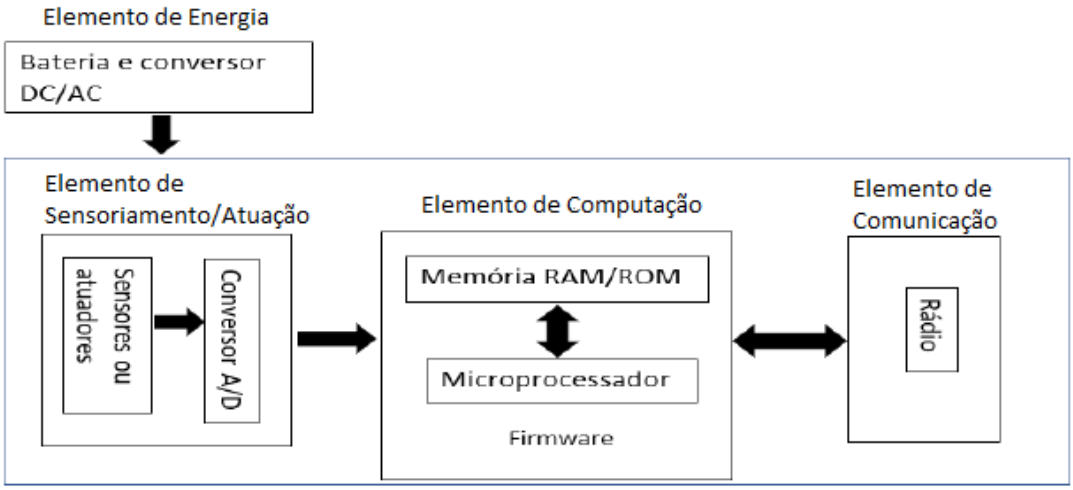

Fonte: Freitas e Branquinho (2017).

### 2.3.1 Nó Sensor

Um Nó Sensor (NS) é um componente que abriga um ou mais sensores, responsáveis por coletar as grandezas físicas do ambiente, tais como: temperatura, pressão, vazão, luminosidade, dentre outras. Além de coletar os dados provenientes dos sensores e entregá-los à ERB, os NS também possuem a função de roteamento (AKYILDIZ et al., 2002), já que em alguns casos é necessário passar por outros NS para se chegar à ERB.

Um NS é composto basicamente por um elemento de computação baseado em um microcontrolador, memória, sensores e atuadores, fonte de energia e dispositivo de comunicação (FREITAS, 2018). Como bem visto na Figura 3, são mostrados os principais componentes de um NS.

Figura 3 – Principais componentes de um Nó Sensor.




Fonte: Freitas (2018).

### 2.3.2 Estação Rádio Base (ERB)

A Estação Rádio de Base, segundo Ilyas e Mahgoub (2014, tradução nossa):

> Consiste em um componente responsável por receber, armazenar e processar os dados de vários NS. A ERB é ligada a um computador e comunica-se com os NS sem a utilização de cabos. Portanto, a função da ERB é fazer a conexão entre a RSSF e uma rede cabeada.

### *2.3.3 Gateway*

Um *gateway* é um elemento responsável por conectar a RSSF com uma rede externa, normalmente uma rede de computadores (LIMA et al., 2009). Seu único objetivo em uma RSSF é realizar uma passagem do meio de comunicação sem fio para um meio cabeado.

## 2.4 Limitações das Redes de Sensores Sem Fio

As RSSF possuem algumas limitações importantes que devem ser levadas em consideração no momento de sua implementação, configuração e gerenciamento (FREITAS, 2018).

Uma vez que, sem uma efetiva comunicação, os dados não são transportados até o elemento de borda, para Bento (2009):

> Os NS utilizados na RSSF são geralmente, limitados em termos de capacidade de processamento, assim como também de armazenamento. Isso dificulta o desenvolvimento de algumas técnicas, como, por exemplo, a implementação de algoritmos de segurança.

Segundo Panda (2014) acerca de outro problema, seguindo a mesma linha de raciocínio:

> Outro problema crítico está relacionado à questão de capacidade energética. É muito comum a implantação de RSSF em locais de difícil acesso, onde a substituição das baterias dos sensores é uma tarefa que será realizada o mínimo de vezes possível.

As interências em redes sem fio são comuns, principalmente quando utilizam-se faixas de frequência não licenciadas em sua implantação, como a de 2,4 GHz, por exemplo. Alguns equipamentos que também utilizam essas faixas de frequência, pelo fato dessas serem gratuitas, podem interferir no sinal do canal de comunicação, causando perda de pacotes (RUFINO, 2014).

## 2.5 Modelos de Propagação

As telecomunicações móveis apresentam algumas complicações de propagação muito particulares, tornando a comunicação sem fio confiável mais difícil do que as comunicações fixas. A implantação de redes de telecomunicações sem fio, às vezes, exige a instalação de equipamentos próximos aos limites operacionais dos transceptores, sendo necessário entender os fatores que podem interferir na qualidade do sinal (RAPPAPORT, 2009).

Em vista disso, a modelagem de propagação de sinal é uma ferramenta importante no projeto de sistemas sem fio. Assim, os seguintes modelos foram desenvolvidos para prever o comportamento de propagação em diferentes ambientes.

### 2.5.1 Espaço Livre

O Modelo de Propagação no Espaço Livre é o mais básico e, também o mais importante, pois determina a menor atenuação que normalmente se obtém em um enlace de rádio. Esse modelo foi proposto por Harald T. Friis do *Bell Telephone Laboratory* em 1946. A dedução da fórmula de Friis considera, primeiramente, uma antena isotrópica na transmissão e a potência de transmissão *Pt* que nela é injetada.

De acordo com Rappaport (2009):

> O Modelo de Propagação no Espaço Livre assume as condições ideais de propagação, onde as antenas de transmissão e recepção estão localizadas em um ambiente livre de obstáculos de absorção, ou que reflete superfícies e com um caminho de linha de visão claro entre elas. H. T. Friis apresentou a Equação 1 para calcular a potência do sinal recebido *Pr* no espaço livre à distância *d* do transmissor.

A Equação 1 apresenta a fórmula de Friis para calcular a potência do sinal recebido *Pr* no espaço livre à uma distância *d* do transmissor:

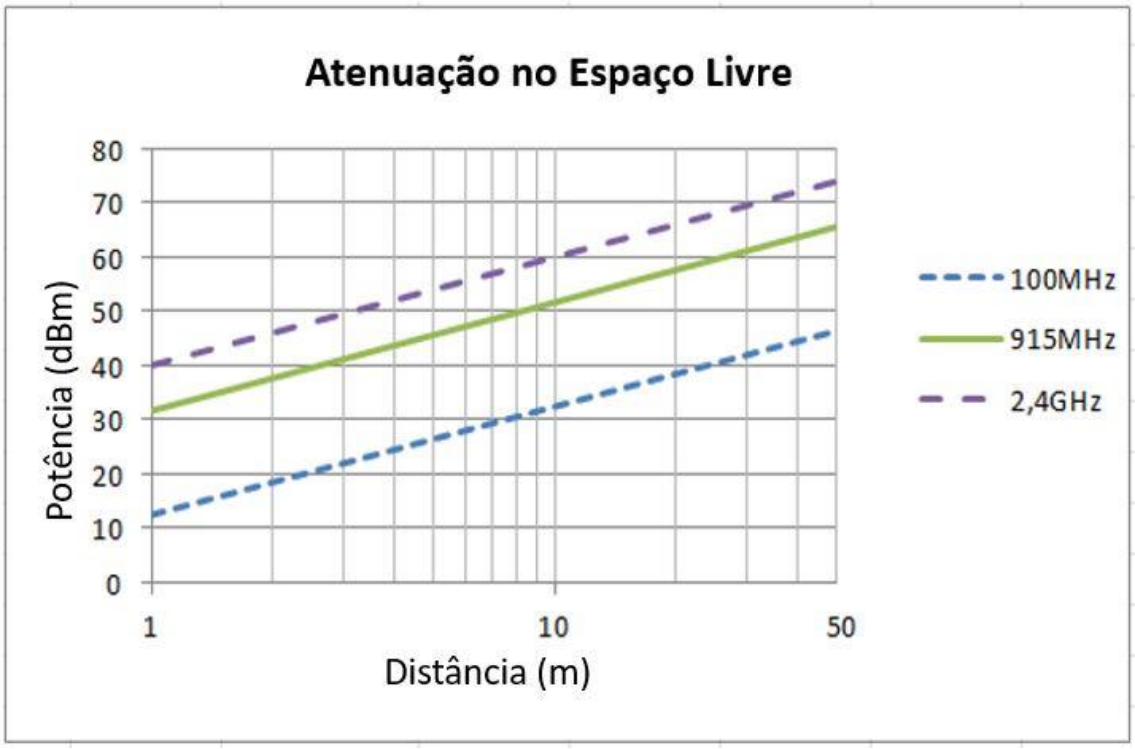
$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \quad (1)$$

Na Equação 1, $P_t$ representa a potência do sinal transmitido, $L$ representa a perda do sistema, os ganhos das antenas emissora e receptora são representados por $G_t$ e $G_r$ respectivamente , e por fim, $\lambda$ é o comprimento de onda definido pela Equação 2 em função da velocidade da luz no vácuo $v$ e da frequência $f$ do sinal de rádio:

$$\lambda = \frac{v}{f} \quad (2)$$

A Figura 4 apresenta o valor de *Pr* calculado pela Equação 1 em função da distância *d* e da frequência do sinal de rádio.

Figura 4 – Atenuação no espaço livre.

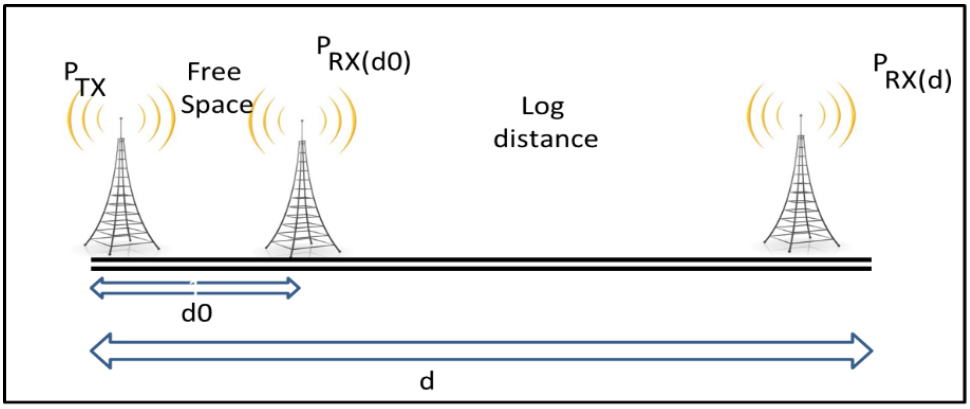



Fonte: Elaboração própria.

### 2.5.2 *Log Distance*

O Modelo de Propagação *Log Distance* é baseado em resultados experimentais sobre os tipos de ambiente em que o sinal de rádio se propaga. Esse modelo parte do princípio de que a potência recebida, a uma distância *d,* pode ser calculada levando em consideração um fator denominado *path loss,* e, uma potência de referência recebida a uma distância $d_0$ próxima do transmissor, conforme descrito na Figura 5 (RAPPAPORT, 2009).

Figura 5 – Modelo *Log Distance.*




Fonte: Elaboração própria.

Entre o transmissor e a posição $d_0$ (que deve ser bem menor que $d$), é considerada a atenuação no espaço livre. Após esse ponto, o sinal sofrerá uma atenuação maior que a do espaço livre, dependendo da perda de percurso representada por β, de acordo com a Equação 3, onde $P_{rx}$ representa a potência de recepção, $P_{tx}$ a potência de transmissão e $\beta$ uma variável que representa um fator da atenuação.

$$P_R(d) = P_R(d_0) - 10\beta log\left(\frac{d}{d_0}\right) \quad (3)$$

A variação do valor de $\beta$ permite caracterizar os diferentes tipos de ambientes, conforme a Tabela 1.

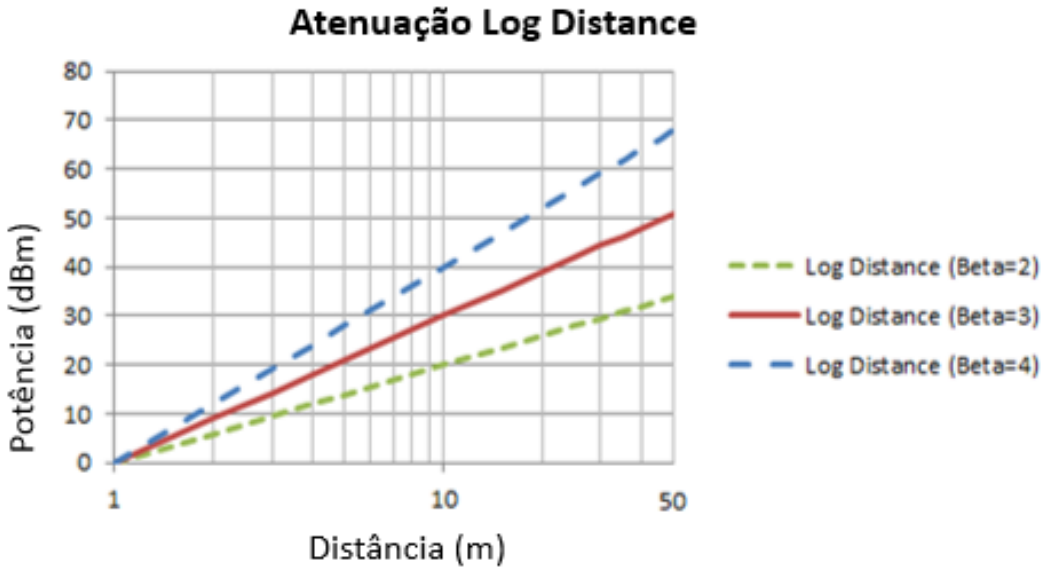
Tabela 1 – Valores de β para diferentes ambientes.

| Ambiente | Path Loss $\beta$ |
|---|---|
| Espaço Livre | 2 |
| Área Urbana | 2.7 a 3.5 |
| *Indoor* em corredores | 1.6 a 1.8 |
| *Indoor* pouco obstruído | 2.2 a 2.7 |
| *Indoor* obstrução média | 2.8 a 3.5 |
| Ambientes abertos semi livres | 3 a 4 |
| *Indoor* com muita obstrução | 4 a 6 |

Fonte: Adaptado de Haykin e Moher (2008).

