UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ - CAMPUS CURITIBA
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ENGENHARIA ELETRÔNICA

Estudos e Experimentos com Redes LPWAN utilizando LoRaWAN

Relatório de Pesquisa

Edital 02/2019 - PROPPG - Programa Institucional de Iniciação Científica da UTFPR

Autores: Afonso Henrique Kingeski
Marcos Eduardo Pivaro Monteiro (Coorientador)
João Luiz Rebelatto (Orientador)

Curitiba, 9 de outubro de 2020

Resumo

A difusão da Internet das Coisas fez surgirem inovações na área de telecomunicação, entre elas as LPWAN(do inglês *Low Power Wide Area Networks*), tecnologias que visam transmitir dados a dezenas de quilômetros com ínfimo custo econômico e energético. Dentre as tecnologias LPWAN, se destaca o LoRa(do inglês *Long Range*), uma técnica de modulação CSS(do inglês *Chirp Spread Spectrum*) de propriedade da Semtech Corporation, cujo protocolo de comunicação mais divulgado é o LoRaWAN. Nesta pesquisa foi feita a análise e documentação dos métodos necessários para a criação de uma rede LoRaWAN, além da configuração e incorporação de um *End Node* na rede. Todo o processo auxiliou na consolidação dos conceitos abordados e gerou documentos que facilitam a reprodução dos procedimentos para novos desenvolvimentos.

Palavras-Chave: LoRa, LoRaWAN, LPWAN.

Sumário

1 INTRODUÇÃO

A difusão da Internet das Coisas (*IoT*, do inglês *Internet of Things*) possibilitou a disseminação de novas tecnologias de telecomunicação e transmissão de dados. A busca por dispositivos que cobrissem uma grande área geográfica a um baixo custo de energia fez com que surgisse um novo ramo da *IoT*, as LPWAN(do inglês Low Power Wide Area Networks) - uma vez que as tecnologias convencionais, como o *Bluetooth* ou *ZigBee*, cobrem uma área reduzida [1].

As tecnologias LPWAN objetivam cobertura de áreas na escala de dezenas de quilômetros a um baixíssimo custo econômico e de consumo de bateria [2]. Entretanto, tais tecnologias enfrentam vários desafios de implementação e disseminação, que vão desde dificuldades em padronização a trade-offs para manter o baixo custo.

A cobertura de grandes áreas deve-se aos mecanismos de modulação de sinal utilizadas: *narrowband* e espalhamento espectral (do inglês,*spread specthrum techniques*). As técnicas de *narowband* consistem em codificar o sinal em larguras de banda bastante baixas - cerca de 25 kHz -, diminuindo o custo de energia e nível de ruído[3]. Algumas LPWAN trabalham ainda com *ultra-narrowband* (UNB) - por volta dos 100 Hz - reduzindo ainda mais o ruído ao preço de reduzir a eficácia de *End Nodes* individuais e aumentando o tempo de recebimento de sinal.

As técnicas de espalhamento espectral dispersam um sinal de baixa largura de banda em uma frequência maior, mantendo a densidade de energia. Desta forma, o sinal é mais resistente a interferências e dificulta ser detectado por terceiros. Porém, há um aumento significativo de processamento necessário para decodificar o sinal no receptor.

Uma das formas com que as LPWAN reduzem seu custo total é diminuindo o custo de manutenção, aumentando significativamente a vida útil de suas baterias[4]. Para isto, é preciso definir as melhores opções de protocolos e disposição os dispositivos em na rede, as topologias. Na topologia de malha, um *End Node*, ou ponto de acesso, escolhe o melhor caminho para um sinal ir de um ponto A para um ponto B através de múltiplos saltos. Este tipo de topologia pode ser custoso para grandes áreas, tanto do ponto de vista logístico quanto do ponto de vista energético, pois alguns *End Nodes* podem ser sobrecarregados e as informações congestionadas. Assim, a maioria das LPWAN optaram pela topologia estrela, onde um *End Node* conversa diretamente com uma estação base, economizando tempo de processamento e energia. Consequentemente, a bateria de um único *End Node*, pode durar mais de 10 anos[3]. A criação destas estações base também permitiu que estas se apropriassem de processamentos mais complexos, facilitando o trabalho dos *End Nodes*. Algumas LPWAN suportam topologia de malha ou em árvore, mas aumenta-se a complexidade dos protocolos de transmissão[3].

Não só o custo energético das LPWAN é baixo, mas também seus componentes para implementação[5]. Como o hardware dos transmissores é muito menos complexo

e poucos *gateways* podem cobrir cidades inteiras, um projeto LPWAN pode custar somente algumas dezenas de dólares. Além disso, várias bandas não-licenciadas podem ser usadas livremente ou usam-se bandas de operadoras de redes móveis para evitar o custo de licenciamento[3].

O escopo de aplicações das LPWAN é abrangente, mas limitado. O uso da tecnologia para propósitos que necessitem uma maior taxa de transmissão ainda é inviável. Portanto, pesquisas na área de novas técnicas de modulação seriam bem-vindas. Por fim, os progressos obtidos até agora pelas LPWAN têm se mostrado promissores. Estas tecnologias têm expandido e complementado outras já existentes, disseminando ainda mais o conceito de *IoT*.

1.1 Motivação

Uma das tecnologias LPWAN que se destacam, pelo seu vasto uso em *IoT* e em trabalhos acadêmicos, é LoRa(do inglês *Long Range*), uma técnica de modulação CSS(do inglês *Chirp Spread Spectrum*) de propriedade da Semtech Corporation [1]. Ela possui propriedades únicas que justificam seu amplo uso: atinge uma grande área de cobertura; baixo custo dos componentes; grande durabilidade de bateria; capacidade de recepção simultânea nos *gateways* e resistência ao efeito Doppler [6].

A LoRa é a primeira implementação CSS de baixo custo para uso comercial[7]. A troca de potência ou cobertura por taxa de transmissão em uma mesma largura de banda se deve a três parâmetros base: a largura de banda (BW, do inglês *Bandwidth*), o fator de espalhamento e um indicador CR(do inglês, *Code Rate*). As larguras de banda geralmente utilizadas são as de 125 kHz, 250 kHz ou 500 kHz, valores diretamente proporcionais a taxa efetiva de dados transmitidos. O fator de espalhamento pode variar entre 7 e 12 e, aumentando seu valor, tem-se uma melhor confiabilidade no sinal transmitido - em caso de precariedade do canal, por exemplo - mas a taxa de transmissão cai significativamente. Variando de 0 a 4, o CR é um parâmetro que determina a taxa do código corretor de erro[8].

Embora existam progressos na comercialização e padronização da tecnologia, um obstáculo a ser superado é a escalabilidade da rede, uma vez que probabilidade de cobertura cai exponencialmente com o aumento do número de dispositivos na rede [9]. A escalabilidade do LoRa não só é afetada pela interferência de transmissões que utilizam o mesmo fator de espalhamento, mas também por interferência em canais com fator de espalhamento diferente, visto que eles não são completamente ortogonais [10]. Tentativas de aprimoramento incluem o uso de múltiplas antenas e replicação de transmissões [11], e novos métodos de replicação codificada [12].

Uma questão importante é a baixa taxa de dados por segundo ofertada, o que limita aplicações que necessitam de um grande tráfego de dados ou baixa latência [13].Para amenizar o problema, um dispositivo na rede poderia utilizar seus vizinhos

para impulsionar uma transmissão mais volumosa, já que geralmente os dispositivos *IoT* estão ociosos[13].

Outra abordagem é a utilização de comunicação direta entre os dispositivos da rede. Além de permitir uma maior taxa de dados e oferecer uma menor latência, o uso de comunicação direta entre os dispositivos diminui a carga da rede é reduzida [14]. Os revezes da comunicação direta é que informações da transmissão não são armazenadas pela rede, possíveis interferências internas e problemas de segurança e privacidade[14], embora já existam iniciativas para o último[15].

Neste contexto, esta pesquisa possibilita uma melhor compreensão das etapas de comunicação na rede LoRaWAN e, salientando a comunicação dos dispositivos com a rede, é possível utilizar a mesma biblioteca empregada na criação do *firmware* customizado para implementar a comunicação direta entre dois *End Nodes*.

1.2 Objetivos

1.2.1 Objetivo Geral

Análise e documentação dos procedimentos necessários para a criação de um rede LoRaWAN, bem com a inserção de um *End Node* na rede.

