UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE ENGENHARIA DE COMPUTAÇÃO

VICTOR ALBERTI COSTA

SISTEMA DE MONITORAMENTO VEICULAR APLICADO AO TRANSPORTE COLETIVO

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2021

VICTOR ALBERTI COSTA

SISTEMA DE MONITORAMENTO VEICULAR APLICADO AO TRANSPORTE COLETIVO

Trabalho de Conclusão de Curso apresentado ao Curso de engenharia de computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Gustavo Weber Denardin
Universidade Tecnológica Federal do Paraná

Coorientador: Jeferson Jose De Lima
Universidade Tecnológica Federal do Paraná

PATO BRANCO
2021

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO - TCC

SISTEMA DE MONITORAMENTO VEICULAR APLICADO AO TRANSPORTE COLETIVO

Por

Victor Alberti Costa

Monografia apresentada às 18 horas 00 min. do dia 02 de setembro de 2021 como requisito parcial, para conclusão do Curso de Engenharia da Computação da Universidade Tecnológica Federal do Paraná, Campus Pato Branco. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação e conferidas, bem como achadas conforme, as alterações indicadas pela Banca Examinadora, o trabalho de conclusão de curso foi considerado APROVADO.

Banca examinadora:

Prof. Dr. Diogo Ribeiro Vargas	Membro
Prof. Dr. Marcelo Teixeira	Membro
Prof. Dr. Gustavo Weber Denardin	Orientador
Profa. Dra. Viviane Dal Molin de Souza	Professor(a) responsável TCCII

Dedico este trabalho à minha família, que esteve ao meu lado em todos os momentos. E aos amigos que me apoiaram durante esta importante jornada. Obrigado!

AGRADECIMENTOS

Agradeço à minha família, por acreditar em mim, estar ao meu lado sempre e investir nos meus sonhos.
Agradeço a minha namorada, por sempre me apoiar, me ouvir, incentivar e acreditar em mim, e também por criar o nome PatoBus para o aplicativo.
Agradeço aos meus orientadores, pela amizade e todo suporte durante o desenvolvimento deste trabalho.
Agradeço a UTFPR que me proporcionou momentos e amizades incríveis.
Agradeço a todos os professores que sustentaram o meu desejo aprender.

A todos vocês, meus mais sinceros agradecimentos. Nada disso seria possível sem vocês.

RESUMO

Costa, Victor. Sistema de monitoramento veicular aplicado ao transporte coletivo. 2021. 57 f. Trabalho de Conclusão de Curso – Curso de engenharia de computação, Universidade Tecnológica Federal do Paraná. Pato Branco, 2021.

Este trabalho apresenta uma proposta de desenvolvimento de um sistema de monitoramento veicular aplicado ao transporte coletivo para a cidade de Pato Branco, a fim de desenvolver uma solução capaz de atender empresas de transporte coletivo através do desenvolvimento de um dispositivo embarcado para rastreamento de ônibus, implantação de uma infraestrutura de rede LoRaWAN para transmissão de dados do rastreador por meio da tecnologia LoRa, além de um servidor *cloud* para processamento e armazenamento de toda informação enviada pela frota de ônibus, sendo essas informações centralizadas e acessíveis as empresas de transporte a partir de uma central de monitoramento baseada na visualização de *dashboards* interativos e atualizados em tempo real. Além disso, uma forma de facilitar a vida das pessoas que dependem do ônibus como transporte urbano, a partir de um aplicativo que permite ao passageiro acompanhar os ônibus em tempo real, os pontos de parada desse ônibus e informações relacionadas a empresa de transporte, como linhas disponíveis e preço das passagens. Os resultados apresentados confirmam o funcionamento do trabalho proposto, porém, a limitação de um único servidor *cloud* e poucos materiais para implantação de uma infraestrutura de rede LoRaWAN impossibilitaram a cobertura da rede LoRaWAN em toda a cidade de Pato Branco. Contudo, esse trabalho foi desenvolvido sobre uma arquitetura e tecnologias que permitem a escalabilidade da solução, sendo assim, o conteúdo abordado pode ser utilizado como base para a implantação do sistema no mundo real.