A Figura 6 apresenta a atenuação baseada no modelo *Log Distance* para 3 valores diferentes de $\beta$ considerando que, para saber a atenuação total do sistema, deve ser acrescentada a atenuação do espaço livre, de acordo com a Equação 2.

Figura 6 – Atenuação *Log Distance* para diferentes β.




Fonte: Elaboração própria.

## 2.6 Distribuições contínuas probabilísticas

Nos itens 2.5.1 e 2.5.2, os cálculos da atenuação do sinal em função do ambiente utilizaram valores constantes como potência, ganho de antena e fator de atenuação. Esses são parâmetros determinísticos e dependem essencialmente do tipo de ambiente, gerando um fator de atenuação constante a cada posição e invariante no tempo.

Entretanto, como pode ser comprovado experimentalmente, existe um grau de incerteza em relação ao sinal recebido. Assim, para completar a caracterização do ambiente, é necessário considerar a variação em função de cada ponto devido às condições do ambiente. Com isso, chega-se na Equação 4 do modelo de *Shadowing*, em que $X_{dB}$ é uma variável aleatória identificada por modelos probabilísticos que determinam as características do ambiente (RAPPAPORT, 2009).

$$P_r(d)[dBm] = P_r(d_0)[dB] - 10\beta \log\left(\frac{d}{d_0}\right) + X_{dB} \quad (4)$$

A utilização de modelos probabilísticos é útil para descrever a propagação não guiada de sinais. Uma variável aleatória contínua pode assumir um número infinito de valores possíveis. Essa característica pode ser usada na modelagem de sinais de rádio.

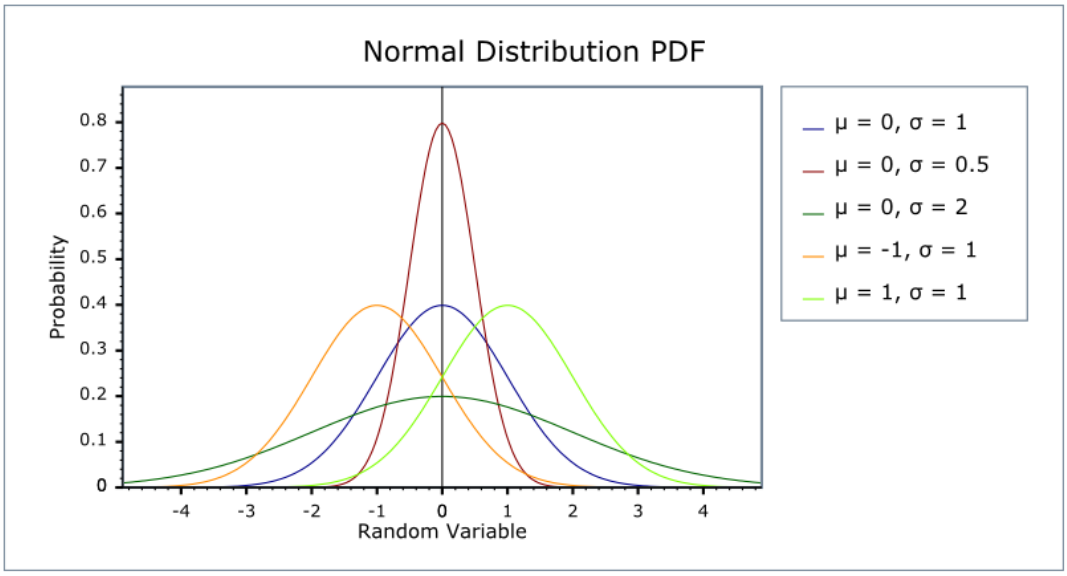
### 2.6.1 Distribuição Normal

Na teoria das probabilidades, a distribuição normal (Gaussiana, de Gauss ou de Laplace – Gauss) é uma distribuição de probabilidade contínua e amplamente utilizada (RAPPAPORT, 2009). Ela é usada para descrever flutuações em torno de um valor médio e é caracterizada pela média $\mu$ e desvio padrão $\sigma$. Sua função de densidade de probabilidade (PDF, *Probability Density Function*) é descrita pela Equação 5.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (5)$$

A Figura 7, ilustra exemplos de distribuição normal, nos quais é possível observar que, quanto maior o desvio padrão ($\sigma$), maior o espalhamento da distribuição em relação a um valor médio ($\mu$).

Figura 7 – Distribuição normal.




Fonte: Boost (2020).

### 2.6.2 Distribuição de Weibull

Essa distribuição é usada em campos aplicados à engenharia – tais como confiabilidade, controle de qualidade, finanças e climatologia – e pode ser usada para modelar os tempos de chegada dos pacotes no tráfego da rede. Ela é descrita na

Equação 6 considerando três parâmetros: a) forma ($\beta$); b) escala ($\eta$), e, c) localização ($\gamma$) (RAPPAPORT, 2009).

$$f(x) = \frac{\beta}{\eta}\left(\frac{x-\gamma}{\eta}\right)^{\beta-1} e^{-\left(\frac{x-\gamma}{\eta}\right)^{\beta}} \quad (6)$$

Onde:

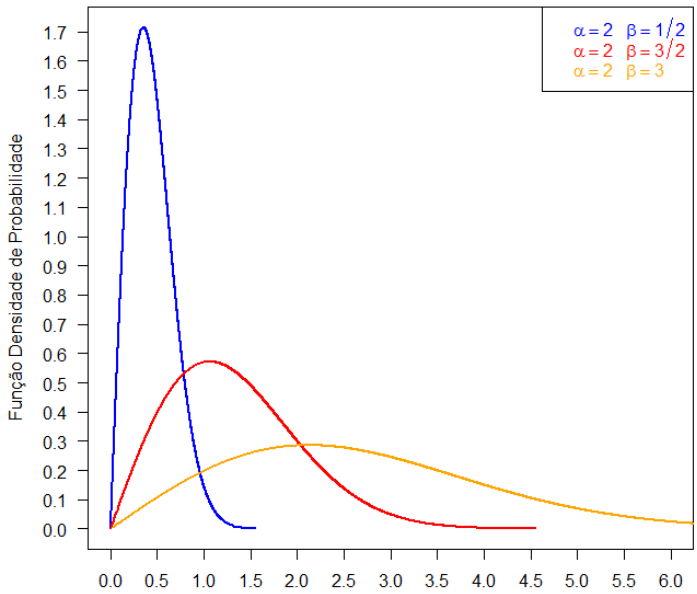
$$f(x) \geq 0, \quad x \geq 0 \text{ ou } \gamma \geq 0, \quad \beta > 0, \quad \eta > 0, \quad -\infty < \gamma < \infty$$

Um valor negativo para o parâmetro de localização "$\gamma$" desloca a distribuição para a esquerda, enquanto um valor positivo desloca para a direita. Se "$\gamma$" for definido como zero, a PDF se tornará uma função de dois parâmetros, definida apenas por variáveis reais não negativas, conforme mostrado na Equação 7.

$$f(x) = \left(\frac{\beta}{\eta}\right)\left(\frac{x}{\eta}\right)^{\beta-1} e^{-\left(\frac{x}{\eta}\right)^{\beta}} \quad (7)$$

O parâmetro de forma $\beta$ também é conhecido como o declive de Weibull. É importante salientar que quando ajustado para valores entre "2" e "4", a distribuição de Weilbull se aproxima da distribuição normal. Em contrapartida, para valores abaixo de "1,25", fornece uma curva inclinada para a direita. E, para valores acima de "10", resulta uma curva inclinada para a esquerda. Para $\beta$ igual a "1", a distribuição de Weibull se reduz a uma distribuição exponencial. A Figura 8 ilustra alguns dos casos acima citados.

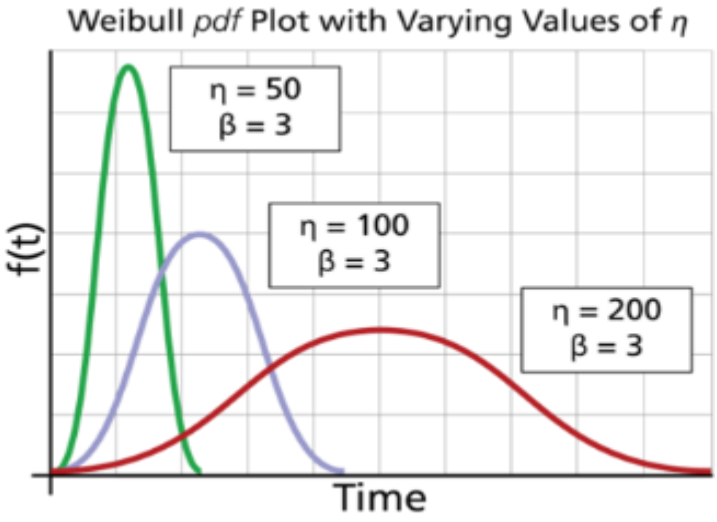
Figura 8- Distribuição de Weibull em diferentes valores de escala.




Fonte: Portal Action (2020).

Se o fator de escala $\eta$ for aumentado, mantendo o fator de forma $\beta$ constante, a distribuição será esticada para a direita enquanto a altura da curva é abaixada, mantendo sua forma e localização. Inversamente, se η é diminuído e β é mantido constante, a distribuição é empurrada para a esquerda e sua altura é aumentada, conforme ilustrado na Figura 9.

Figura 9 – Distribuição de Weibull em diferentes valores de escala.




Fonte: ReliaSoft (2020).

### 2.6.3 Distribuição de Rayleigh

A distribuição de Rayleigh contínua é usada para modelar o intenso desvanecimento rápido e os múltiplos caminhos de sinais densamente dispersos ao alcançar um receptor. Está associada à magnitude da soma de vetores com amplitudes com distribuição normal. Ela é um caso especial da distribuição Weibull com um parâmetro de escala $\eta$ definido como “2” (RAPPAPORT, 2009).

## 2.7 Protocolos utilizados em IoT

De forma objetiva, Matthiesen (2018) descreve:

> O número de protocolos existentes utilizados em RSSF visando à aplicação em IoT cresce a cada dia. Esse tópico trará os principais protocolos (como o Wi-Fi, Bluetooth, Zigbee, 6LoWPAN) e os protocolos emergentes que surgiram nos últimos anos.

### 2.7.1 *Wi-Fi*

Com o crescimento da transmissão sem fio, foi criado em 1997 o padrão IEEE 802.11, que especifica as características da Rede de Área Local Sem Fio (*Wireless Local Area Network*, WLAN). A marca *Wi-Fi* segue esse padrão e tem sido muito utilizada (IEEE COMPUTER SOCIETY, 2016; WIFI ALLIANCE, 2019).

Segundo a Revista IEEE Computer Society (2016, *tradução nossa*), trouxe um breve histórico:

> Com o crescimento da transmissão sem fio, foi criado em 1997 o padrão IEEE 802.11, que especifica as características da Rede de Área Local Sem Fio (*Wireless Local Area Network*, WLAN).

Complementa Akeela e Elziq (2017):

> Esse padrão define as transmissões nas frequências de 2,4GHz e 5GHz, conferindo simplicidade aos usuários na criação de WLAN. Essa facilidade está tornando essas frequências saturadas e suscetíveis à interferência de sinal. Há um novo padrão chamado 802.11ah que utiliza as frequências abaixo de 1GHz, tornando o alcance maior, de até 1 km.

### 2.7.2 *Bluetooth*

Segundo a IEEE Computer Society (2011), outro padrão desenvolvido foi o IEEE 802.15, que especifica as características da Rede de Área Pessoal Sem Fio (*Wireless Personal Area Network*, WPAN). O *Bluetooth* segue o padrão IEEE 802.15.1.

Recentemente, o *Bluetooth Special Interest Group* lançou uma nova especificação conhecida como *Bluetooth 4.0* ou *Bluetooth Low Energy* (BLE), no qual a conectividade dos aparelhos pode ser ponto-a-ponto, *broadcast* (um dispositivo se conecta a vários) e *mesh* (vários dispositivos se conectam a vários dispositivos).

### 2.7.3 Zigbee

O padrão IEEE 802.15.4 especifica as características para WPANs (transmitindo com uma taxa baixa), e, também é conhecido como LR-WPAN (*Low-Rate Wireless Personal Area Network*) (IEEE COMPUTER SOCIETY, 2011). O Zigbee utiliza o padrão 802.15.4 e adiciona mais duas camadas, utilizando-as para encriptação de segurança, roteamento e possível criação de uma rede *mesh* (IEEE COMPUTER SOCIETY, 2005).