1.2.2 Objetivos Específicos

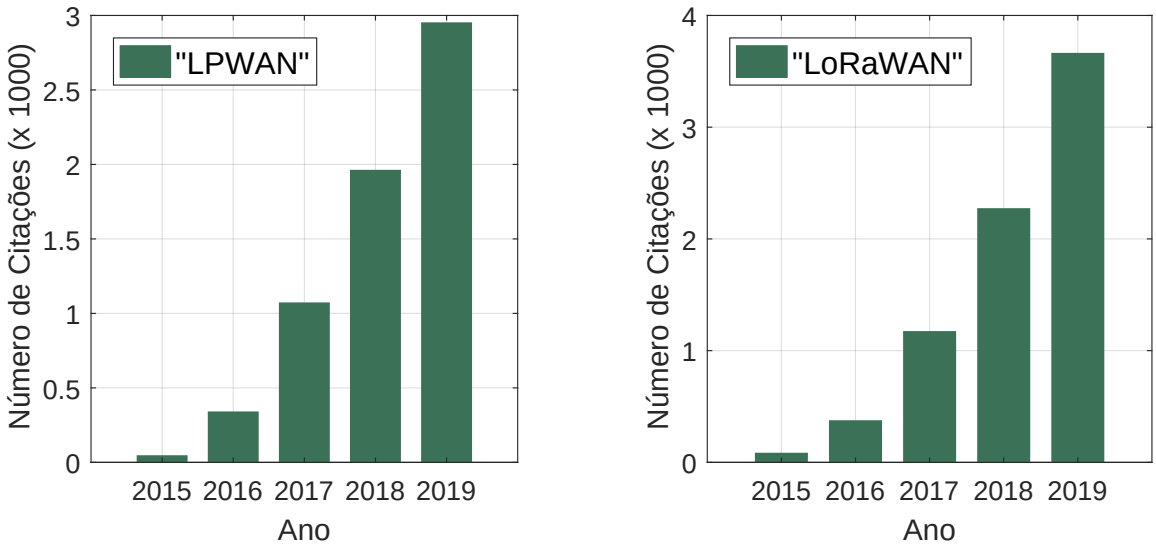
- Criação de uma rede LoRaWAN
- Configuração do Módulo LoraWAN RD49C
- Adaptação de *firmware* customizado

1.3 Justificativa

A quantidade de trabalhos publicados nos últimos anos é um indicativo da relevância e da atualidade da linha de pesquisa abordada neste projeto. Para se ter uma ideia, a quantidade de trabalhos no tema de "LoRaWAN" saltou de 81 em 2015 para 3660 em 2019, um crescimento de mais de 4000% em 4 anos. Esse crescimento no período 2015-2019 está ilustrado na Fig. 1, juntamente com a quantidade de menções ao tema "LPWAN", ambos obtidos de [16].

A pesquisa desenvolvida nesse projeto contribui para melhorar o desempenho de redes LoRaWAN, com impacto direto ou indireto nos seguintes contextos:

- **Científico:** Por se tratar de um tema atual e relevante, tendo em vista a grande quantidade de trabalhos publicados recentemente, o presente projeto contribui no contexto científico através da disponibilização dos resultados para a comunidade acadêmica;



(a) (b)

Figura 1: Quantidade de menções anuais aos temas "LPWAN" e "LoRaWAN" nos últimos 5 anos (2015-2019). Fonte: Google Scholar [16].

- **Ambiental:** Redes de sensores mais eficientes permitirão a implantação de aplicações de monitoramento nos mais diversos ambientes: subaquático (para monitorar qualidade da água, por exemplo), florestal (*e.g.*, para identificar queimadas), agropecuário (visando aumento da produtividade), *etc.* Além disso, redes com maior eficiência energética requerem a utilização uma quantidade menor de baterias, as quais, se descartadas livremente, são prejudiciais ao meio-ambiente;
- **Econômico:** Redes mais eficientes podem gerar economia de energia e aumento de produtividade no escopo da Indústria 4.0, por exemplo.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 LORAWAN

A LoRa convive e conversa com arquiteturas de rede já existentes. Isto permite que diversas pilhas de protocolos funcionem com a tecnologia[17]. Estes protocolos, junto a arquitetura de rede, definem características como segurança, vida útil de baterias, capacidade e qualidade do serviço. Atualmente, o protocolo de comunicação mais divulgado é a LoRaWAN. Utilizando a topologia estrela-de-estrelas, sua arquitetura de rede está esquematizada na Figura 2. Os *gateways* fazem as comunicações entre *End Nodes* através da comunicação sem fio *single-hop*. Os *End Nodes* não são atrelados a um único *gateway*, ou seja, os dados transmitidos por um *End Node* são aceitos por diversos *gateways*. Estes dados são então enviados para o servidor da rede na nuvem, através de outro tipo de transmissão, como Wi-Fi(do inglês, *wireless fidelity*) ou *Ethernet*. Toda a parte de processamento é deixada então para o servidor de rede, que irá realizar testes de segurança, filtrará dados redundantes e fará outros procedimentos mais complexos[7], como gerenciar as taxas de transmissão, por exemplo. Por fim, o servidor de aplicação tem a principal função de descriptografar as mensagens do *End Node* e repassá-las para a utilização da implementação.

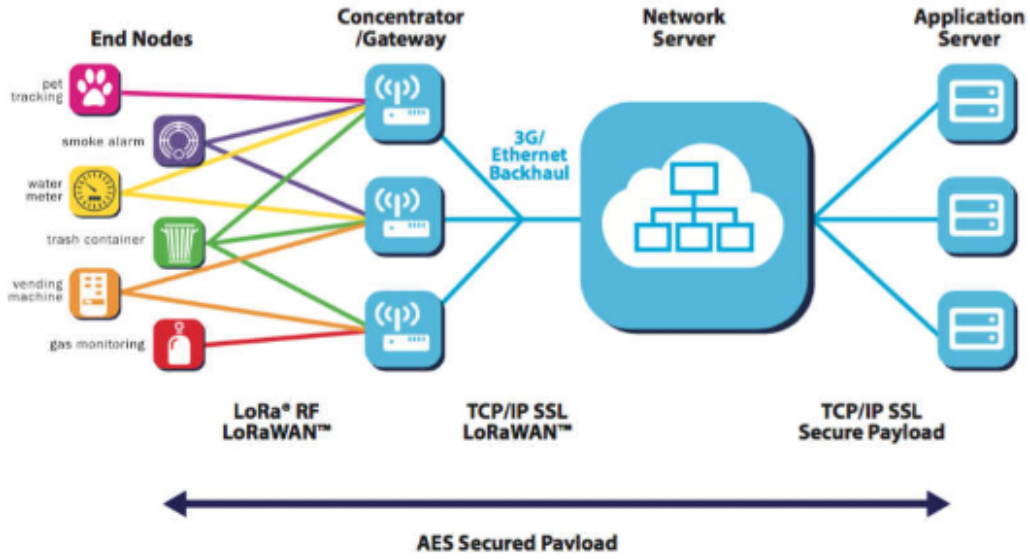


Figura 2: Esquemático do protocolo LoRaWAN. Fonte: Lora Alliance [7]

Os *End Nodes* na rede LoRaWAN são assíncronos, utilizando do método ALOHA. Neste método, os *End Nodes* não precisam "despertar" para sincronização e checar mensagens, e só se comunicam quando tem as informações prontas para envio, seja por programação ou devido ao acontecimento de um evento[17]. Como a sincronização é uma das maiores causas de gasto de bateria, o ALOHA vai ao encontro das necessi-

dades do LoRa. Além disso, como os *End Nodes* transmitem em diferentes taxas de transmissão, os *gateways* do LoRaWAN recebem mensagens simultâneas de múltiplos canais. Como os diferentes valores de fator de espalhamento fazem com que os sinais tenham diferentes taxas de transmissão e sejam quase ortogonais entre si, o *gateway* ainda recebe diferentes taxas de transmissão em um mesmo canal ao mesmo tempo, permitindo recepção de uma enorme quantidade de *End Nodes*.

Para aumentar o escopo de atendimento e optimização de diferentes aplicações, a LoRaWAN possui diferentes classes de *End Nodes*. Existem três classes, denominadas A, B e C, que trocam a latência de comunicação por duração de bateria. A classe A é a que menos consome energia e abrange as aplicações onde o *End Node* só precisa receber do servidor logo após transmitir. Ela possui duas curtas janelas de recepção após a transmissão para o servidor, com uma pequena variação de tempo aleatória. A informação do servidor em outro momento deve esperar até a próxima janela de recepção. A classe B possui janelas de recepção adicionais em períodos programados e recebe um *beacon* do *gateway* para que o *End Node* abra a janela de comunicação e o servidor tenha conhecimento disto. A classe C mantém a janela de recepção aberta, com exceção de quando está transmitindo[18].