Palavras-chave: Sistema de monitoramento veícular. LoRa. Transporte Coletivo. LoRaWAN. Cidade Inteligente. Aplicativo. Sistemas Embarcados. IoT. Servidor Cloud.

ABSTRACT

Costa, Victor. Vehicle monitoring system applied to public transport. 2021. 57 f. Trabalho de Conclusão de Curso – Curso de engenharia de computação, Universidade Tecnológica Federal do Paraná. Pato Branco, 2021.

This work presents a proposal for the development of a vehicle monitoring system applied to the public transport to the city of Pato Branco, to develop a solution capable of serving public transport companies through development of an embedded device for bus tracking, deployment of a LoRaWAN network infrastructure for data transmission tracker using LoRa technology, in addition to a cloud server for processing and storage of the entire information sent by the bus fleet, with this information being centralized and accessible to carriers from a monitoring center based on the visualization of interactive dashboards and updated in real time. Finally, a way of facilitate the life of those who depend on the bus as a means of urban transport, through an application that allows passengers to follow buses in real time, bus stops and carrier information, such as available lines and ticket prices. The results presented confirm the functioning of the proposed work, however, the limitation of a single cloud server and few materials for the implementation of a LoRaWAN network infrastructure made it impossible to cover the LoRaWAN network in the entire city of Pato Branco. However, this work was developed on an architecture and technologies that allow the scalability of the solution, so the content covered can be used as the basis for deploying the system in the real world.

Keywords: Vehicle monitoring system. LoRa. Public transportation. LoRaWAN. Smart City. App. Embedded systems. IoT. Cloud Server.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABP	*Activation by Personalization*
AMQP	*Advanced Message Queuing Protocol*
AP	*Access Point*
BW	*Bandwidth*
CSS	*Chirp Spread Spectrum*
CR	*Coding Rate*
DEPATRAN	Departamento municipal de transito de Pato Branco
DR	*Data Rate*
ESP-IDF	Espressif-IDF
FEC	*Forward Error Correction*
GSM	*Global System for Mobile*
GPS	*Global Positioning System*
HTTP	*Hyper Text Transport Protocol*
IoT	*Internet of things*
ISM	*Industrial Sientific and Medical*
LoRa	*Long Range*
LPWAN	*Low Power Wide Area Network*
MAC	*Media Access Control*
MCU	*Microcontrolador*
MQTT	*Message Queue Telemetry Transport*
NMEA	*National Marine Electronics Association*
OTAA	*Over-the-Air Activation*
OSI	*Open System Interconnection*
REST	*Representational State Transfer*

RFU	*Reserved For Future Usess*
RFID	*Radio-Frequency Identification*
RPi	Raspberry Pi
RSSI	*Received signal strength indication*
RTOS	*Real-time Operating System*
SF	*Spreading Factor*
SGBD	Sistema gerenciador de banco de dados
SNR	*Signal to Noise Ratio*
SPI	*Serial Peripheral Interface*
TTN	*The Things Network*
UART	*Universal asynchronous receiver/transmitter*

SUMÁRIO

1 INTRODUÇÃO

A mobilidade urbana pode ser descrita como a condição de deslocamento humano e de bens, diretamente associado a facilidade de se locomover em meio urbano (GOMIDE; GALINDO, 2013). Tamanha é a relevância do assunto que em 2012 entrou em vigor a Lei 12.587 que instituiu a Política Nacional de Mobilidade Urbana (PNMU) em uma de suas diretrizes visa organizar os modos de transporte, no qual as cidades com mais de 20 mil habitantes devem elaborar um Plano Municipal de Mobilidade Urbana, para o desenvolvimento de projetos que combatam as dificuldades de movimentação em centros urbanos, dentre eles o incentivo na melhoria do transporte público para redução de veículos individuais e consequentemente redução de congestionamentos e demais problemas relacionados a mobilidade urbana (BRASIL, 2012). Um estudo conduzido pelo Instituto de Pesquisa Economica Aplicada (IPEA), aponta que sistemas de ônibus urbanos e metropolitanos são a modalidade de transporte público predominante no Brasil, operando em cerca de 85% dos municípios (CARVALHO et al., 2011).