Um dos problemas do Zigbee é a baixa taxa de transmissão e a sua cobertura que chega apenas até 75 m, enquanto o Wi-Fi, utilizando a mesma faixa de frequência, consegue alcançar 1 km.

### 2.7.4 6LoWPAN

O protocolo IPv4, utilizado para a comunicação na Internet, atualmente tem limitações quanto à quantidade de endereços disponíveis, que impactam as aplicações para IoT. Por isso, o grupo *Força Tarefa de Engenharia da Internet* (*Internet Engineering Task Force*, IETF) introduziu, em 2006, o IPv6 (DEERING; HINDEN, 1998).

A transmissão de pacotes IPv6, através de LR-WPANs (6LoWPAN), não é direta, posto que a implementação do protocolo IPv6 requer que o *link* possa suportar pacotes com tamanho de até 128 *bytes*, uma vez que os quadros IEEE 802.15.4 podem ter, no máximo, 102 octetos após o cabeçalho MAC. O espaço disponível pode ficar ainda menor caso seja utilizada segurança. Assim, a compressão de cabeçalhos IPv6 é importante para o sucesso em RSSF (PALATTELLA et al., 2013).

Ademais, utilizando-se o 6LoWPAN não há necessidade de gateways, nem da criação de novos protocolos, tornando o sistema mais simples, considerando que todo o sistema é baseado no protocolo IP (MULLIGAN, 2007). De acordo com Hennebert e Santos (2014) o grande problema com o 6LoWPAN está relacionado à

segurança e à privacidade dos dados, já que os mecanismos de segurança utilizados atualmente são muito pesados para serem integrados em pequenos dispositivos IoT.

### 2.7.5 Radiuino

O Radiuino é uma plataforma flexível de código aberto (*hardware* e *software*) para o desenvolvimento de redes de sensores sem fio, que permite controle completo sobre o protocolo de comunicação, incluindo configuração de pacotes (MATTHIESEN, 2018). A plataforma emprega módulos de comunicação homologados pela Agência Nacional de Telecomunicações (ANATEL) e opera com frequência de 915 MHz na faixa de frequências Industrial, Científica e Médica (ISM, do inglês *Industrial Sientific and Medical*). Além disso, o Radiuino usa o ambiente de desenvolvimento integrado (IDE, *Integrated Development Environment*) da plataforma Arduino para desenvolver o *firmware* em linguagem C++.

### 2.7.6 Novos protocolos

Outros protocolos emergentes são Z-Wave, LoRaWAN e SigFox, sendo todos capazes de operar nas faixas de frequência ISM.

O protocolo Z-Wave não é *open source*, contribuindo com dificuldade para o desenvolvimento de aplicações, na justificativa de ser um produto e fazer parte de demais dispositivos de variadas empresas (Z-WAVE ALLIANCE, 2020).

Segundo Schatz (2019), LoRaWAN é mais flexível, permitindo que qualquer interessado se filie à LoRa Alliance e desenvolva dispositivos, porém o seu *hardware* não é barato. Já o SigFox, por sua vez, possui *hardware* de baixo custo, mas vende o seu serviço de conectividade com a Internet, necessitando o usuário pagar uma taxa para utilizar a rede.

## 2.8 Enlace de rádio

Uma rede de sensores sem fio (RSSF) monitora as condições do ambiente que a cerca através de variados sensores e suas principais aplicações são meio-ambiente, saúde, residencial e industrial. Os Nós Sensores são responsáveis por coletar informações do meio e enviar para a ERB. Essa, por sua vez, recebe as informações e está ligada a um *gateway*. O *gateway* conecta a RSSF a internet ou a

um computador, isso pode ser feito através de *Ethernet*, *Wi-Fi*, USB ou porta serial. (AKYILDIZ; VURAN, 2010).

Além dos módulos de comunicação, o bom funcionamento da RSSF também se deve às antenas utilizadas. A antena é um condutor que converte um sinal guiado (comunicação por fio) em um sinal não guiado (transmissão pelo ar). As características de uma antena são determinadas pelo seu diagrama de irradiação (MATTHIESEN, 2018).

## 2.9 Fading

Quando o dispositivo sem fio se desloca no ambiente, seja *indoor* ou *outdoor*, o sinal recebido sofre variações de intensidade denominadas *fading*. Basicamente, existe o *fading* lento e o *fading* rápido superposto (RAPPAPORT, 2009).

O *fading* lento está relacionado à diminuição da intensidade de sinal com base na distância. Em função da variação das alturas do terreno e dos objetos, o sinal flutua em torno de um valor médio determinado por uma reta (com eixo da distância sendo logaritmo) que determina o fator de atenuação β, conforme descrito na seção 2.5.2.

O *fading* rápido está relacionado às variações da fase da portadora, que podem fazer a intensidade do sinal flutuar rapidamente. Esse tipo de *fading* pode ser *flat*, em que todas as componentes de frequência sofrem a mesma atenuação, ou pode ser seletivo, no qual, em função dos múltiplos percursos, existem atenuações diferentes dentro da faixa de frequência do sinal. Esse efeito é desastroso para o sinal de faixa larga, uma vez que destrói a correlação das componentes harmônicas do sinal, provocando sérias distorções.

## 2.10 Definição de Sensibilidade

A sensibilidade de um receptor de rádio é a menor potência de recepção para a qual é atendida a qualidade esperada, que no caso é a BER. Esse valor está intimamente ligado com a Relação Sinal Ruído (SNR, *Signal to Noise Ratio*), posto que, sabendo esse valor e a potência total de ruído, é possível determinar a sensibilidade (RAPPAPORT, 2009).

**PROPOSTA DE UMA BANCADA DE EMULAÇÃO**

É possível encontrar diversos estudos sobre ensino do comportamento do canal de RF como em Tanin (2007), no qual há a preocupação no treinamento de

LPWANs, abordando apenas a configuração de rede, o conjunto de dados e a engenharia de *software*. Também se encontra em Zhou et al. (2004), em que soluções são propostas apenas para roteamento e acesso ao meio através de um modelo de simulação.

Considerando todas estas variantes, propõe-se abordar neste trabalho a emulação do canal de propagação utilizando-se LPWANs. Inicialmente, deve-se destacar o notável espaço que os laboratórios remotos e virtuais vêm obtendo. Assim, Feisel e Rosa (2005, *tradução nossa*) expõe que:

> A incorporação de treinamentos práticos em laboratórios virtuais e / ou remotos serviu, entre outras razões, para refletir sobre quais são os objetivos de aprendizado dos laboratórios na formação de carreiras como a engenharia. Embora hoje ninguém conceba um ensino em qualquer ramo da engenharia sem incluir o treinamento experimental, há muitos anos há pouca atenção para definir, medir e investigar as habilidades que os alunos devem desenvolver em laboratórios.

A Bancada de Emulação de Canal de Rádio, proposta nesta dissertação, visa emular o comportamento de um *link* de comunicação por meio de 02 (dois) nós (não orientados) em diferentes ambientes e condições.

Em suas fases iniciais, um aplicativo de *software* escrito em linguagem *Python,* aplica os modelos de propagação, conforme abordados na seção 2.5, gerando as distribuições estatísticas (conforme apresentado na seção 2.6), que são alimentadas em um atenuador variável, emulando o canal.

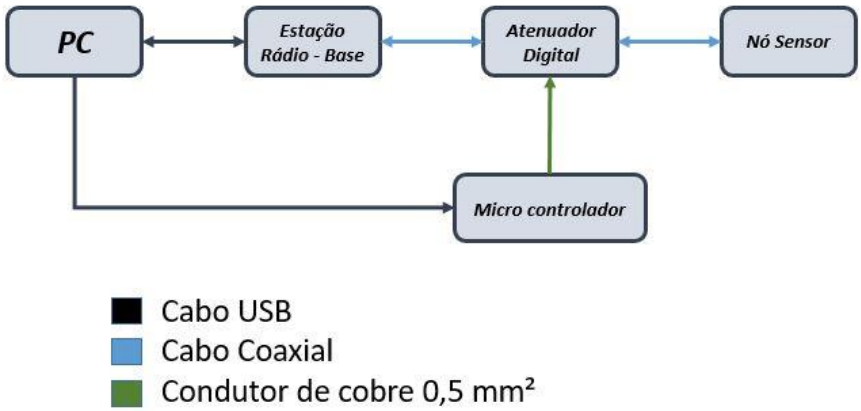
Posto isso, a Bancada de Emulação utiliza-se de uma plataforma *open source* (projeto de *hardware, firmware e software*) para o desenvolvimento de RSSF, baseado na plataforma Radiuino, na qual, emprega os módulos de comunicação *BE900* e *BE990*, ambos homologados pela Agência Nacional de Telecomunicações (ANATEL).

Os módulos carregam um processador ATMega328P e um transceptor de RF *CC1101* com um filtro de largura de banda, operando na faixa de frequências ISM de 915 MHz. O módulo BE990 integra um amplificador de potência (PA, do inglês *Power Amplifier*) e um amplificador de baixo ruído (LNA, do inglês *Low Noise Amplifier*) ao transceptor CC1190 para melhorar o desempenho de recepção e transmissão do sinal de RF.

Para a programação da Estação Rádio Base (ERB) e dos Nós Sensores (NS), usa-se o ambiente de desenvolvimento integrado do Arduino (IDE) para carregar o *firmware* a ser embarcado, escrito em C++.

Um computador pessoal (PC, do inglês *Personal Computer*) é conectado a ERB, que é montada em um adaptador *UartsBee USB* para serial, e ao microcontrolador (através do cabo USB). Sequencialmente, um atenuador variável digital, com um limite de atenuação de 31,5 dB, fica entre a ERB e o Nó Sensor, conectado a ambos via cabos coaxiais, eliminando-se dessa forma as antenas. Um microcontrolador é conectado aos pinos de controle do atenuador, ajustando, assim, o nível de atenuação entre a ERB e os Nós Sensores. A Figura 10 apresenta o diagrama de blocos da bancada de emulação:

Figura 10 – Diagrama de blocos da Bancada de Emulação.




Fonte: Elaboração própria.

O PC gera uma distribuição de probabilidade, segundo os modelos apresentados nas seções 2.5 e 2.6, que é alimentada, valor por valor, ao Microcontrolador, que por sua vez define dinamicamente o nível de atenuação no Atenuador Digital. Em sequência, o computador também cria e envia pacotes de 52 bytes para a ERB. Estes são transmitidos, pacote por pacote, para o NS, também montado em um adaptador *UartsBee USB* para serial. Para cada pacote enviado, o valor de atenuação digital é alterado para simular a propagação em uma variedade de condições.

Por fim, o Nó Sensor recebe os pacotes, mede a intensidade do sinal recebido através da variável RSSI (do inglês *Receive Signal Strength Indicator*) e o retorna à ERB, para que as informações possam ser processadas pelo computador.

# MATERIAL E MÉTODOS

Atualmente, existem diversas tecnologias para a construção de uma rede LPWAN:

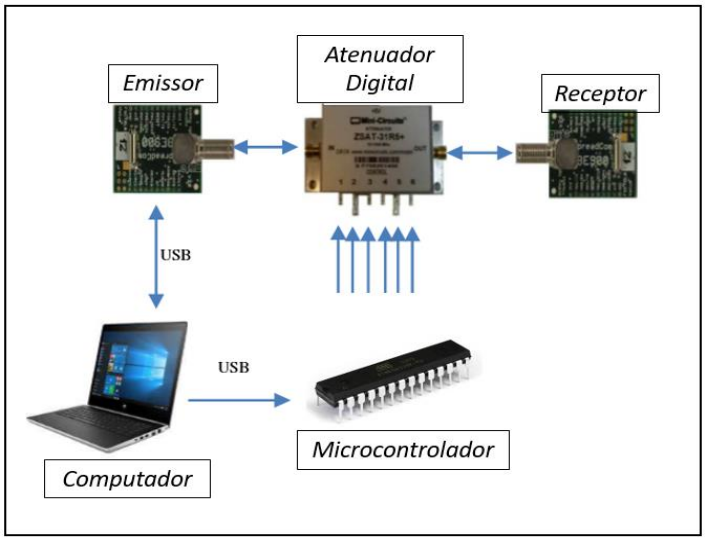
> Com o aumento de aplicações na IoT, surgem uma série de projetos que precisam de tecnologias habilitadoras para redes de longo alcance, baixo consumo de energia e mais baratas. Assim as redes LPWAN (do inglês Low-Power Wide-Area Network), surgem como uma nova alternativa para a conectividade em redes M2M (do Inglês Machine-to-Machine) na IoT (GARCIA; KLEINSCHMIDT, 2017, p. 1009).

Dentre essas tecnologias, a escolhida para desenvolver o presente trabalho foi a “Radiuino”, principalmente por ser uma plataforma aberta criada, especificamente, para o LPWAN, baseada no ambiente de desenvolvimento do Arduino (IDE, do inglês *Integral Development Environment*) (RADIUINO, 2020).

## 4.1 Hardware

Para a emulação do canal de RF com base no diagrama de blocos da Figura 10, a Estação Rádio Base e o Nó Sensor foram interligados através de segmentos de cabos coaxiais e de um Atenuador Digital controlado por um microcontrolador, conforme ilustrado na Figura 11:

Figura 11 - Conexão da bancada de emulação.