A LoRaWAN possui a encriptação AES para trocas de chave utilizando um identificador IEEE EUI64 [18]. Ela garante tanto a autenticidade do *End Node* na rede quanto uma camada de segurança para que as informações do usuário não sejam acessadas pelo servidor de rede. Assim que um *End Node* é adicionado à rede, suas mensagens são encriptadas usando duas chaves: a chave da rede(NwkSKey, do inglês *Network Session Key*) e a chave da aplicação(AppSKey, do inglês *Application Session Key*). Cada uma dessas chaves é única para cada *End Node* e sessão de transmissão, mudando a cada ativação. A NwkSkey verifica a autenticidade das mensagens entre os *End Nodes* e o servidor da rede enquanto a AppsKey encripta as mensagens desde o *End Node* até o servidor de aplicação.

2.2 LMIC

LMIC se refere a IBM LoRaWAN C-library, uma implementação da especificação Lora MAC usando a linguagem C [19]. Ela suporta as classes A e B de dispositivos e aborda todos os estados lógicos MAC e restrições de tempo, considerando inclusive possíveis desvios do *clock*[19]. A biblioteca pode ser modificada para diversas plataformas usando o HAL(do inglês *Hardware Abstraction Layer*), ou seja, re-implementando as funções HAL com a semântica especificada. Desta forma, a execução do protocolo é assegurado pela biblioteca.

3 DESENVOLVIMENTO

Neste capítulo, serão apresentadas todas as práticas e atividades realizadas durante a pesquisa. Primeiro, um texto exemplificando todas as etapas efetuadas e depois algumas seções, em formato tutorial, com procedimentos mais particulares para facilitar a réplica destes.

O primeiro passo foi a implementação de uma rede LoRaWAN utilizando o Gateway LoRaWAN RD43HAT da Radioenge, conectado ao Raspberry Pi. O sistema operacional Raspbian foi instalado utilizando um gravador de imagem, o Win32DiskManager[20], em um *SD Card*. Com o Raspbian[21] instalado no Raspberry Pi, foi possível habilitar SSH (*Secure Shell*) e controlá-lo remotamente através do *software* Putty[22].

Foi habilitado o SPI(*Serial Peripheral Interface*) no Raspberry Pi e instalado o Git[23] para obter as bibliotecas e aplicações necessárias à rede. Então, foram instalados o *Lora Gateway project*[24], uma biblioteca com o código fonte para a construção de um receptor de radiofrequência, e o *packet forwarder*[25], um programa que encaminha os pacotes de radiofrequência recebidos pelo *gateway* para o servidor e vice-versa, ambos fornecidos pela Semtech. A configuração das larguras de banda utilizadas puderam ser obtidas no repositório *gateway-conf*[26], da TheThingNetwork, uma vez que as configurações regionais do Brasil coincidem com as configurações da Austrália.

Com o *packet forwarder* funcionando, foi necessária a instalação de outros elementos da rede LoRaWAN. Para isto, foram utilizados os serviços do Chirpstack LoRaWAN Network Server [27], que fornece os componentes da rede em código aberto assinados por uma chave PGP(*Pretty Good Privacy*). O Chirpstack utiliza um servidor MQTT(*Message Queuing Telemetry Transport*) para envio e recebimento de *payloads* nas aplicações. Portanto, foi instalado no Raspberry Pi a versão mais recente do Mosquitto[28], um *broker* de mensagens que implementa o protocolo de comunicação MQTT. Além disto, o Chirpstack utiliza o banco de dados PostgreSQL para o armazenamento de dados persistentes e o banco de dados Redis para armazenamento de dados não-persistentes. Desta forma, ambos os serviços foram instalados no Raspberry Pi e dois bancos de dados PostgreSQL foram criados, um para o *ChirpStack Network Server* e outro para o *ChirpStack Application Server*. O Chirpstack oferece em um repositório todos os pacotes dos serviços assinados por uma chave PGP. Assim, o repositório e a chave de assinatura foram adicionados e os seguintes serviços foram instalados:

- *ChirpStack Gateway Bridge:* processa a comunicação com os *gateways* LoRaWAN.
- *ChirpStack Network Server:* implementação de um LoRaWAN *Network Server*
- *ChirpStack Application Server:* implementação de um LoRaWAN *Application Server*

No arquivo de configuração do *ChirpStack Network Server*, foram adicionados o *login* e senha de seu banco de dados criado anteriormente e as informações sobre a faixa de frequência e sub-faixas utilizadas - de acordo com a configuração do *packet forwarder*. No arquivo de configuração do *ChirpStack Application Server* também foram adicionados o *login* e senha de seu respectivo banco de dados, além de uma senha na base 64 para o JWT(*JSON Web Token*).

Embora a Radioenge disponibilize um *software* para configuração do módulo, foi adotada a comunicação serial para a configuração através de comandos AT, por meio de um conversor USB Serial FTDI. Foi utilizado o programa Termite[29], um terminal RS232, para envio e recebimento de mensagens do módulo.

Primeiro, foi alterada a máscara de canais do módulo para que ela se adequasse às sub-faixas configuradas no *packet forwarder*. O parâmetro *Adaptative Datarate* foi desativado e a taxa de dados foi ajustada para um valor maior que zero. Então, o modo de ativação do módulo na rede foi escolhido. Ao definir o modo de ativação, o módulo retorna os parâmetros necessários para incluí-lo na rede LoRaWAN, como o EUI(*Extended Unique Identifier*) do dispositivo e a *Application Key*. Ambos os modos de ativação, OTAA(*Over the Air Authentication*) e ABP(*Authentication by Personalisation*), foram testados.

O *firmware* do Módulo LoRaWAN RD49C não é código aberto, portanto foi necessária a busca de repositórios que permitissem um *firmware* customizado. Primeiramente, foi desativada a proteção de *bytes* do módulo utilizando o programa STM32Cube Programmer[30]. Então, foi utilizado um ST-Link V2 para gravar um novo *firmware* através do *software* STM32 System Workbench[31], um programa baseado no Eclipse[32]. Para testes, um programa simples de comutação GPIO(*General Purpose Input Output*) foi obtido através dos pacotes STM32Cube MCU[33], da Semtech. O pacote STM32CubeL0[34] não contém o microcontrolador presente no Módulo LoraWAN RD49C (STM32L071CZ), mas contém o STM32l073RZ, de pinagem semelhante. Corrigindo as divergências entres os microcontroladores, a comutação funcionou corretamente.

Para o *firmware* customizado, a primeira opção foi o repositório Arduino-Lmic, uma biblioteca IBM LMIC[19] modificada para funcionar com a Arduino IDE [35]. Então, foi adicionada a biblioteca Arduino-LMIC na IDE do Arduino e selecionada a placa Nucleo-64 junto a Nucleo L073RZ. No arquivo de configuração, foi definido o transmissor do Módulo LoraWAN RD49C (SX1272) e os parâmetros para ativação do módulo na rede LoRaWAN. Por fim, foi aberto um exemplo da biblioteca para a entrada do módulo na rede por OTAA. Infelizmente, a transmissão de dados era inconsistente e, portanto, outro *firmware* era exigido.

O repositório encontrado foi LMiC[36], um variante da biblioteca IBM LMIC[19] com foco na portabilidade. O código foi alterado para que as configurações e conexões

das portas do microcontrolador combinassem com o Módulo LoraWAN RD49C. Mesmo com os parâmetros da biblioteca devidamente ajustados, havia um problema de sincronização na recepção de mensagens, identificado como uma contagem de tempo acelerada pelo LSI(*Low Speed Internal*)*clock* do microcontrolador O programa STM32Cube MX[37] foi utilizado para obter as configurações necessárias para que o HSE (*High Speed External*)*clock* gerasse a contagem de tempo. Desta forma, o módulo foi aceito pela rede.

As seções abaixo referem-se a procedimentos específicos realizados durante o projeto, usando frases afirmativas para facilitar o entendimento e a sua reprodução em trabalhos futuros.

3.1 Criação da rede LoRaWAN

3.1.1 Preparação do Raspberry Pi

Para isso, deve-se realizar os seguinte procedimentos: instale o sistema operacional Raspbian[21] no Raspberry Pi. Deve-se utilizar um gravador de imagem, como o Win32DiskManager[20], e baixar a imagem do Raspbian[21] no site oficial do Raspberry Pi. Com o Win32DiskManager instalado, conecte um SD Card ao computador e selecione-o no programa. Então, no programa, selecione o arquivo ".img", descompactado do arquivo do Raspbian, e clique em "Write". Desta forma, começará o processo de gravar a imagem no SD Card. Após o término do processo, insira o SD Card no Raspberry Pi e ele estará pronto para o uso.

Para controlar o Raspberry Pi remotamente, é preciso habilitar o SSH. para tal, digite no terminal

```
1 sudo raspi-config
```

vá até "5 Interface Options"e então "P2 Enable SSH". Com o SSH habilitado, conecte o Raspberry Pi à *Internet*. Novamente no terminal, entre o comando

```
1 ifconfig
```

e procure pela informação "inet addr"no campo "wlan0". Esse é o endereço de IP do Raspberry Pi na rede *Wi-fi*, que pode ser usado para conectar com outro dispositivo Linux ou com um cliente SSH, como o Putty [22].