A procura por soluções inteligentes frente aos impactos de um crescimento urbano adequa aos conceitos de cidade inteligente, ou do inglês, *smart city*, que visa trazer soluções aos problemas na cidade com o amparo tecnológico, a fim de um desenvolvimento mais sustentável e que contemple toda a população (ANDRADE; GALVAO, 2016). Relacionado a *smart city*, o conceito de Internet das Coisas, do inglês, *Internet of Things* (IoT) é um conjunto de tecnologias que possibilita extrair informações do meio físico ou agir sobre ele, permitindo sua conexão a uma rede de comunicação (SINGER, 2012).Para Formisano et al. (2015), a fusão entre IoT e computação em nuvem, do inglês *Cloud computing*, fornece soluções com flexibilidade e principalmente escalabilidade aos contexto urbano, para um crescimento mais eficiente, sustentável e garantindo a qualidade de vida da população.

Conforme Moura (2004), investimentos em Sistemas Integrados de Rastreamento (SIR) estão em ascenção desde 1988, e são motivados por asseguradores e gerenciadores de risco para que empresas na área de transporte invistam em SIR em troca de benefícios e vantagem competitiva, frente aos concorrentes. Sobretudo, a motivação de investimentos em SIR, é dada por conta de dois aspectos principais:

- **Segurança:** Evitar ou identificar com mais agilidade o roubo de carga ou do próprio veículo.
- **Eficácia:** Facilidade na tomada de decisão, maximizando o controle de recursos e redução de custos operacionais.

E associado ao transporte coletivo, os benefícios do monitoramento veicular podem ser utilizados para auxiliar os passageiros e melhorar a qualidade de prestação de serviço, através da tecnologia. Com os dados coletados pelo SIR, é possível extrair informações de localização atual do veículo, previsão de horário de chegada e itinerário de uma linha de ônibus. Além de promover interesses para empresas de transporte e seus passageiros, órgãos competentes

pela fiscalização e melhoria do transporte, tráfego e condições do trânsito, podem utilizar das mesmas informações para gestão do transporte urbano (RODRIGUES; CUGNASCA; FILHO, 2009).

Este trabalho compromete-se no desenvolvimento de um sistema baseado nos conceitos de IoT, *cloud computing*, rede LoRaWAN e rastreadores GPS com tecnologia LoRa, aplicados a *smart cities* visando a melhoria da mobilidade urbana e a fim de desenvolver um serviço de telemetria para empresas na área de transporte coletivo. Aos passageiros dessa empresa, uma forma de facilitar a vida destas pessoas que dependem do ônibus como transporte urbano, com equidade, segurança e acessibilidade, conforme previsto por lei (BRASIL, 2012).

Para concretização desse trabalho, o sistema proposto consiste no desenvolvimento de um sistema de monitoramento veicular aplicado ao transporte coletivo, baseado em coletar as coordenadas geográficas do ônibus constantemente, através do Sistema de Posicionamento Global (GPS) e através da tecnologia de radiofrequência LoRa encaminha-las até um servidor na nuvem (*cloud server*) para formar uma base de dados (*database*) com a posição do veículo no decorrer do tempo. A respeito do *database*, extrair informações a cerca da posição atual do veículo, velocidade, trajeto, pontos de ônibus, distância percorrida e afins, que sirvam de auxílio na gestão de frota de ônibus por parte da empresa e também para os usuários do transporte público, através de um aplicativo *mobile* (app).

Aplicações referentes a sistemas de monitoramento veicular, geralmente empregam o uso de GPS combinado com a rede *Global System for Mobile* (GSM) para transmissão de dados, tais como os trabalhos de Chen e Liu (2010) e Jyothi e Harish (2016). Contudo, esse trabalho propõe-se a utilização da tecnologia de rede *Long Range* (LoRa), uma das soluções da *Low Power Wide Area Network* (LPWAN) disponíveis no mercado, segundo Centenaro et al. (2016).