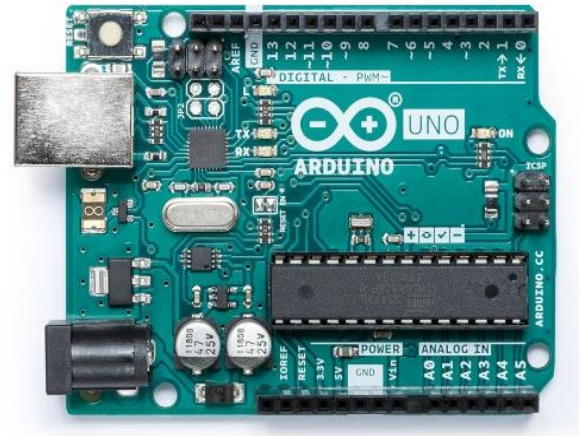



Fonte: Elaboração Própria.

O microcontrolador utilizado no desenvolvimento da bancada de emulação é o Atmega328p, presente na placa de desenvolvimento Arduino Uno. O Arduino Uno, por sua vez, é uma plataforma de prototipagem eletrônica, constituída de *hardware* e *software*, implementada em uma placa única, projetada com um microcontrolador Atmel AVR com suporte de entrada/saída embutido, configurada por uma linguagem de programação padrão, a qual tem origem em *Wiring* e é essencialmente C/C++ (ARDUINO, 2020).

O microcontrolador possui a capacidade computacional para realizar uma série de tratativas das informações enviadas pelo computador. Após isso, é realizada a transmissão dos dados via um meio pré-determinado. Os elementos principais da placa Arduino Uno podem ser observados na Figura 12, em que há seis pinos digitais que são configurados como saídas para o controle do atenuador digital.

Figura 12 - Placa de Desenvolvimento Arduino Uno.




Fonte: Site Oficial Arduino (2019).

### 4.1.1 Rádio utilizado

Antes de realizar o desenvolvimento da RSSF, é necessário se certificar de que todos os equipamentos utilizados estão funcionando de maneira correta. O Nó Sensor e a Estação Rádio Base irão utilizar o módulo de comunicação BE900, mostrado na Figura 13, o qual possui um microcontrolador Atmega328, um transceptor CC1101 integrando um Amplificador de Potência (PA) e um Amplificador de Baixo Ruído (LNA) (RADIUINO, 2020).

Figura 13 - Módulo de comunicação BE990.


Fonte: Site Oficial Radiuino (2020).

#### 4.1.1.1 Verificando a potência de recepção

É necessário verificar a potência de recepção do BE900. Para isso, são utilizados dois módulos de comunicação, um funcionando como Estação Rádio Base e outro como Nó Sensor. Para evitar a saturação do sinal devido à curta distância entre os dois módulos, foi utilizado um atenuador de 70 dB entre os módulos.

A Estação Rádio Base será configurada com o *firmware* de Gerente Local para enviar um pacote apenas requisitando a qualidade do sinal (RSSI), de *downlink* (a potência recebida pelo Nó Sensor) e de *uplink* (a potência recebida pela ERB). Com esse valor, é possível verificar se a potência de recepção está coerente com a potência de transmissão do Nó Sensor.

#### 4.1.1.2 Ajustando o *offset*

Cada um dos módulos BE900 possui um cristal oscilador de 26 kHz para oferecer precisão de tempo necessária para aplicações com sincronismo de tempo, mas esse cristal apresenta um deslocamento no canal da frequência operada devido ao processo de fabricação do componente, possuindo, assim, uma tolerância de ± 20 ppm a ± 30ppm (FARNELL, 2020).

Devido a esse deslocamento, é necessário aferir o módulo para verificar em qual frequência ele está operando e calcular o seu *offset*, que é um valor em hexadecimal que realiza uma compensação no circuito oscilador, para que, dessa forma, o módulo opere na frequência desejada. A diferença entre as frequências (915 Mhz e a frequência medida), é dividida pelo passo da frequência, que é de 1,59 kHz, como pode ser visto na Equação 8:

$$(Offset)_{16} = \frac{(915Mhz - Fmedida)}{1{,}59} \qquad (8)$$

## 4.2 Atenuador digital

O atenuador digital de canais modelo ZSAT-31R5+, fabricado pela *Mini Circuits*, permite selecionar um valor de atenuação entre 0,5 dB e 31,5 dB a partir de uma sequência de seis *bits* aplicados a seus pinos de controle. Para cada *bit* do atenuador, denotados por #1 a #6 na Figura 14, tem-se uma atenuação correspondente de 0,5 a 16 dB, de modo que as 63 combinações dos 6 *bits* permitem configurar uma atenuação de 0,5 a 31,5 dB em passos de 0,5 dB. A combinação “000000” equivale a uma atenuação de 0 dB, mas que pode alcançar, segundo o fabricante, uma perda de inserção máxima de 7 dB.

Figura 14 - Especificação dos bits de atenuação.

**Digital Step Attenuator Electrical Specifications**

| MODEL NO. | FREQUENCY (MHz) | | PRIMARY ATTENUATION STEPS (dB) @ CONTROL PORT | | | | | | ATTENUATION (dB) LOGIC STATE | | VSWR (:1) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $f_L$ | $f_U$ | #1 | #2 | #3 | #4 | #5 | #6 | (1,1,1,1,1,1)** Nom. | (0,0,0,0,0,0) Max. | L | M | U |
| ZSAT-31R5 | 10 | 1000 | 0.5±0.18 | 1±0.25 | 2±0.25 | 4±0.3 | 8±0.4 | 16±0.5 | 31.5 | 7.0 | 2.0 | 1.5 | 1.6 |

L=10 to 100 MHz M=100 to 500 MHz U=500 to 1000 MHz

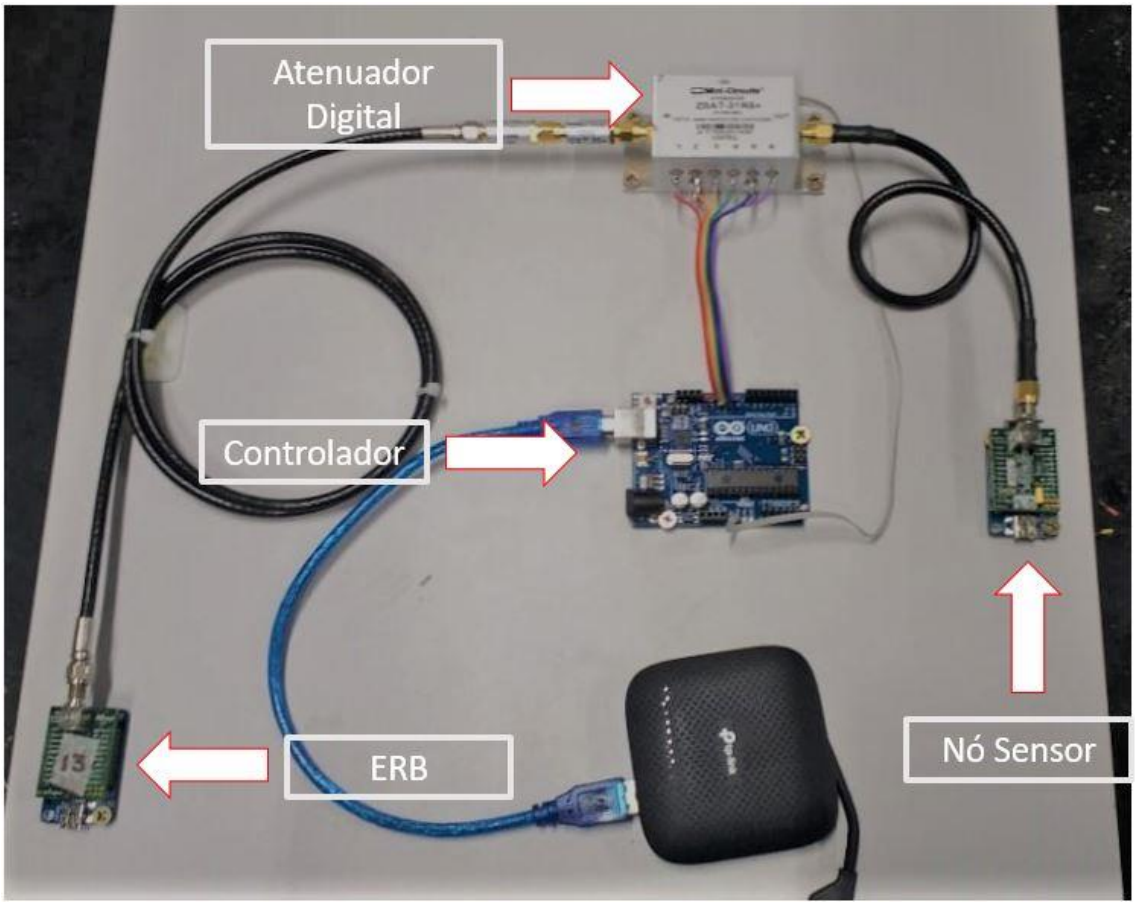
Fonte: Adaptado de *MINI CIRCUITS* (2020).

## RESULTADOS

O protótipo da Bancada de Emulação é apresentado na Figura 15. A dimensão do protótipo e a disposição dos componentes foram selecionados estrategicamente para atender às necessidades de um *kit* para uso em aplicações de pesquisa e treinamento, facilitando a visualização das conexões e interligações do *link* de rádio.

Na configuração final do protótipo, a bancada foi dimensionada em uma chapa de MDF (sendo as dimensões da placa de 750x750 mm), enquanto os cabos coaxiais (entre a ERB e o atenuador digital e entre o atenuador digital e o NS) ficaram com comprimentos de 32 cm e 17 cm, respectivamente.

Figura 15 - Vista superior da bancada de emulação.




Fonte: Elaboração própria.

A Bancada de Emulação foi projetada para facilitar o processo de aprendizado, logo, possui uma interface fácil de interagir, a qual pode gerar rapidamente os gráficos que ajudam na comparação de conjuntos de dados,

auxiliando na compreensão dos fenômenos de propagação do sinal de rádio.

As etapas para instalar e configurar a bancada estão listadas a seguir:

**1) *Setup* dos componentes:** a conexão física entre os componentes deve ser feita como mostrado na Figura 15. Os cabos coaxiais possuem medidas padronizadas devido ao cálculo das perdas sistêmicas, sendo de 32 cm a conexão da ERB com o atenuador digital, e de 17 cm a conexão entre o antenuador digital e o NS.

**2) Módulo de Comunicação e Microcontrolador:** os módulos precisam ser configurados para que a comunicação por RF possa ser estabelecida. Configurações como endereço de nó, canal de comunicação, potência de transmissão, frequência base e deslocamento, modulação, taxa de dados e taxa serial (*baudrate*) devem ser configuradas. Além disso, o microcontrolador precisa ser configurado com o *firmware* apropriado.

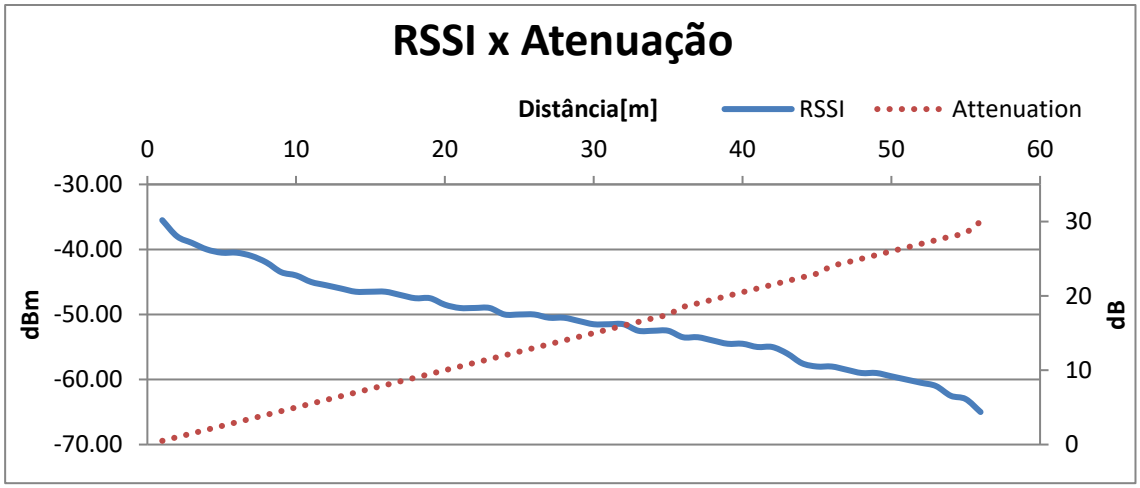
**3) Atenuação:** a combinação de atenuação fixa e variável, e as perdas do sistema compõem a atenuação final. Ela não deve estar abaixo de 25 dBm, pois, isso acarretaria saturação do receptor, e, também não pode exceder 90 dBm, porque isso levaria a grandes perdas de pacotes. É importante lembrar que a bancada apresenta algumas perdas sistêmicas, como a atenuação fixa dos conectores e dos cabos coaxiais, que precisam ser consideradas e medidas. Fazendo a conexão direta entre a ERB e o NS, eliminando-se o Atenuador Digital, mediu-se 15,5 dB de perdas sistêmicas. Foi definido que a RSSI média de *downlink* dos dados a serem emulados deve determinar a quantidade de atenuação (fixa e variável) a ser configurada na bancada.

**4) Configuração da bancada:** uma vez configurada a bancada, o programa deve ser iniciado. A interface mostrada na Figura 21 (comentada a seguir) precisa ser preenchida com os dados da rede a ser emulada. Além disso, as portas seriais da ERB e do microcontrolador devem ser selecionadas. À medida que a emulação avança, a interface apresenta o número do pacote atual e a RSSI de *uplink*. No final do processo, o *software* gera os gráficos e cria um arquivo em formato Excel com os dados e com as estatísticas da emulação.