3.1.2 Instalação do *Packet Forwarder* e do *Lora Gateway project*

É preciso habilitar o SPI(*Serial Peripheral Interface*) no Raspberry Pi. Para isto, digite no terminal

```
sudo raspi-config
```

e, com as setas do teclado, selecione “5 Interfacing Options” e então “P4 SPI”. Selecione “Yes” para habilitar o SPI e para quando o sistema solicitar *reboot*. Quando o sistema voltar, logue-se novamente e escreva o comando

```
ls /dev/*spi*
```

O Raspberry Pi deve retornar “/dev/spidev0.0 /dev/spidev0.1”

Agora, é preciso baixar os repositórios *lora_gateway*[24] e *packet_forwarder* [25]. Para isto, instale o Git[23] no Raspberry Pi, usando

```
sudo apt-get install git
```

Para baixar o *lora_gateway* utilize o comando

```
git clone https://github.com/Lora-net/lora_gateway.git
```

e, após o *download* deste, utilize o comando

```
git clone https://github.com/Lora-net/packet_forwarder.git
```

para o *packet_forwarder*. Pode-se instalar os repositórios através do comando *make*:

```
cd ~/lora_gateway
make all
cd ~/packet_forwarder
make all
```

Caso o erro “make: Nothing to be done for ‘all” ocorra, utilize

```
make clean
```

antes do comando *make*.

3.1.3 Configuração das larguras de banda

Baixe o Nano[38] com o comando

```
sudo apt-get install nano
```

Acesse o diretório do *packet_forwarder* com

```
cd ~/packet_forwarder/lora_pkt_fwd
```

e então utilize o editor de texto para alterar o arquivo "global_conf.json". Com o Nano, utilize

```
sudo nano global_conf.json
```

Acesse o repositório *gateway-conf*[26], da TheThingsNetwork, e copie para a área de transferência a configuração da Austrália, o arquivo "AU-global_conf.json". Apague tudo que está no arquivo "global_conf.json"e cole a configuração da Austrália. Além disso, altere, no campo "gateway_conf", o parâmetro "server_address". Neste caso, mude o campo para "server_address": "127.0.0.1". No campo "gateway_conf" também pode-se atualizar o *fake_gps* com os parâmetros

```
"fake_gps":true,
"ref_latitude":0,
"ref\_longitude":0,
"ref\_altitude":0,
```

Essas configurações, à exemplo do "gateway_ID", podem ser sobrescritas pelo arquivo "local_conf.json". Salve o arquivo.

O *Packet Forwarder* pode ser executado mudando o diretório com

```
cd ~/packet_forwarder/lora_pkt_fwd
./lora_pkt_fwd
```

Caso ele não inicie, será necessário reiniciá-lo. Vá para o diretório do lora_gateway e execute o programa "reset_lgw.sh" com o parâmetro "start", com os comandos

```
cd ~/lora_gateway
./reset_lgw.sh start
```

Então, tente executar o *Packet Forwarder* novamente. “CRTL+C” desativa o *Packet Forwarder*.

3.1.4 Instalação dos serviços *Chirpstack*

A Chirpstack utiliza um servidor MQTT para publicar e receber *payloads*. Utilize o comando

```
sudo apt-get install mosquitto
```

para instalar a versão mais recente do Mosquitto. Depois, obtenha a chave PGP com

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1
    CE2AFD36DBCCA00
```

Caso o comando não funcione, tente

```
sudo gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 1
    CE2AFD36DBCCA00
```

Então, baixe os pacotes com

```
sudo echo "deb https://artifacts.chirpstack.io/packages/3.x/deb stable
    main" | sudo tee /etc/apt/sources.list.d/chirpstack.list
```

e atualize com

```
sudo apt-get update
```

Para instalar o *ChirpStack Gateway Bridge*, utilize

```
sudo apt-get install chirpstack-gateway-bridge
```

O serviço pode ser executado com o parâmetro “start” no comando

```
sudo systemctl start chirpstack-gateway-bridge
```

Outros parâmetros são ”stop”, ”restart”e ”status”. O *log-output* do serviço pode ser acessado com o comando

```
journalctl -u chirpstack-gateway-bridge -f -n 50
```

Antes de instalar o *ChirpStack Network Server* é preciso criar as bases de dados *PostgreSQL* e *Redis*. Instale-os através dos comandos

```
sudo apt-get install postgresql
sudo apt-get install redis-server
```

Abra o *prompt* do *PostgreSQL* com

```
sudo -u postgres psql
```

e crie uma base de dados com os comandos:

```
create role chirpstack_ns with login password 'dbpassword';
create database chirpstack_ns with owner chirpstack_ns;
```

Utilize **\q** para sair do *prompt*. Para testar o servidor, tente conectar-se a ele com

```
psql -h localhost -U chirpstack_ns -W chirpstack_ns
```

Para instalar o *ChirpStack Network Server* use

```
sudo apt-get install chirpstack-network-server
```

Alguns parâmetros no arquivo "chirpstack-network-server.toml"devem ser alterados para se adequarem às configurações do *Packet Forwarder* e para adicionar as informações do banco de dados criado. Para alterar o repositório, é preciso estar como *root*. Utilize o comando

```
sudo su
```

para entrar nesse modo. Como *root*, utilize

```
cd/etc/chirpstack-network-server
```

e acesse o arquivo com o editor de textos

```
nano chirpstack-network-server.toml
```

Em primeiro lugar, vá até a primeira linha não-comentada após “[postgresql]”. Comente esta linha usando o caractere “#”. Então escreva, logo abaixo

```
dsn="user=chirpstack_ns password=dbpassword dbname=chirpstack_ns
    sslmode=disable"
```

Desça pelo arquivo até encontrar o campo “[network_server.band]”. Renomeie a banda para “AU_915_928” com

```
name=AU_915_928
```

Embaixo, logo após “[network_server.network_settings]”, escreva

```
enabled\_uplink\_channels=[8, 9, 10, 11, 12, 13, 14, 15]
```

Os números vêm do arquivo ”global_conf.json”que não tem a primeira frequência em 915MHz, e sim 916.8MHz. Por fim, comente todas as configurações de canais extras, ou seja, desde “Extra Channel Configuration” até “Class B Settings”. Salve o arquivo modificado. Você pode sair do modo *root* digitando

```
exit
```

O serviço pode ser executado com o parâmetro “start” no comando

```
sudo systemctl [start|stop|restart|status] chirpstack-network-server
```

O *log-output* do serviço pode ser acessado com o comando

```
journalctl -u chirpstack-network-server -f -n 50
```

O *ChirpStack Application Server* também precisa de uma base de dados. Entre novamente no *prompt* do *PostgreSQL* com

```
sudo -u postgres psql
```

e crie uma base de dados com os comandos:

```
create role chirpstack_as with login password 'dbpassword';
create database chirpstack_as with owner chirpstack_as;
\c chirpstack_as
create extension pg_trgm;
create extension hstore;
```

Utilize **\q** para sair do *prompt*. Para testar o servidor, tente conectar-se a ele com

```
psql -h localhost -U chirpstack_as -W chirpstack_as
```

Para instalar o *ChirpStack Application Server* use

```
sudo apt-get install chirpstack-application-server
```

Como *root*, utilize

```
cd /etc/chirpstack-application-server
```

e acesse o arquivo de configuração com o editor de textos

```
nano chirpstack-application-server.toml
```

Da mesma forma que o *ChirpStack Network Server*, altere a primeira linha não comentada após "[postgresql]" para

```
dsn="user=chirpstack_as password=dbpassword dbname=chirpstack_as
    sslmode=disable"
```

Altere também o parâmetro "jtw_secret", logo acima de "[join_server]", com uma senha na base 64. Você pode adquirir uma com o comando

```
openssl rand -base64 32
```

A linha deve ficar parecida com

```
jwt_secret="wMbgFGKkDObEvrPooikRBEfcb3QoSESf0eBSLW9gR5w="
```

Salve o arquivo e então saia do modo *root* com **exit**. O serviço pode ser executado com o parâmetro “start” no comando

```
sudo systemctl [start|stop|restart|status] chirpstack-application-
    server
```

O *log-output* do serviço pode ser acessado com o comando

```
journalctl -u chirpstack-application-server -f -n 50
```

3.1.5 Configuração da *Web Interface*

Com todos os serviços sendo executados, abra um navegador e digite o IP do Raspberry Pi na rede - disponível em”inet addr”no campo ”wlan0”, quando executado o comando