1.1 Objetivo geral

Este trabalho se propõe a desenvolver um sistema de monitoramento veicular aplicado ao transporte coletivo, baseado em rastreadores GPS com tecnologia LoRa, e incorporado a uma rede de comunicação LoRaWAN, operando sobre um servidor na nuvem, que processa e armazena as informações disponibilizas para empresas de transporte coletivo, por meio de uma central de monitoramento e para seus passageiros, através de um aplicativo. Além disso, priorizando tecnologias e soluções *open source*, com intuito de implementar na cidade de Pato Branco uma solução eficiênte e de baixo custo.

1.2 Objetivo especifico

- Integrar o protocolo LoRaWAN em um Sistema Operacional de Tempo Real, do inglês, *Real Time Operating System* (RTOS).
- Implementar a comunicação e decodicação de um módulo GPS no microcontrolador.

- Instalação/configuração do *gateway LoRaWAN*.
- Implementar uma central de monitoramento de tempo real, baseado em painéis, do inglês, *dashboards*.
- Desenvolver um app para Android/IoS.
- Implementar uma interface para configuração do dispositivo embarcado.
- Integrar o sistema a um serviço de mensageria.

2 REFERENCIAL TEÓRICO

A fundamentação teórica deste trabalho encontra-se neste capítulo e aborda os assuntos utilizados para a elaboração do trabalho proposto. Sendo aplicada ao tema de rastreamento veicular, que está relacionado a Internet das Coisas (Seção 2.1), *Low Power Wide Area Network* (Seção 2.2), Banco de dados (Seção 2.3), Mensageria (Seção 2.5) e Microserviços (Seção 2.4) pelas características da aplicação.

2.1 Internet das coisas

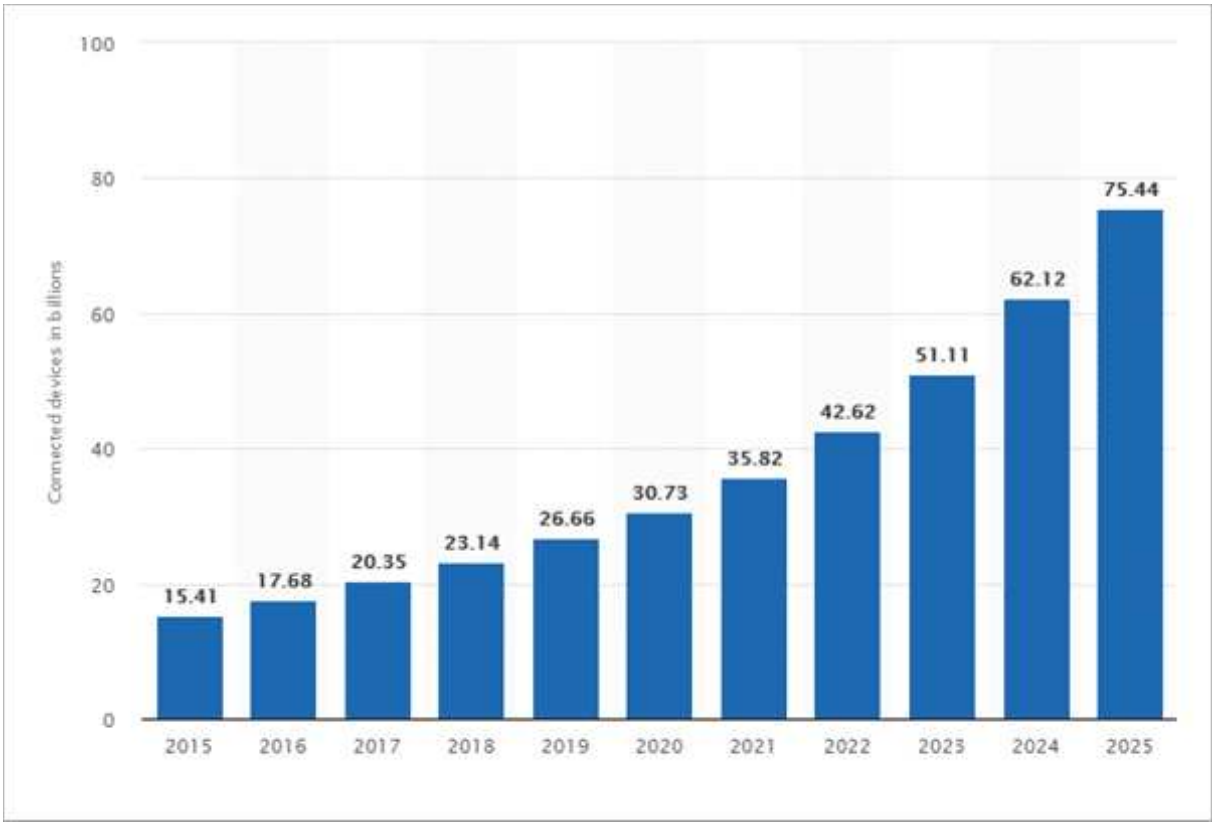
Em 1999 o cientista da computação Kevin Ashton, cofundador e diretor executivo da Auto-ID Center, pesquisava soluções que otimizassem os processos da cadeia de suprimentos da Procter & Gamble (P&G). Para isso, Kevin Ashton propôs a ligação entre a Internet e a tecnologia de identificação por radiofrequência, ou do inglês, *Radio-Frequency Identification* (RFID) para mapear os produtos e demais bens da empresa. A ideia de conectar dispositivos à internet era discutida desde 1970, na maioria das vezes chamada por *embedded internet* ou *pervasive computing*. Contudo, foi após uma apresentação a P&G, batizada com o título *Internet of Things* (IoT), que a ideia se difundiu (LUETH, 2014).