Após avaliar a Bancada de Emulação por meio de uma sequência de testes, é possível observar, na Figura 16, a relação entre a atenuação e a intensidade do sinal emulado, representando a atenuação no Espaço Livre previamente calculado e fazendo uso da Equação 1.

Enquanto a atenuação variou de 0 a 31,5 dB, a RSSI variou de -35 a cerca de -67dBm. Para a atenuação máxima de 31,5 dB, a RSSI alcançou -67 dB, que é, aproximadamente, o valor inicial de -35 dB, somando-se a atenuação de 31,5 dB.

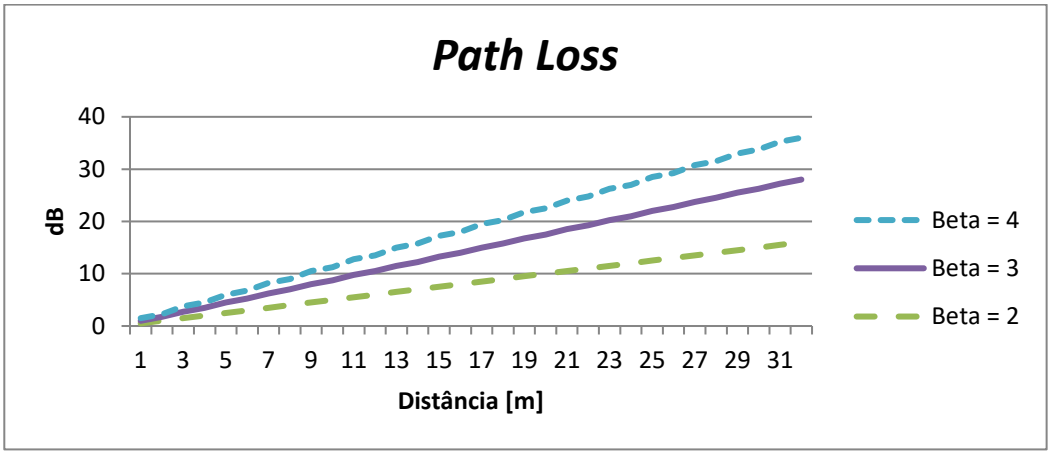
Figura 16 - RSSI medida no Nó Sensor x Atenuação inserida.




Fonte: Elaboração própria.

A Figura 17 apresenta a emulação da atenuação de Espaço Livre para diferentes valores de β, de acordo com o modelo *Log Distance* da Equação 2. Comparando com a Figura 6, é possível observar que os valores de atenuação medidos se assemelham aos valores calculados.

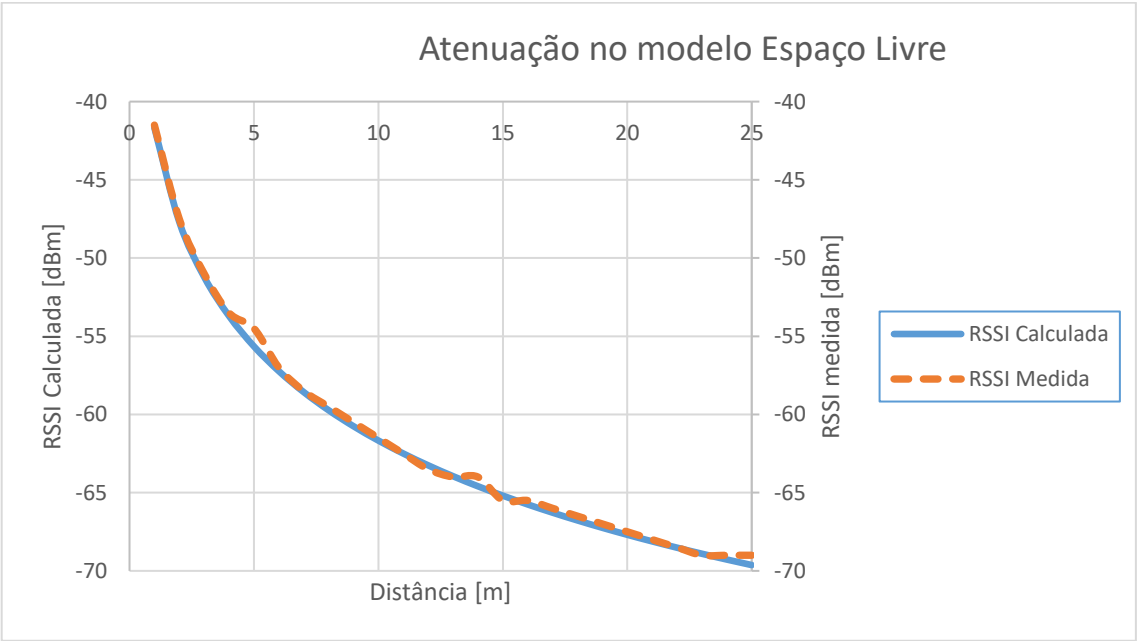
Figura 17 -Resultados Emulados pelo modelo Log Distance




Fonte: Elaboração própria.

Na Figura 18, é feita uma comparação entre a RSSI calculada e a RSSI medida no modelo do Espaço Livre para uma frequência de 915MHz, uma potência de transmissão de -37,5 dBm e uma variação de distância do Nó Sensor à ERB de 0 a 25m. Fazendo-se uso da Equação 1, o *software* calcula o valor de atenuação a partir da distância, que possui uma variação fixa (de metro em metro), inserindo, assim, o valor de atenuação correspondente no canal através do Atenuador Digital. É possível observar como o valor emulado (medido) aproxima-se do calculado.

Figura 18 - Comparação entre o valor emulado e o valor calculado

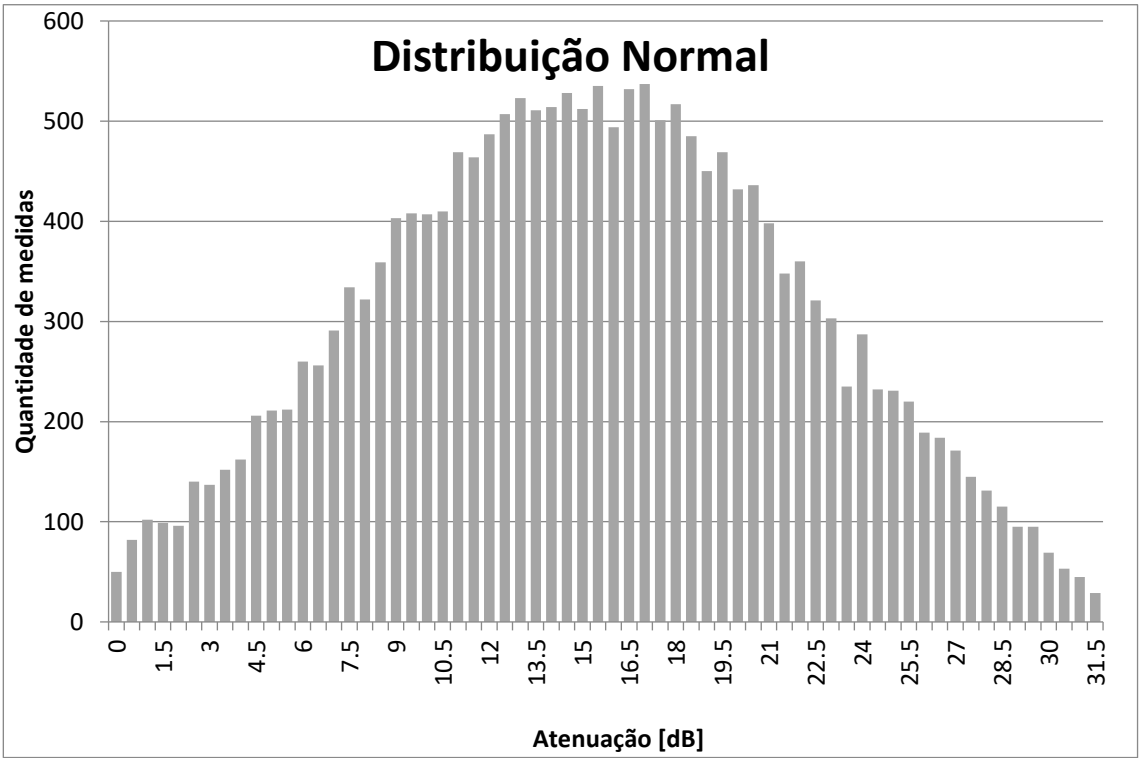



Fonte: Elaboração própria.

Finalmente, na Figura 19, são apresentadas as atenuações obtidas na emulação do modelo de propagação *Log Distance*, com parâmetro $\beta = 3$, acrescido da variável aleatória $X_{dB}$ relativa à distribuição de probabilidade Normal (Equação 4), para uma média (μ) de 16 dB e 20 mil medições.

É possível observar na Figura 18 que os valores de atenuação gerados pelo *software* e inseridos na rede através do atenuador digital tendem a uma distribuição aproximadamente normal, como apresentado na Figura 7.

Figura 19 – Emulação da atenuação Normal.




Fonte: Elaboração própria.

Uma série de testes foi realizada para comparar o comportamento do sinal obtido com a Bancada de Emulação, tendo como referência RSSFs em aplicações reais. As RSSFs de teste foram configuradas para comunicação ponto a ponto, compreendendo em uma ERB e um NS e empregando os mesmos módulos de comunicação.

A Bancada de Emulação foi, então, configurada com os mesmos parâmetros: potência de transmissão de 0 dBm, técnica de modulação 2FSK, taxa de transmissão de 9600 bps, taxa de dados de 38,38 kbps e espaçamento de canal de 125kHz.

No primeiro teste (Teste 1), os nós da rede foram colocados ao ar livre, como mostrado na Figura 19, e separados por 165 metros com uma linha de visada parcial entre eles. A ERB foi montada no telhado de uma edificação, conforme mostrado na Figura 20 (Lado “a”) e conectada a uma antena setorial de 12,5 dBi de ganho, com largura de feixe de 42 graus, inclinada para apontar para o NS. Já o NS foi montado em um parque (Figura 20, Lado “b”) e conectado a uma antena setorial de 7 dBi de ganho.

O Teste 1 com a configuração descrita acima operou por 48 horas de forma ininterrupta, de modo a coletar dados sobre o comportamento do sinal em dois dias da semana inteiros, com um total de 98000 pacotes trocados, em dias de céu limpo.

Para o Teste 2, a mesma configuração foi mantida. Dessa vez, no entanto, o NS foi colocado numa posição mais distante no parque: a 310 metros. Essa rede de teste funcionou por 24 horas ininterruptas, com um total de 48000 pacotes trocados, novamente em dias de céu limpo.

Figura 20 - Configuração da RSSF de teste: ERB e Nó Sensor.

(a) (b)

Fonte: Próprio Autor.

Para o Teste 3, a ERB foi conectada a uma antena omnidirecional de 2,15 dBi de ganho e colocada em ambiente interno. Já o NS, conectado à antena setorial de 7 dBi de ganho, foi colocado ao ar livre, a 25,7 metros de distância, separado da ERB

por duas paredes de alvenaria. Essa rede de teste funcionou por 24 horas, totalizando 48000 pacotes. É possível observar o posicionamento dos NS em relação a ERB na Figura 21.

Figura 21 - Posicionamento dos NS em relação a ERB.


Fonte: Próprio Autor.

A análise dos dados das redes de teste mostrou que o comportamento do sinal em condições reais de aplicação tendeu à natureza de uma distribuição normal quanto às medidas de RSSI de recepção. Para determinar o tamanho da amostra (n) usado nas emulações, foi utilizado o método *Taro Yamane* (YAMANE, 1970).

Para garantir uma margem de erro (*e*) de 5%, os tamanhos de amostra para cada emulação foram calculados usando a Equação 08. Onde "*n*" representa o tamanho da amostra e "*N*" é o número total de amostras de cada rede de teste:

$$n = \frac{N}{1+Ne^2} \qquad (8)$$

Portanto, para emular a rede do Teste 1, o número mínimo de amostras deve ser de 399, enquanto que nas redes dos Testes 2 e 3 deve haver um mínimo de 397 amostras.

A Bancada de Emulação foi, então, configurada, por sua vez, com o número apropriado de amostras obtidas das coletas de dados de cada rede de teste, de modo a emular o mesmo comportamento de sinal, considerando uma distribuição normal.

Assim, a Tabela 2 mostra que os resultados obtidos com a Bancada de Emulação foram muito próximos aos das redes de testes em termos de intensidade do sinal, média e desvio padrão, atestando a qualidade da emulação oferecida pelo protótipo desenvolvido neste trabalho.

Tabela 2 – Comparação dos resultados da Bancada de Emulação com casos de teste de RSSFs reais.

| | **Número de medições** | **RSSI Mínima [dBm]** | **RSSI Máxima [dBm]** | **Média RSSI [dBm]** | **Desvio Padrão** |
|---|---|---|---|---|---|
| **Teste 1 *Outdoor*** | 98000 | -67.00 | -59.50 | -62.78 | 1.13 |
| **Bancada de Emulação** | 399 | -64.50 | -60.50 | -62.21 | 0.62 |
| **Teste 2 *Outdoor*** | 48000 | -83.00 | -73.50 | -78.05 | 1.24 |
| **Bancada de Emulação** | 397 | -84.5 | -77.50 | -80.88 | 1.20 |
| **Teste 3 *Indoor*** | 48000 | -70.00 | -58.50 | -63.47 | 0.62 |
| **Bancada de Emulação** | 397 | -65.00 | -61.00 | -63.01 | 0.38 |

Fonte: Próprio Autor.