```
ifconfig
```

no terminal - mais a porta 8080. A *url* no navegador deverá se parecer com **http://192.168.3.12:8080/**. Na aba do lado esquerdo, clique em “Network-servers” e então ”+Add”. Escolha um nome para o Network Server e em “Network-server server” coloque **0.0.0.0:8000**, informação disponível no arquivo ”chirpstack-network-server.toml”, abaixo de “[network_server.api]”. Salve em “Add Network-Server” e vá para a aba da esquerda, “Gateway-profiles” e depois em “+Create”. Digite um nome para o Gateway e em “Network-server*” selecione o Network Server criado anteriormente. Em “Enabled channels”, escreva 8, 9, 10, 11, 12, 13, 14, 15 - número das subfaixas escolhidas em ”chirpstack-network-server.toml- e clique em “Create Gateway-profile”.

Em “Organizations”, clique em “+Create”, preencha os campos e marque a caixa “Organization can have Gateways”. Salve a organização com “Create Organization”, ou simplismente use a organização criada Chirpstack pronta. Agora selecione a organização e vá para “Service-profiles” e “+Create”. Dê um nome para o serviço e salve com “Create Service-profile”.

Por fim, selecione “Device-profile” e “+Create”. Escolha um nome para o perfil do dispositivo, selecione o Network Server criado e a versão LoRaWAN do dispositivo. Para o Módulo LoraWAN RD49C, altere o campo “LoRaWAN MAC version” para “1.0.3”, “LoRaWAN Regional Parameters revision” para “A” e o restante deixe zerado. Se o dispositivo for ativado por OTAA marque a opção “Device supports OTAA” na aba “Join (OTTA/ABP)”. Caso opte por ABP, coloque “1” no campo de “RX1 delay*”

e escreva 8, 9, 10, 11, 12, 13, 14, 15 em “Factory-preset frequencies (Hz)”. Salve o perfil em “Create Device-Profile”.

Em “Gateways”, na aba da esquerda, clique em “+Create”. Após digitar o nome e descrição do Gateway, selecione os respectivos perfis criados em “Network-server*” e “Gateway-profile*”. O “Gateway ID” está disponível no arquivo ”local_conf.json”. Pode-se obtê-lo com os comandos

```
cd ~/packet_forwarder/lora_pkt_fwd
sudo nano local_conf.json
```

Salve o *gateway* em “Create Gateway”.

Em “Application”,clique em “+Create”. Escolha um nome, descreva a aplicação, selecione o perfil do serviço criado anteriormente e então “Create Application”. Novamente em “Application”, clique na aplicação criada e, na aba “Devices”, clique em “+Create”. Digite o nome, a descrição do dispositivo e selecione o perfil criado em “Device-profile”. Preencha o campo “Device EUI” e clique em “Create Device”. Se optou por OTAA a página o redirecionará para uma aba denominada ”Keys(OTAA)”, onde haverá o campo “Application Key” para ser preenchido. O outro campo pode ser deixado em branco. Caso tenha optado por ABP, há os campos “Device address”, “Network session key” e “Application session key” a serem preenchidos. Clique em “Set Device-Keys” e o dispositivo foi adicionado a rede. A forma de obter estes parâmetros é discutido na próxima seção.

3.2 Configuração do Módulo LoraWAN RD49C

Para controlar o módulo por meio de comandos AT, é preciso conetá-lo a um conversor USB/Serial FTDI. O esquemático do módulo está na Figura 3 e a descrição dos pinos na Tabela 1. Conecte o FTDI e o módulo em um *protoboard* e faça as conexões necessárias. Lembre-se que a porta RX do módulo deve estar conectada à porta TX do FTDI, e vice-versa. Recomenda-se também que os *grounds* estejam curto-circuitados. Então, baixe e instale o programa Termite[29]. No programa, clique nas configurações e defina a ”Baud rate”para 9600, padrão do módulo. Conecte o FTDI ao computador e selecione a porta utilizada no programa. Desta forma, o módulo deve estar pronto para receber os comandos AT.

Primeiramente, altere a máscara de canais para que adeque ao arquivos do ”global-conf.json”: no Termite, escreva

```
AT+CHMASK=ff00:0000:0000:0000:0002:0000
```

Então, altere o “ADR” do dispositivo para zero, através do comando

Tabela 1: Numeração e função dos pinos do Módulo LoraWAN RD49C da Radioenge.
Fonte: Elaborado pelo autor.

Pino	Nome	Tipo	Descrição
1	GND	Alimentação	Conectado ao Ground
2	RX_1	Entrada	RX da interface UART de comandos
3	TX_1	Saída	TX da interface UART de comandos
4	VCC	Alimentação	Conectado à Alimentação
5	VCC	Alimentação	Conectado à Alimentação
6	GPIO0	Entrada/Saída	Pino de uso geral ou entrada analógica
7	GPIO1	Entrada/Saída	Pino de uso geral ou entrada analógica
8	GND	Alimentação	Conectado ao Ground
9	GPIO2	Entrada/Saída	Pino de uso geral ou entrada analógica
10	GPIO3	Entrada/Saída	Pino de uso geral ou entrada analógica
11	GPIO4	Entrada/Saída	Pino de uso geral ou entrada analógica
12	GPIO5	Entrada/Saída	Pino de uso geral ou entrada analógica
13	GPIO6	Entrada/Saída	Pino de uso geral ou entrada analógica
14	GPIO7	Entrada/Saída	Pino de uso geral ou entrada analógica
15	GPIO8	Entrada/Saída	Pino de uso geral ou entrada analógica
16	GPIO9	Entrada/Saída	Pino de uso geral ou entrada analógica
17	GND	Alimentação	Conectado ao Ground
18	ANT	Saída RF	Pino de uso geral ou entrada analógica
19	GND	Alimentação	Conectado ao Ground
20	*SWDIO	-	-
21	*GND	Alimentação	Conectado ao Ground
22	*SWCLK	-	-
23	*RESET	-	-

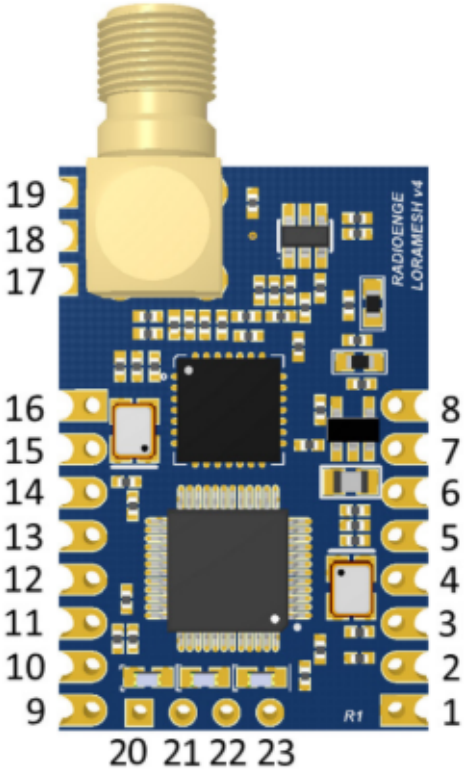


Figura 3: Vista superior do Módulo LoraWAN RD49C da Radioenge. Fonte: Módulo LoRaMESH - Manual de Utilização [39]

```
AT+ADR=0
```

e o *datarate* para algum valor entre 0 e 5, com o comando

```
AT+DR=4
```

Pode-se definir o modo de ativação do módulo com o comando

```
AT+NJM=0
```

para o ABP e

```
AT+NJM=1
```

para o OTAA. Após enviar um dos comandos, o módulo deve retornar as chaves e valores necessários para ativá-lo na rede, como o EUI e a APPKEY.

3.3 Adaptação de *firmware* customizado

Instale um programa chamado STM32CubeProgrammer[30], da STMicroelectronics. Com o programa devidamente baixado e instalado, deve-se conectar o módulo da Radioenge com um ST-Link V2, seguindo a pinagem do dispositivo na Figura 3. Serão utilizadas as porta 1, 4, 5, 8, 20, 21, 22 e 23. É recomendado que todos os pinos *ground*, inclusive os do ST-Link, estejam curto-circuitados.

Com o módulo conectado com o ST-Link V2 e o programa STM32CubeProgrammer aberto, deve se selecionar, no canto superior direito, a opção ST-LINK e clicar em "Connect" . Possivelmente, uma mensagem com o erro "ST-LINK error(DEV_TARGET_CMD_ERR)" aparecerá. Neste caso, você deverá acionar o pino de RESET do Mote (ligando-o ao *ground*), clicar em "Connect" novamente e, quase imediatamente depois, liberar o pino. O programa deve reconhecer o dispositivo, com o erro "Data read failed" aparecendo em tela e oferecendo a opção de desconectar o dispositivo. Caso isto não ocorra, significa que o *timing* de retirar o pino RESET após clicar em "Connect" foi demasiado lento ou demasiado rápido. Algumas tentativas podem ser necessárias até acertar este *timing*.

Com o dispositivo reconhecido, feche o erro "Data read failed", vá até "Option Bytes", no canto esquerdo, e abra a aba "Read Out Protection". Na segunda coluna, o valor do RDP deve estar como "BB". Mude para "AA" e depois clique em "Apply". Uma mensagem deve avisar que a operação foi concluída com sucesso. Desta forma, o *firmware* atual do dispositivo deve ser apagado clicando na opção "Full chip erase", no canto inferior esquerdo, e ele está pronto para receber um novo.

Para gravar um novo *firmware* será utilizado o STM32 System Workbench [31], uma IDE baseada no Eclipse[32]. No site, é possível baixar *plug-ins* caso já tenha o *software* do Eclipse instalado ou baixar um instalador com a IDE pronta para o uso. Para configurá-la, precisamos de um projeto qualquer. Para exemplificar, utilizaremos um projeto simples, de piscar os LEDs do dispositivo, de um pacote de ferramentas e softwares integrados denominado STM32Cube[33].

Nestes pacotes não existe o microcontrolador exato usado pela Radioenge (STM32l071CZ), mas pode-se utilizar um bastante parecido, o STM32l073. Então, baixe o pacote STM32CubeL0[34]. Lembre-se de baixar o pacote e não o software STM32 Cube MX, disponível na mesma página. Com o pacote baixado, será preciso alterar um arquivo de configuração, pois a porta especificada do LED é diferente entre os dois microcontroladores. Vá até a paste "STM32L0xx_Nucleo", onde existe um arquivo denominado "stm32l0xx_nucleo". Clique com o botão direito do *mouse* sobre ele e vá em "Propriedades". Desmarque o atributo "Somente leitura". Então, abra o arquivo e substitua os valores de **LED2_PIN** de **GPIO_PIN_5** para **GPIO_PIN_8**. Também mude as configurações de **GPIOA** para **GPIOB**. No total, quatro defines devem ser alterados, ficando da seguinte forma:

```
#define LED2_PIN                                GPIO_PIN_8
#define LED2_GPIO_PORT                          GPIOB
#define LED2_GPIO_CLK_ENABLE()                __HAL_RCC_GPIOB_CLK_ENABLE()
#define LED2_GPIO_CLK_DISABLE()               __HAL_RCC_GPIOB_CLK_DISABLE()
```

Salve e feche o arquivo. Abra o System Workbench e, no canto superior esquerdo, abra a aba “File”. Então, clique em “Open Projects from File System...” e abra o projeto. Para chegar até lá, clique em ”Projects”, escolha ”NUCLEO-L073RZ”, ”Examples”e então ”GPIO”e ”GPIO_IOToggle”. Na pasta ”SW4STM32”encontrará ”STM32L073RZ_NUCLEO”, que contém um programa simples de piscar o LED verde da placa. No canto superior do System Workbench, na aba de “Project”, clique em “Build All”, ou simplesmente utilize as teclas “CTRL+B”, para compilar o projeto.

Por fim, na aba “Run”, clique em “Run Configurations” e depois duas vezes em “Ac6 STM Debugging”. Nesta configuração, na segunda aba “Debugger” vá até “Configuration Script”, selecione “Automated Configuration” e do lado “Show generator options...”. Logo abaixo, mude o *reset mode* para “Software system reset”. Com isso configurado, você já pode clicar em “Apply” e então “Run”. Após um tempo de envio para a placa, ela deve começar imediatamente a rodar o projeto.

4 RESULTADOS OBTIDOS

As Figuras 4 e 5 mostram o equipamento utilizado para implementar a rede LoRaWAN. A primeira mostra o Gateway LoRaWAN RD43HAT da Radioenge acoplado ao Raspberry Pi, enquanto a segunda mostra o Módulo LoraWAN RD49C conectado à placa FTDI FT232RL, para comunicação serial, e ao ST-LINK V2, para a gravação de novo *firmware*.

Figura 4: Gateway LoRaWAN RD43HAT da Radioenge sobre o Raspberry Pi. Fonte: Autoria Propria.

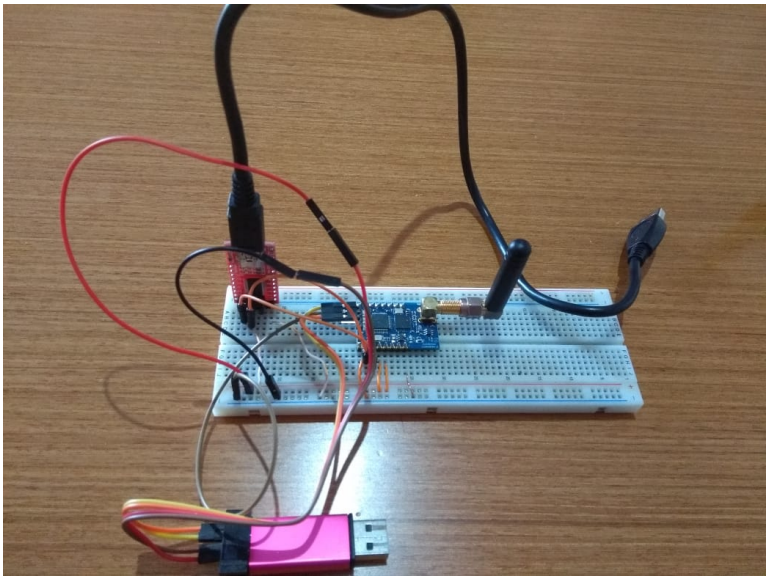


Figura 5: Módulo LoraWAN RD49C da Radioenge conectado ao St-Link V2 e a placa FTDI FT232RL. Fonte: Autoria Propria.

Cada serviço do Chirpstack fornece um *log-output*, que mostra os pacotes de radi-

ofrequência recebidos e enviados. A Figura 6 mostra o *log-output* do *packet forwarder* e as Figuras 7, 8 e 9 representam, respectivamente, o *log-output* do *ChirpStack Gateway Bridge*, *ChirpStack Network Server* e *ChirpStack Application Server* quando um módulo entra na rede.

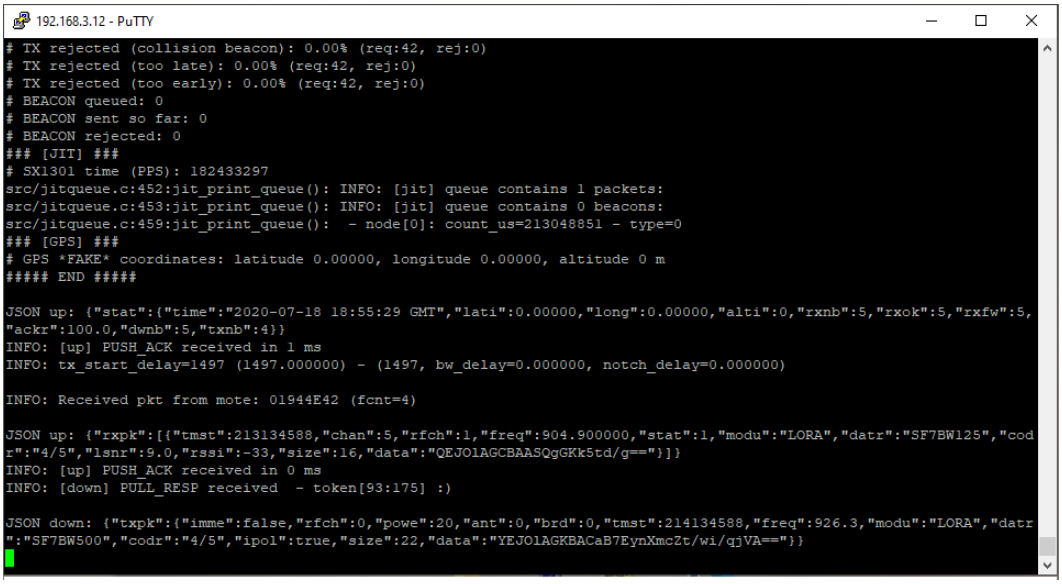


Figura 6: *Log-output* do *Packet Forwarder*. Fonte: Autoria Própria.

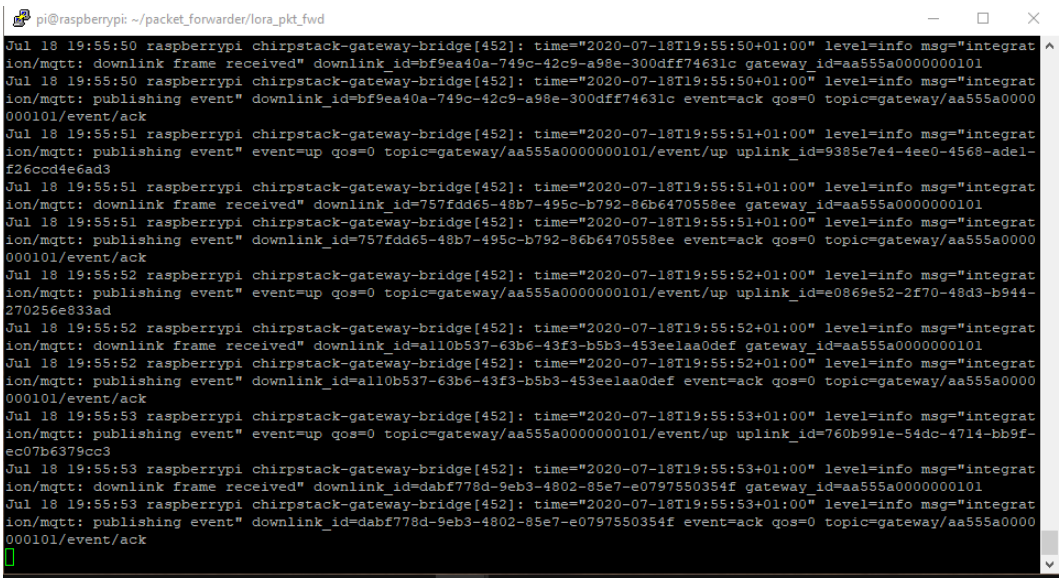


Figura 7: *Log-output* do *Chirpstack Gateway Bridge*. Fonte: Autoria Própria.

A Figura 10 mostra o Termite quando o Módulo LoraWAN RD49C envia o pedido de entrada na rede e é aceito, utilizando a biblioteca LMiC [36], enquanto a Figura 11 representa o STM32CubeMX quando utilizado para obter os valores dos parâmetros do *clock* para o mesmo *firmware*.

Em síntese, os resultados obtidos foram muito satisfatórios. A biblioteca utilizada

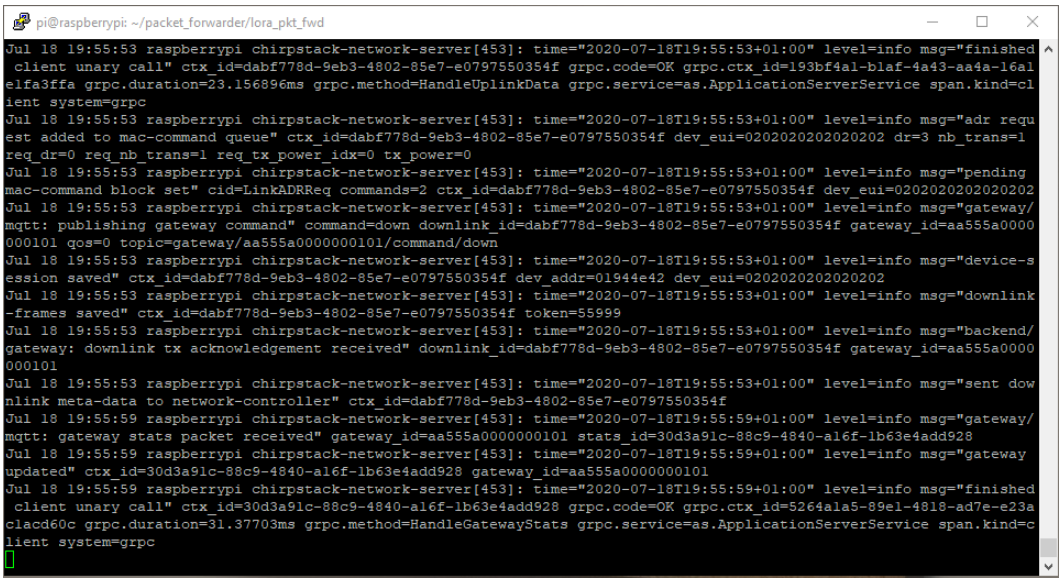


Figura 8: *Log-output* do *Chirpstack Network Server*. Fonte: Autoria Própria.

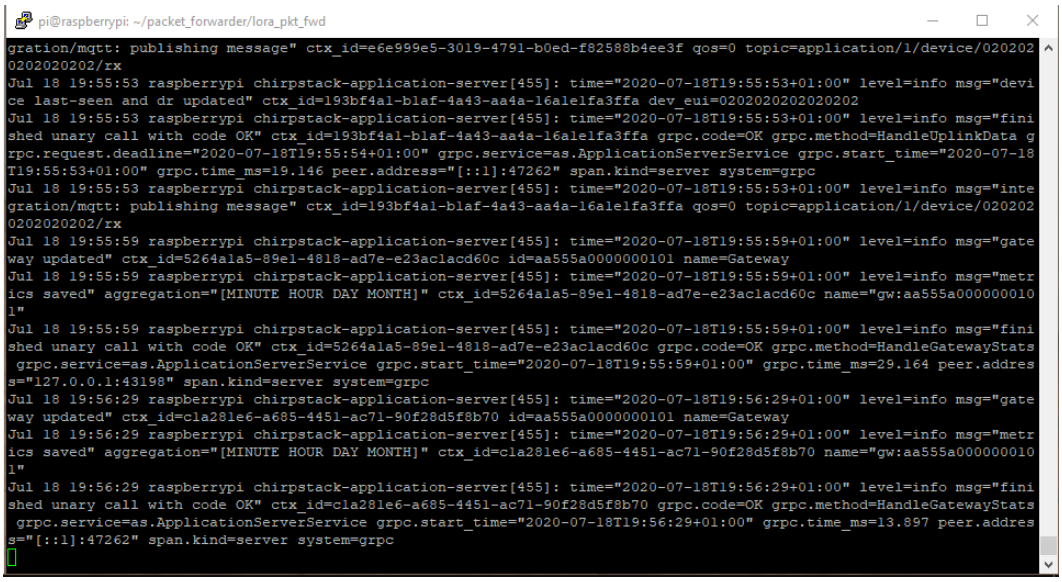


Figura 9: *Log-output* do *Chirpstack Application Server*. Fonte: Autoria Própria.

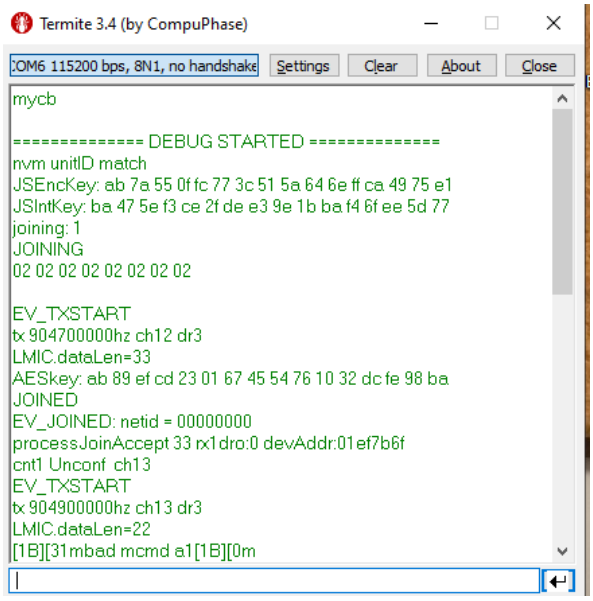


Figura 10: Visualização dos pacotes de radiofrequência no Termite. Fonte: Autoria Própria.

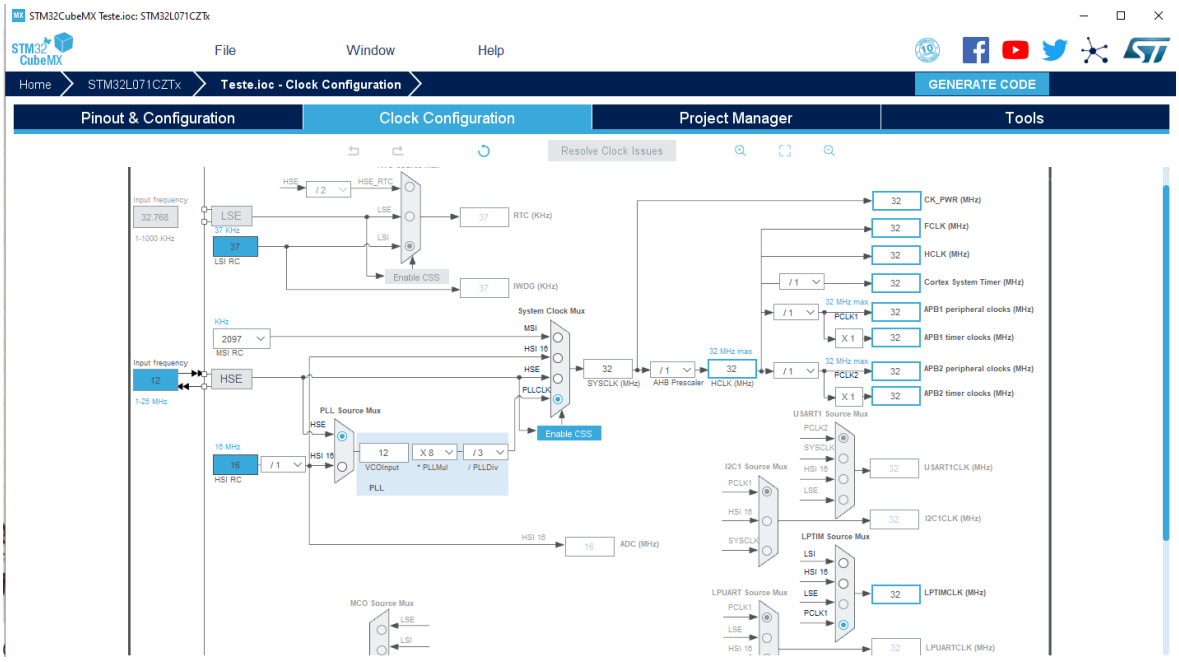


Figura 11: Configuração utilizada para ajuste dos parâmetros do *clock* no STMCUBEMX. Fonte: Autoria Própria.

para o *firmware* se provou bem adaptável, uma vez que vários dos parâmetros necessários foram facilmente alterados no código. Além disso, a solução apresentada ao final do projeto é bastante estável, em razão do envio e recebimento de mensagens entre as camadas, bem como a entrada do módulo na rede, serem realizadas com êxito.

5 CONCLUSÕES

As atividades realizadas nesta iniciação científica se provaram bastante proveitosas. O estudo de artigos, *datasheets* e manuais permitiram uma grande compreensão não somente do protocolo LoRaWAN em si, mas também das tecnologias LPWAN. Muito embora estas ainda enfrentem problemas de implementação e padronização, existe um interesse acadêmico e econômico muito grande por trás da tecnologia, e novas soluções surgem diariamente.

A modulação LoRa, junto ao protocolo LoRaWAN, provou-se uma solução bastante notável e confiável, destacando-se entre as demais tecnologias do setor. A criação de uma rede LoRaWAN do zero favoreceu um melhor entendimento das características e propriedades do LoRa, além de propiciar conhecimentos não diretamente relacionados, como o uso do terminal no Raspberry Pi e atuar com programação de microcontroladores.

Quanto à documentação de todos os processos realizados, esta facilitará a reprodução destes para o trabalho de pesquisa da comunidade acadêmica. Além disto, ela permite um ponto de partida mais acessível para novos desenvolvimentos tecnológicos.

6 REFERÊNCIAS

[1] J. P. Shanmuga Sundaram, W. Du, and Z. Zhao, "A Survey on LoRa Networking: Research Problems, Current Solutions, and Open Issues," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 1, pp. 371–388, Firstquarter 2020.