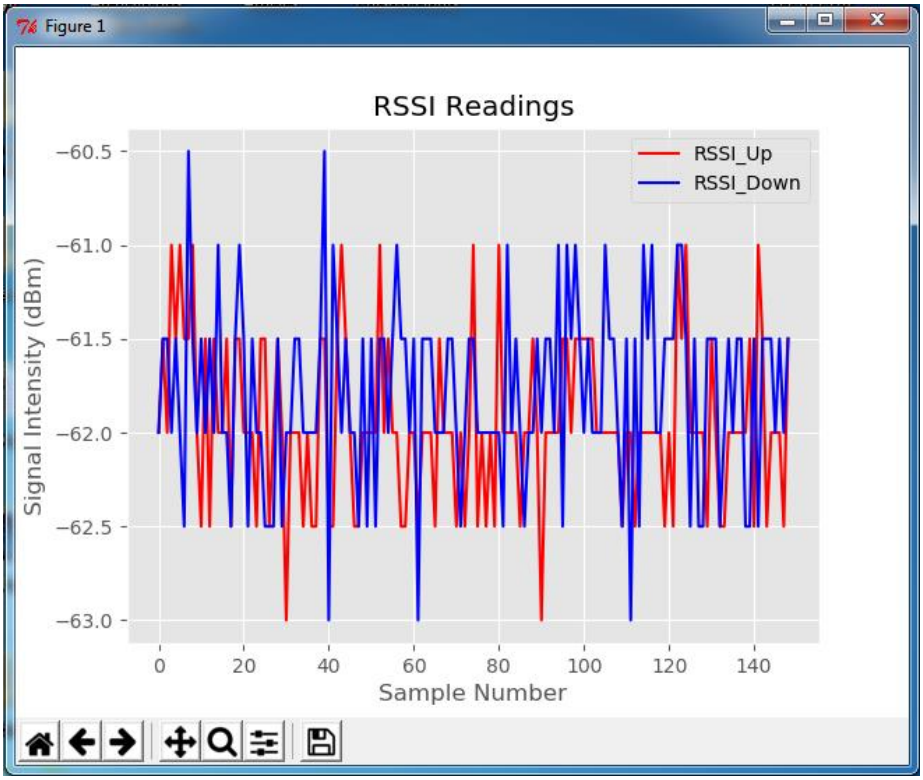
Uma interface amigável para a Bancada de Emulação foi desenvolvida eliminando a tela padrão *Phyton* e, assim, dando flexibilidade à configuração, como mostra a Figura 22.

Figura 22 – Interface com usuário para Bancada de Emulação.


Fonte: Próprio Autor.

A interface também pode gerar um gráfico de linhas relacionando a densidade de pacotes em relação à intensidade de sinal (RSSI) em *Uplink* e *Downlink*, conforme mostrado na Figura 23.

Figura 23 – RSSI de *Uplink* e *Downlink* para uma sequência de pacotes.

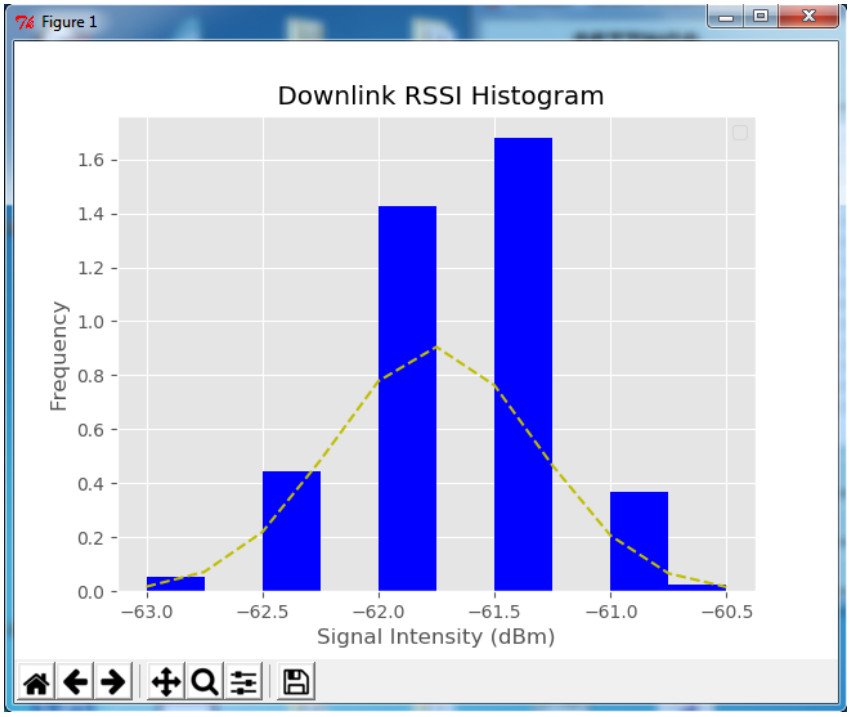



Fonte: Próprio Autor.

É possível observar, na Figura 23, que os valores medidos permaneceram em torno de um valor médio (-62 dBm), alcançando um valor máximo de -60,5 dBm e um valor mínimo de -63 dBm.

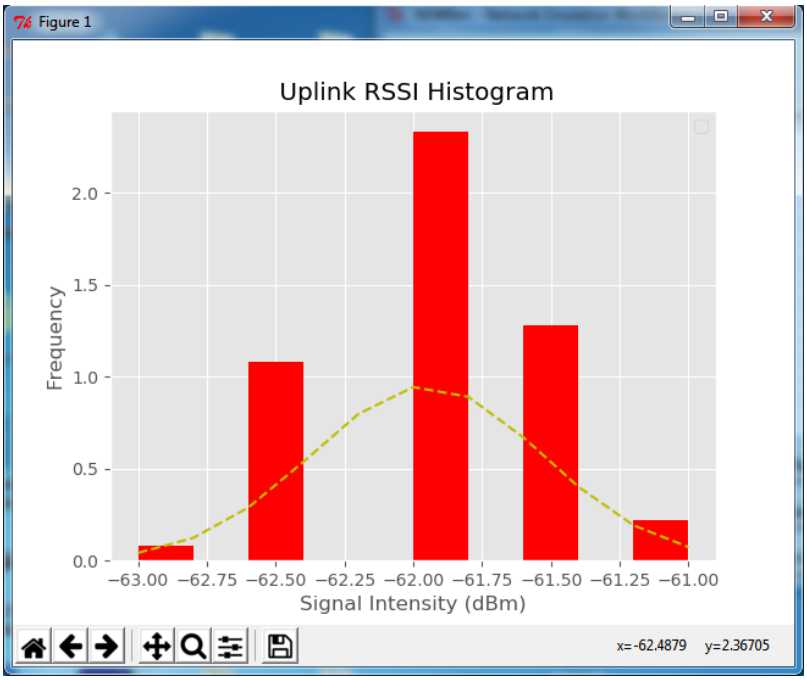
Nas Figuras 24 e 25, são apresentados os histogramas das medidas de RSSI de *Downlink* e *Uplink* comparados aos gráficos de uma distribuição normal, calculada usando os dados coletados pelo experimento.

Figura 24 – Comparação RSSI de *Downlink* com uma distribuição normal 01.




Fonte: Próprio Autor.

Figura 25 – Comparação RSSI de *Uplink* com uma distribuição normal 02.




Fonte: Próprio Autor.

## CONCLUSÃO

Este trabalho apresenta o desenvolvimento de uma Bancada de Emulação que pode ser utilizada para testar tecnologias de LPWAN e na demonstração de como essas tecnologias se comportam em um ambiente instável e não repetitivo, como a comunicação de RF, usando os modelos de Espaço Livre e *Log Distance* e as distribuições probabilísticas Normal, Weibull e Rayleigh como exemplo.

Os resultados obtidos com a Bancada de Emulação desenvolvida, usando a tecnologia Radiuino LPWAN, demonstram que é possível emular o canal de comunicação rádio entre um Nó Sensor em uma Estação Rádio Base, interconectados através de cabos coaxiais e um Atenuador Digital, para imprimir ao sinal RF transmitido as características de propagação de um ambiente real.

Avaliar e analisar conceitos de propagação de um canal de RF torna-se muito mais eficaz quando um sistema emulado usa componentes reais de IoT. Ademais, a Bancada de Emulação desenvolvida permite testar e comparar outras tecnologias para cenários pré-determinados, visando facilitar a definição da tecnologia a ser utilizada, já que as tecnologias de comunicação para aplicações em IoT devem ser personalizadas por aplicação.

Vale ressaltar que a grande vantagem deste tipo de emulação é o controle sobre a atenuação aplicada ao canal, que pode ser como a proposta neste trabalho (para o modelo de Espaço Livre de propagação e *Log Distance*), incluindo qualquer outro tipo de distribuição probabilística.

Em trabalhos futuros, poderá ser construída uma biblioteca que conterá resultados de algumas redes de referência que operam em diferentes ambientes, com variações de condições climáticas, distâncias e modulações. Estes dados da biblioteca fornecerão exemplos de parâmetros a serem utilizados no modelamento de aplicações reais, além disso a variação desses parâmetros e seus efeitos no sinal auxiliará na apreciação dos fenômenos de propagação do sinal.

Por fim, em uma perspectiva futura, a Bancada de Emulação permitirá, ainda, uma versão *online*, que poderá ser acessada e controlada remotamente atendendo ao conceito de *weblabs.*

## REFERÊNCIAS BIBLIOGRÁFICAS


AKEELA, R.; ELZIQ, Y. Desing and verification of IEEE 802.11ah for IoT and M2M applications IoT and M2M. In: IEEE INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMUNICATIONS WORKSHOPS, 15 ed, 2017. Kona, Hawaii. **Anais...** Estados Unidos: IEEE PerCom, 2017.

AKYILDIZ, I.; WEILIAN, S.; SANKARASUBRAMANIAM, Y.; CAYIRCI, E. A survey on sensor networks. In: Communications Magazine, **IEEE,** vol.40, no.8, pp. 102-114, 2002.

AKYILDIZ, I. F..; VURAN, M. C. **Wireless Sensor Networks.** 1ª edição. Nova Jersey: Wiley, 2010.

ARDUINO**. Arduino Uno Rev3**. 2020. Disponível em: <www.arduino.cc>. Acesso em: 20 fev. 2020.

BACCOUR, N.; KOUBAA, C.; MOTTOLA, S. L.; ZUNIGA. M. A.; YOUSSEF, H. BOANO, C. A.; ALVES, M. Radio link quality estimation in wireless sensor networks: A survey. **ACM Transactions on Sensor Networks (TOSN),** v. 8, n. 4, p. 34, 2012.

BENTO, J. **Segurança em Redes de Sensores Wireless.** 2009. 102f. Dissertação (mestrado) – Programa de Pós-Graduação em Engenharia Electrotécnica e de Computadores Major Telecomunicações, Faculdade de Engenharia da Universidade do Porto, Portugal, 2009.

BOOST. **Normal (Gaussian) Distribution**. Disponível em: <https://www.boost.org/doc/libs/1_47_0/libs/math/doc/sf_and_dist/html/math_toolkit/dist/dist_ref/dists/normal_dist.html>. Acesso em: 15 fev. 2020.

BRANQUINHO, O. **Tecnologias de Redes sem Fio**. 1. ed. Rio de Janeiro, RJ: Rede Nacional de Ensino e Pesquisa, 2014.

BRASIL. **Agência Nacional de Telecomunicações.** Glossário de termos da Anatel: Canal de Radiofrequência (RF). Disponível em: <https://www.anatel.gov.br/legislacao/component/fsf/?view=faq&catid=3&faqid=162>. Acesso em: 06 fev. 2020.

CAVALCANTI, F. R.; MACIEL, T. F.; FREITAS JUNIOR, W. C.; SILVAS, Y. C. B. **Comunicação móvel celular**. 1 ed. Rio de Janeiro: Elsevier, 2018.

DEERING, S.; HINDEN, R. Internet Protocol, Version 6 (IPv6) Specification, 1998. Disponível em: <https://tools.ietf.org/html/rfc2460>. Acesso em: 21 fev. 2020.

DÉO, A. L. B. **Proposta de um modelo de referência Open Source para IoT:** Foco em PME. 2018. 149f. Dissertação (mestrado) – Programa de Pós-Graduação em Gestão de Redes de Telecomunicações, Pontifícia Universidade Católica de Campinas, Campinas, 2018.

FARNELL. SMD Microprocessor crystal. Disponível em: <http://www.farnell.com/datasheets/1883657.pdf>. Acesso em: 21 fev. 2020.

FEISEL, Lyle D.; ROSA, Albert J. “The role of the laboratory in undergraduate engineering education”, **Journal of Engineering Education,** v. 94, n.1, 2005.
2020.

FREITAS, R. N. **Impacto no desempenho de RSSF com diferentes técnicas de cifragem**. 2018. 124f. Dissertação (mestrado) – Pós-Graduação em Gestão de Redes de Telecomunicações, Pontifícia Universidade Católica de Campinas, Campinas, 2018.

GARCIA, P. S,; KLEINSCHMIDT, J. H. Tecnologias Emergentes de Conectividade na IoT: Estudo de Redes LPWAN. In: SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS. 35 ed. 2017. São Pedro/SP. **Anais…** São Paulo: SBrT, 2017. p. 1009-1013.

HAYKIN S.; MOHER, M. **Modern Communication Systems**. Porto Alegre, Brasil: Bookman, 2008.

HENNEBERT, C., SANTOS, J. D. Security Protocols and Privacy Issues into 6LoWPAN Stack: a synthesis. IEEE Internet of Things Journal, v. 1, n. 5, 2014.

IEEE COMPUTER SOCIETY**. IEEE Standard for Information technology-- Local and metropolitan area networks-- Specific requirements-- Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPAN)IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)** IEEE, 2005. Disponível em: <http://ieeexplore.ieee.org/document/1490827/>.

IEEE COMPUTER SOCIETY. **IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)** IEEE, 2011.

IEEE COMPUTER SOCIETY. **IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) SpecificationsIEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)** IEEE, 2016.

ILYAS, M; MAHGOUB, I. **Handbook of Sensor Networks:** Compact Wireless and Wired Sensing Systems. 1 ed. CRC Press, 2014.

LEE, W. C. Y. **Mobile Communications Design Fundamentals**, 2nd edition. Nova Jersey: John Wiley and Sons, 1993.

LIMA, R. A.; FONTANA, E.; MARTINS-FILHO, J. F.; PRATA, T. L.; CAVALCANTI, G. O.; LIMA, R. B.; OLIVEIRA, S. C.; CAVALCANTI, F. J. M. M. Satellite telemetry system for pollution detection on insulator strings of high-voltage transmission lines. **SBMO/IEEE MTT-S International,** 2009.

**Lora Alliance**. Disponível em: <https://lora-alliance.org>. Acesso em: 20 fev. 2020.

MATTHIESEN, E. A. N. 2018. **Proposta de escopo de engenharia de IoT: estudos de casos**. 2018. pp. 80-97. Dissertação (mestrado) – Pontifícia Universidade Católica de Campinas, Centro de Ciências Exatas, Ambientais e de Tecnologias, Programa de Pós-Graduação em Infraestrutura Urbana. Campinas – SP.

MINI CIRCUITS. **Digital Step Attenuator ZSAT-31R5+.** [s.d.]. Disponível em: < https://www.minicircuits.com/pdfs/ZSAT-31R5+.pdf>. Acesso em: 18 fev 2020.

MULLIGAN, G. The 6LoWPAN architecture. In: WORKSHOP ON EMBEDDED NETWORKED SENSORS. 4 ed. 2007. New York/USA. **Anais…** New York: EmNets ’07, 2007.

MUNDO Educação. **Umidade do ar.** Disponível em <https://mundoeducacao.bol.uol.com.br/geografia/umidade-ar.htm>. Acesso em: 01 fev. 2020.

PALATTELA, M. R. et al. **Standardized protocol stack for the internet of (important) things**. IEEE Communications Surveys and Tutorials, v. 15, n. 3, p. 1389–1406, 2013.

PANDA, M. Security in Wireless Sensor Networks using Cryptografic Techniques. **American Journal of Engineering Research (AJER),** n. 01, p. 50-56, 2014.

PORTAL Action. **Modelos probabilísticos contínuos: 6.13 - Distribuição Weibull**. Disponível em: <http://www.portalaction.com.br/probabilidades/613-distribuicao-weibull>. Acesso em: 20 fev. 2020.

**Radiuino.** Disponível em: <radiuino.cc>. Acesso em: 13 fev. 2020.

RAPPAPORT, T. S. **Comunicações Sem Fio: Princípios e Prática**, 2 ed. Pearson, 2009.

RAZA, U.; KULKARNI, P. and SOORIYABANDARA, M. **Low Power Wide Area Networks: An Overview**, IEEE Commun. Surv. Tutorials, vol. 19, no. 2, pp. 855–873, 2017.

RELIASOFT CORPORATION. **Life Data Analysis Reference**. Disponível em: <http://www.synthesisplatform.net/references/Life_Data_Analysis_Reference.pdf>. Acesso em: 20 fev. 2020.

RUFINO, N. M. de O. **Segurança em Redes Sem Fio.** 4º ed. Brasil: [s.n.], 2014.

SCHATZ, G. **SigFox Vs. LoRa:** A Comparison Between Technologies and Business Models. Disponível em: <https://www.link-labs.com/blog/sigfox-vs-lora>. Acesso em: 20 fev. 2020.

**Sigfox**. Disponível em: <https://www.sigfox.com/en>. Acesso em: 18 fev. 2020.

TANIN, E. **Teaching Wireless Sensor Networks at the University of Melbourne**, IEEE Distrib. Syst. Online, vol. 8, no. 6, pp. 3–3, jun. 2007.

WI-FI ALLIANCE. Wi-Fi CERTIFIED ™ n: **Longer-Range, Faster-Throughput, Multimedia-Grade Wi-Fi ® Networks Executive Summary**, 2009. Disponível em: <https://www.wifi.org/download.php?file=/sites/default/files/private/wp_20091006_802_11n_Industry_20120319_0.pdf>. Acesso em: 25 fev. 2020.

YAMANE, TARO. **Statistic: An Introductory Analysis.** Tokyo: Harper International Edition, 1970.

ZHOU, G.; HE, KRISHNAMURTHY, S. and STANKOVIC,A. **Impact of radio irregularity on wireless sensor networks**, in Proceedings of the 2nd international conference on Mobile systems, applications, and services - MobiSYS '04, p. 125, 2004.

Z-WAVE ALLIANCE. **About Z-Wave Technology - Z-Wave Alliance**. Disponível em: <https://z-wavealliance.org/about_z-wave_technology/>. Acesso em: 20 fev. 2020.

## ANEXO A – Firmware Controlador da Bancada de Emulação

```
int attPins[] = {8, 9, 10, 11, 12, 13};
int attControl[]={0,0,0,0,0,0};

void setup() {
  // initialize the serial communication:
  Serial.begin(9600);
  // initialize the ledPin as an output:
  for (int i=0; i<6; i++) {
        pinMode(attPins[i], OUTPUT);
    }
}

void loop() {
  byte atenuacao;

  // check if data has been sent from the computer:
  if (Serial.available()) {
    // read the most recent byte (which will be from 0 to 255):
    atenuacao = Serial.read();
    // set the attenuator control vector:
    for (int i=0; i<6; i++) {
      if(1 << i & atenuacao){
        attControl[i]=1;
      }
      else{
        attControl[i]=0;
      }
    }
    // turn the pins high or low according to the attenuator control
vector:
    for (int i=0; i<6; i++) {
      digitalWrite(attPins[i],attControl[i]);
    }

  Serial.write(atenuacao);
  }
}
```