[2] J. Petäjäjärvi, K. Mikhaylov, A. Roivainen, and T. Hänninen, "On the Coverage of LPWANs: Range Evaluation and Channel Attenuation Model for LoRa Technology," *Proc. 14th Int. Conf. ITS Telecommun. (ITST)*, pp. 55–59, Dec 2015.

[3] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[4] G. Strazdins, A. Elsts, K. Nesenbergs, and L. Selavo, "Wireless sensor network operating system design rules based on real-world deployment survey," *Journal of Sensor and Actuator Networks*, vol. 2, pp. 509–556, 08 2013.

[5] C.Kuhlins, B. Rathonyi, A. Zaidi, and M. Hogan, "Cellular networks for massive iot," Ericson, Tech. Rep. Uen 284 23-3278, Jan 2020.

[6] A. Ikpehai, B. Adebisi, K. M. Rabie, K. Anoh, R. E. Ande, M. Hammoudeh, H. Gacanin, and U. M. Mbanaso, "Low-Power Wide Area Network Technologies for Internet-of-Things: A Comparative Review," vol. 6, no. 2, pp. 2225–2240, April 2019.

[7] "A technical overview of lora and lorawan," Lora Alliance, Tech. Rep., Nov 2015.

[8] L. Vangelista, A. Zanella, and M.Zorzi, "Long-range iot technologies: The dawn of lora," 09 2015, pp. 51–58.

[9] O. Georgiou and U. Raza, "Low Power Wide Area Network analysis: Can LoRa scale?" vol. 6, no. 2, pp. 162–165, April 2017.

[10] A. Mahmood, E. Sisinni, L. Guntupalli, R. Rondón, S. A. Hassan, and M. Gidlund, "Scalability analysis of a LoRa network under imperfect orthogonality," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1425–1436, March 2019.

[11] A. Hoeller, R. D. Souza, O. L. Alcaraz López, H. Alves, M. de Noronha Neto, and G. Brante, "Analysis and performance optimization of LoRa networks with time and antenna diversity," *IEEE Access*, vol. 6, pp. 32 820–32 829, 2018.

[12] J. Sant'Ana, A. Hoeller Jr, R. Souza, S. Montejo Sánchez, H. Alves, and M. Neto, "Hybrid Coded Replication in LoRa Networks," *IEEE Transactions on Industrial Informatics*, vol. PP, pp. 1–1, 01 2020.

[13] S. Hsu, C. Lin, C. Wang, and W. Chen, "Breaking Bandwidth Limitation for Mission-Critical IoT Using Semisequential Multiple Relays," vol. 5, no. 5, pp. 3316–3329, Oct 2018.

[14] K. Mikhaylov, J. Petäjäjärvi, J. Haapola, and A. Pouttu, "D2D communications in LoRaWAN Low Power Wide Area Network: From idea to empirical validation," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2017, pp. 737–742.

[15] J. Kim and J. Song, "A secure device-to-device link establishment scheme for LoRaWAN," *IEEE Sensors Journal*, vol. 18, no. 5, pp. 2153–2160, March 2018.

[16] "Google scholar," www.scholar.google.com, acessado em 07/07/2020.

[17] "Lora modulation basics," Semtech, Tech. Rep. AN1200.22, May 2015.

[18] N. Sornin, *LoRaWAN 1.1 Specification*, Lora Alliance, Oct. 2017, rev. 1.1.

[19] *IBM LoRaWAN in C (LMiC)*, IBM Zurich Research Laboratory, 8803 Rüschlikon, Switzerland, Jul. 2016, version 1.6.

[20] "Win32 disk imager," https://sourceforge.net/projects/win32diskimager/, acessado em 29/07/2020.

[21] "Raspberry pi os," https://www.raspberrypi.org/downloads/raspberry-pi-os/, acessado em 29/07/2020.

[22] "Putty," https://www.putty.org/, acessado em 30/07/2020.

[23] "Git," https://git-scm.com/, acessado em 29/07/2020.

[24] "Lora gateway project," https://github.com/Lora-net/lora_gateway, acessado em 29/07/2020.

[25] "Packet forwarder," https://github.com/Lora-net/packet_forwarder, acessado em 29/07/2020.

[26] "Ttn master gateway configurations," https://github.com/TheThingsNetwork/gateway-conf, acessado em 29/07/2020.

[27] "Chirpstack open-source lorawan network server," https://www.chirpstack.io/, acessado em 04/08/2020.

[28] "Eclipse mosquitto," https://mosquitto.org/, acessado em 04/08/2020.

[29] "Termite," https://www.compuphase.com/software_termite.htm, acessado em 30/07/2020.

[30] “Stm32cubeprogrammer software for all stm32,” https://www.st.com/en/development-tools/stm32cubeprog.html#get-software, acessado em 02/08/2020.

[31] “Stm32 system workbench,” https://www.openstm32.org/Installing%2BSystem%2BWorkbench%2Bfor%2BSTM32%2Bwith%2Binstaller, acessado em 02/08/2020.

[32] “Eclipse,” https://www.eclipse.org/, acessado em 02/08/2020.

[33] “Stm32cube mcu mpu packages,” https://www.st.com/en/embedded-software/stm32cube-mcu-mpu-packages.html#products, acessado em 02/08/2020.

[34] “Stm32cube mcu package for stm32l0 series,” https://www.st.com/content/st_com/en/products/embedded-software/mcu-mpu-embedded-software/stm32-embedded-software/stm32cube-mcu-mpu-packages/stm32cubel0.html, acessado em 02/08/2020.

[35] “Arduino,” https://www.arduino.cc/en/Main/Software, acessado em 04/08/2020.

[36] “Lmic,” https://github.com/dudmuck/LMiC, acessado em 04/08/2020.

[37] “Stm32cubemx,” https://www.st.com/en/development-tools/stm32cubemx.html, acessado em 04/08/2020.

[38] “GNU nano,” https://www.nano-editor.org/, acessado em 29/07/2020.

[39] *Módulo LoRaMESH - Manual de Utilização*, Radioenge, Feb. 2020.