## ANEXO B – Software de controle da bancada

```
import schedule
import time
from Tkinter import *
import tkMessageBox
from ScrolledText import ScrolledText
import threading
import ttk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg,
NavigationToolbar2TkAgg
from matplotlib.figure import Figure
from matplotlib import style
from datetime import datetime
import serial
import math
import struct
from time import localtime, strftime
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats
from scipy.stats import weibull_min
from scipy.stats import norm
from matplotlib import mlab

style.use("ggplot")

start_time = strftime("%d_%m_%Y @ %H-%M-%S")

global ser, arduino

PotMind = 0
PotMaxd = 0

# classe que herda os metodos para tratar threads:parar e iniciar
class myThread(threading.Thread):
    def __init__(self, function):
        self.running = False
        self.function = function
        super(myThread, self).__init__()

    def start(self):
        self.running = True
        super(myThread, self).start()

    def run(self):
        while self.running:
```

```
            self.function()

    def stop(self):
        self.running = False

#funcao que gera os graficos de RSSI usando a matplotlib
def grafico_d():
        fig = plt.figure()
        fig.subplots_adjust( hspace=0.65 )
        ax1 = fig.add_subplot(111)
        x = []
        y = []
        try:
            dados = open("Gauss_Screen_Data.csv",'r')

        except Exception as erro:
            tkMessageBox.showinfo(message=erro)
        for line in dados:
            line=line.strip()
            print (line)
            X, Y = line.split(';')
            x.append(float(X))
            y.append(float(Y))
        dados.close()

        ax1.plot(y, "r-", label='RSSI_Up')
        ax1.plot(x,"b-", label='RSSI_Down')
        ax1.set_title('RSSI Readings')
        ax1.set_xlabel('Sample Number')
        ax1.set_ylabel('Signal Intensity (dBm)')
        ax1.legend()
        plt.show()

#funcao que gera o downlink Histogram and Normal usando a matplotlib
def grafico_e():
        fig = plt.figure()
        fig.subplots_adjust( hspace=0.65 )
        ax1 = fig.add_subplot(111)
        x = []
        y = []
        try:
            dados = open("Gauss_Screen_Data.csv",'r')

        except Exception as erro:
            tkMessageBox.showinfo(message=erro)
        for line in dados:
            line=line.strip()
            print (line)
            X, Y = line.split(';')
```

```
            x.append(float(X))
            y.append(float(Y))
        dados.close()

        (mu,sigma) = norm.fit(x)

        n,bins,patches =ax1.hist(x,10, normed=1,facecolor='blue')
        a=mlab.normpdf(bins,mu,sigma)

        ax1.plot(bins,a,'y--')
        ax1.set_title(' Downlink RSSI Histogram')
        ax1.set_xlabel('Signal Intensity (dBm)')
        ax1.set_ylabel('Frequency')
        ax1.legend()
        plt.show()

#funcao que gera o Uplink Histogram and Normal usando a matplotlib
def grafico_f():
        fig = plt.figure()
        fig.subplots_adjust( hspace=0.65 )
        ax1 = fig.add_subplot(111)
        x = []
        y = []
        try:
            dados = open("Gauss_Screen_Data.csv",'r')

        except Exception as erro:
            tkMessageBox.showinfo(message=erro)
        for line in dados:
            line=line.strip()
            print (line)
            X, Y = line.split(';')
            x.append(float(X))
            y.append(float(Y))
        dados.close()

        (mu,sigma) = norm.fit(y)

        n,bins,patches =ax1.hist(y,10, normed=1,facecolor='red')
        a=mlab.normpdf(bins,mu,sigma)

        ax1.plot(bins,a,'y--')
        ax1.set_title(' Uplink RSSI Histogram')
        ax1.set_xlabel('Signal Intensity (dBm)')
        ax1.set_ylabel('Frequency')
        ax1.legend()
        plt.show()

def start_rede():
```

```
     schedule.every(3).seconds.do(medir)
     while True: #loop infinito
        schedule.run_pending() #realiza as tarefas agendadas
        time.sleep(1)

def inicia(ope):
    global op
    op=ope
    v1 = myThread(function = start_rede)
    if(ope==1):
        v1.start()
    elif(ope==0):
        schedule.clear()
        v1.stop()
        botao3.config(state="normal")

def verificar_portas():
    portas_ativas = []
    for n in range(100):
        try:
            ver=serial.Serial("COM"+str(n))
            portas_ativas.append(ver.portstr)
            ver.close()
        except serial.SerialException:
            pass
    return portas_ativas

def callback():
    if tkMessageBox.askokcancel("Sure you want to leave?"):
        inicia(0)
        raiz.destroy()

def enviar():
    try:
        global ser

        print 'Test Sent to Sink Node'

        Pacote = {}
        ser = serial.Serial(p.get(), 9600, timeout=0.5) # seta valores da
serial

        time.sleep(1.5)

        ID_sensor = 1
        ID_base = 0

        # Cria Pacote de 52 bytes com valor zero em todas as posi□es
        for i in range(0,52): # faz um array com 52 bytes
```

```
            Pacote[i] = 0

        # Limpa o buffer da serial
        ser.flushInput()

        # Coloca no pacote o ID_sensor e ID_base
        Pacote[8] = ID_sensor #origem

        Pacote[10] = ID_base #destino
        Pacote[49] = 1 #1 indica pacote de apenas requisicao

        # TX pacote - envia pacote para a base transmitir
        for i in range(0,52):
            TXbyte = chr(Pacote[i]) # Deve converter para caracter em ASCII
para escrever na serial
            ser.write(TXbyte)

        # Tempo de espera para que receba a resposta do sensor
        time.sleep(0.5)

        # RX pacote - recebe o pacote enviado pelo sensor
        line = ser.read(52) # faz a leitura de 52 bytes do buffer que recebe
da serial pela COM

        # Checa se recebeu 52 bytes
        if len(line) == 52:
            print 'OK'
            tkMessageBox.showinfo(message="Port "+ p.get()+ " Connected")

        else:
            print 'ERROR'
            tkMessageBox.showinfo(message="Port "+ p.get() + " NOT Connected")
            time.sleep(0.5)
            ser.flushInput()

    except Exception as error:
        tkMessageBox.showinfo(message=error)

def enviar1():
    try:
        global arduino
        print 'Test Sent to Micro-Controller'

        Pacote1 = {}

        arduino = serial.Serial(p1.get(), 9600, timeout=0.5) # seta valores da
serial

        time.sleep(1.5)
```

```
        # Cria Pacote de 52 bytes com valor zero em todas as posi□es
        for i in range(0,1): # faz um array com 1 byte
           Pacote1[i] = 0

        # Limpa o buffer da serial
        arduino.flushInput()

        for i in range(0,1):
           TXbyte = chr(Pacote1[i]) # Deve converter para caracter em ASCII
para escrever na serial
           arduino.write(TXbyte)

        # Tempo de espera para que receba a resposta do arduino
        time.sleep(0.5)

        # RX pacote - recebe o pacote enviado pelo sensor
        line = arduino.read(52) # faz a leitura de 52 bytes do buffer que
recebe da serial pela COM

        # Checa se recebeu 52 bytes
        if len(line) == 1:
           print 'OK'
           tkMessageBox.showinfo(message="Port "+ p1.get()+ " Connected")

        else:
           print 'ERROR'
           tkMessageBox.showinfo(message="Port "+ p1.get() + " NOT Connected")
           time.sleep(0.5)
           arduino.flushInput()

    except Exception as erro:
         tkMessageBox.showinfo(message=erro)

def medir():
    global ser, arduino, media_rssi

    conteudo.config(state="normal")

    ID_base = 0
    w = int(data_medidas.get())
    media_rssi = 0

    # Cria o vetor Pacote
    Pacote = {}

    #Cria o vetor para salvar os valores das potências
    listaPotDesviod ={}
    listaPotDesviou ={}
```

```
# Cria Pacote de 52 bytes com valor zero em todas as posições
for i in range(0,52): # faz um array com 52 bytes
   Pacote[i] = 0

while True:
   try:
      #Inicializacao das variaveis
      contador_erro_consec = 0
      contador_tot = 0
      contador_pot = 0
      potmediad = 0.0
      potacumulad = 0.0
      potmeddbd = 0.0
      contador_err = 0
      potmediau = 0.0
      potacumulau = 0.0
      potmeddbu = 0.0
      PER = 0

      AcumDPd = 0
      AcumDPu = 0
      AcumVad = 0
      AcumVau = 0
      MedDPd = 0
      MedDPu = 0
      DPd = 0
      DPu = 0

      PotMaxd = -200
      PotMind = 10

      PotMaxu = -200
      PotMinu = 10

      #Gera a distribui□o
      i = 0 # array de x2
      u = float(data_param1.get())# mu = mean
      o = float(data_param2.get())# sigma = standard deviation
      #x = float(data_param3.get())#TBD
      s = np.random.normal(loc=u, scale=o, size=w)
      x2 = np.array(s, 'int')

      Opcao = "1"

      # Limpa o buffer da serial
      ser.flushInput()

      # Coloca no pacote o ID_base
```

```
            Pacote[10] = int(ID_base)
            Pacote[37] = 1 #Liga o LDR

            if Opcao == "1":
               ID_sensor = 1  # Sensor a ser acessado
               Pacote[8] = int(ID_sensor) # Coloca no pacote o ID_sensor

               Atenuador = 0

               primeira_rodada = 1

               filename1 = strftime("Gauss_Screen_Stats @_%d_%m_%Y_%H-%M-
%S.csv")

               print "Start Emulation"
               S = open(filename1, 'w')

               while i < w :
               # Rotina de controle da Atenuacao
                      Atenuador = x2[i]
                      i=i+1
                      config_atenuador = 1
                      primeira_rodada = 0

                      while config_atenuador:
                          arduino.flushInput()
                          if Atenuador >= 64:
                              comando = 63
                          elif Atenuador <= 0:
                             comando = 0
                          else:
                              comando = Atenuador
                          arduino.write(chr(comando))
                          time.sleep(0.5)
                          atenuador_rx = arduino.read(1)
                          if ord(atenuador_rx) == comando:
                              config_atenuador = 0

                      for k in range(0,52): # transmite pacote
                         TXbyte = chr(Pacote[k])
                         ser.write(TXbyte)

                      # Aguarda a resposta do sensor
                      time.sleep(0.5)

                      contador_tot = contador_tot + 1
```

```
                    line = ser.read(52) # faz a leitura de 52 bytes do buffer
que recebe da serial pela COM
                    if len(line) == 52:

                        rssid = ord(line[0]) # RSSI_DownLink
                        rssiu = ord(line[2]) # RSSI_UpLink

                        #RSSI Downlink
                        if rssid > 128:
                            RSSId=((rssid-256)/2.0)-74

                        else:
                            RSSId=(rssid/2.0)-74

                        #RSSI Uplink
                        if rssiu > 128:
                            RSSIu=((rssiu-256)/2.0)-74

                        else:
                            RSSIu=(rssiu/2.0)-74

                        num_pacote = ord(line[12])
                        rssi_t = str(RSSId)
                        rssi_w = (1* pow(10,-3))*pow(10,(RSSId/10.0))
                        media_rssi = media_rssi + rssi_w

                        conteudo.insert(END, str(num_pacote)+  " RSSId:
"+str(RSSId)+ "dBm" + " RSSIu: "+str(RSSIu)+ "dBm"'\n')

                        conteudo.see("end")

                        count = ord(line[12]) # contador de pacotes enviados
pelo sensor


                        s1=("Gauss_Screen_Data.csv")

                        S1 = open(s1, 'a')
                        S1.write(str(RSSId)+';'+str(RSSIu)+'\n')

                        if RSSId > PotMaxd:
                            PotMaxd = RSSId

                        if RSSId < PotMind:
                            PotMind = RSSId

                        if RSSIu > PotMaxu:
                            PotMaxu = RSSIu

                        if RSSIu < PotMinu:
```

```
                        PotMinu = RSSIu

                    listaPotDesviod[contador_pot]= RSSId   #Grava a
potencia de downlink para calculo do desvio padrao
                    listaPotDesviou[contador_pot]= RSSIu   #Grava a
potencia de uplink para calculo do desvio padrao

                    contador_pot=contador_pot+1 #incrementa o contador
utilizado para a media de potencia e para o desvio padrao

                    potmwd = pow(10,(RSSId/10))   #converte a potencia de
downlink em dBm para mW.
                    potacumulad = potacumulad + potmwd  #Soma a potencia em
mW em um acumulador

                    potmwu = pow(10,(RSSIu/10))   #converte a potencia de
uplink em dBm para mW
                    potacumulau= potacumulau + potmwu #Soma a potencia em
mW em um acumulador

                else:
                     conteudo.insert(END,"Packet Loss "'\n')
                     conteudo.see("end")
                     ser.flushInput()
                     time.sleep(1)

                     """contador_erro_consec = contador_erro_consec + 1
#counts the number of consecutive errors
                     contador_err = contador_err + 1
                     ser.flushInput()
                     time.sleep(1)

                if contador_erro_consec >= 3:
                     ser.close() # closes and resets the serial
                     ser = serial.Serial(p.get(), 9600, timeout=0.5) # seta
valores da serial
                     contador_erro_consec = 0
                     time.sleep(1)"""

            for l in range(0,contador_pot):
                AcumVad =AcumVad+ listaPotDesviod[l]   #acumula o valor da
lista para calcular a media
                AcumVau =AcumVau+ listaPotDesviou[l]   #acumula o valor da
lista para calcular a media

            MedDPd = float (AcumVad)/float(contador_pot)
            MedDPu = float (AcumVau)/float(contador_pot)

            for m in range(0,contador_pot):
```

```
                AcumDPd =AcumDPd+ pow((listaPotDesviod[m]- MedDPd),2)
#acumula o valor da variancia
                AcumDPu =AcumDPu+ pow((listaPotDesviou[m]- MedDPu),2)
#acumula o valor da variancia

            DPd = float (AcumDPd)/float(contador_pot)   #termina o calculo da
variancia
            DPu = float (AcumDPu)/float(contador_pot)   #termina o calculo da
variancia

            potmediad = potacumulad /contador_pot
            potmeddbd = 10*math.log10(potmediad)

            print >>S
            print >>S,time.asctime(),';Maximum Downlink Power:;', "%.2f"
%PotMaxd,' ;dBm;'
            print >>S,time.asctime(),';Minimum Downlink Power:;', "%.2f"
%PotMind,' ;dBm;'
            print >>S,time.asctime(),';Mean Downlink Power:;', "%.2f"
%potmeddbd,' ;dBm;'
            print >>S,time.asctime(),';Downlink Standard Deviation:;', "%.2f"
%DPd,' ;dB;'

            potmediau = potacumulau /contador_pot
            potmeddbu = 10*math.log10(potmediau)

            print >>S
            print >>S,time.asctime(),';Maximum Uplink Power:;', "%.2f"
%PotMaxu,' ;dBm;'
            print >>S,time.asctime(),';Minimum Uplink Power:;', "%.2f"
%PotMinu,' ;dBm;'
            print >>S,time.asctime(),';Mean Uplink Power:;', "%.2f"
%potmeddbu,' ;dBm;'
            print >>S,time.asctime(),';Uplink Standard Deviation:;', "%.2f"
%DPu,' ;dB;'

            PER = (float(contador_err)/float(contador_tot))* 100
            print >>S
            print >>S,time.asctime(),';PER:;', "%.2f" %PER,';%;'
            print
            finish_time = strftime("%d_%m_%Y @ %H-%M-%S")
            print "Start Time: %s"% start_time,',', "End Time: %s"   %
finish_time
            S.close()
            ser.close() # fecha a porta COM
            arduino.close()
            break

        except KeyboardInterrupt:
```

```
            S.close()
            ser.close()
            arduino.close()

            break
        conteudo.see("end")

def nova():
        jan=Toplevel()
        fig = Figure(figsize=(6, 4), facecolor='white')
        b=Button(jan, text='Save Graphic', command=lambda:salvar(fig))
        b.grid(row = 0, column=0)
        canvas = FigureCanvasTkAgg(fig, master=jan)
        canvas.get_tk_widget().grid(row=3, column=1, columnspan=3,
sticky='NSEW')
        grafico(fig,canvas)

def salvar(ff):
        ftypes = [('.png (PNG)', '*.png')]
        f = asksaveasfilename(filetypes=ftypes, defaultextension=".png")

raiz=Tk() #cria a tela principal, usando um objeto TKinter
raiz.configure(background='powder blue')
raiz.title(" NEWBen - Network Emulation WorkBench ") #funcao para alterar
titulo da janela

label_dados=Label(master=raiz, font=("Arial", 14, "bold"),text =
"SETTINGS",padx=5,pady=5)
label_dados.configure(background='powder blue')

label_param=Label(master=raiz,font=("Arial", 14, "bold"), text = "
RESULTS",padx=5,pady=5)
label_param.configure(background='powder blue')

label_dados.grid(row=0,column=0,columnspan=4)
label_param.grid(row=0,column=32,columnspan=4)

p = StringVar()
portas = ttk.Combobox(width=16,textvariable = p)
portas.grid(row=1,column=1)
portas['values']= verificar_portas()

p1 = StringVar()
portas1 = ttk.Combobox(width=16,textvariable = p1)
portas1.grid(row=2,column=1)
portas1['values']= verificar_portas()

btn_teste=Button(raiz,text="Sink Node Serial",command=enviar)
btn_teste.configure(font=("Arial", 10, "bold"),background='powder blue')
```

```
btn_teste.grid(row=1,column=0)

btn_teste1=Button(raiz,text="Processor Serial",command=enviar1)
btn_teste1.configure(font=("Arial", 10, "bold"),background='powder blue')
btn_teste1.grid(row=2,column=0)

data_param1=Entry(raiz)
data_param1.grid(row=4,column=1)
data_param1.config(state="normal")

label_a=Label(master=raiz,text = " Level of Attenuation")
label_a.configure(background='powder blue')
label_a.grid(row=4,column=0)

data_param2=Entry(raiz)
data_param2.grid(row=5,column=1)
data_param2.config(state="normal")

label_a1=Label(master=raiz,text = "Standard Deviation")
label_a1.configure(background='powder blue')
label_a1.grid(row=5,column=0)

data_medidas=Entry(raiz)
data_medidas.grid(row=6,column=1)
data_medidas.config(state="normal")

label_med=Label(master=raiz,text = "Number of Readings")
label_med.configure(background='powder blue')
label_med.grid(row=6,column=0)

data_param3=Entry(raiz)
data_param3.grid(row=7,column=1)
data_param3.config(state="normal")

label_a2=Label(master=raiz,text = "Transmission power")
label_a2.configure(background='powder blue')
label_a2.grid(row=7,column=0)

"""data_param4=Entry(raiz)
data_param4.grid(row=8,column=1)
data_param4.config(state="normal")

label_a3=Label(master=raiz,text = "Minimum RSSI Value")
label_a3.configure(background='powder blue')
label_a3.grid(row=8,column=0)

data_param5=Entry(raiz)
data_param5.grid(row=9,column=1)
data_param5.config(state="normal")
```

```
label_a4=Label(master=raiz,text = "Maximun RSSI Value")
label_a4.configure(background='powder blue')
label_a4.grid(row=9,column=0)"""

botao3=Button(raiz,text="Start Readings",command= lambda:inicia(1))
botao3.configure(font=("Arial", 10, "bold"),background='powder blue')
botao3.grid(row=1,column=34,padx=5,pady=5,sticky='NSEW')

conteudo = ScrolledText(raiz,width=35,height=15,padx=5,pady=5)
conteudo.config(state="disabled")
conteudo.grid(row=2,column=34,rowspan=8,padx=5,pady=5,sticky='NSEW')

botao4=Button(raiz,text="Plot the RSSIs Graphics",command=grafico_d)
botao4.configure(font=("Arial", 10, "bold"),background='powder blue')
botao4.grid(row=25,column=34,padx=5,pady=5,sticky='NSEW')
botao4.config(state="normal")

botao5=Button(raiz,text="Plot the Downlink Histogram",command=grafico_e)
botao5.configure(font=("Arial", 10, "bold"),background='powder blue')
botao5.grid(row=31,column=34,padx=5,pady=5,sticky='NSEW')
botao5.config(state="normal")

botao6=Button(raiz,text="Plot the Uplink Histogram",command=grafico_f)
botao6.configure(font=("Arial", 10, "bold"),background='powder blue')
botao6.grid(row=28,column=34,padx=5,pady=5,sticky='NSEW')
botao6.config(state="normal")

raiz.protocol("WM_DELETE_WINDOW", callback)
raiz.mainloop()
```