Segundo Ashton et al. (2009), a realidade da internet até então, é de que quase todos os primeiros 50 petabytes de informação na internet eram proveniêntes da ação humana, em forma de texto, fotos, vídeos e afins. E o problema dessa situação, está nas limitações dos próprios seres humanos em capturas dados sobre quaisquer que fossem as "coisas" relacionadas ao mundo real, devido a limitações de tempo, precisão e até mesmo concentração para realizar o mesmo. Contudo, se os computadores soubessem de todas as "coisa" associadas a um processo, utilizando dados coletados de maneira independente de intervenção humana, teremos a possibilidade de rastrear, quantificar e observar as "coisas" presente no ambiente. E com isso, identificar ou até mesmo predizer quando será necessário reparar, substituir ou até mesmo retirar produtos de circulação.

Nos anos subsequentes, dispositivos IoT passaram a ser cada vez mais comuns e amplamente utilizados, em eletrodomésticos, automóveis, câmeras de segurança e afins. Em 2020, cerca de 30,73 bilhões de dispositivos IoT estão em funcionamento, e a estimativa é de que 2025 o número de dispositivos conectados a internet ultrapasse a marca de 75 bilhões, conforme a Figura 1.

Para Li, Xu e Zhao (2015), o crescimento exacerbado de dispositivos IoT, deve-se ao surgimento de padronizações, que forneceram confiabilidade, compatibilidade e eficácia para operar em escala global. Além disso, criado e mantido por diversas organizações, as padronizações garantem a privacidade e segurança de seus usuários. A Tabela 1 relaciona algumas das tecnologias de comunicação populares em aplicações na área de IoT.

Figura 1 – Estimativa de dispositivos IoT de 2015 a 2025



Fonte: Statista (2016).

Tabela 1 – Tecnologias de comunicação e suas padronizações

Tecnologias de Comunicação	Padronização
ZigBee	IEEE 802.15.4
WLAN	IEEE 802.11
Bluetooth, BLE	IEEE 802.15.1
WBAN	IEEE 802.15.6

Fonte: Adaptado de Li, Xu e Zhao (2015).

Aplicações relacionadas a IoT crescem rapidamente, e adentram fazendas, cidades, industrias e até mesmo casas. Para Chen et al. (2014), as áreas de aplicações de IoT podem ser divididas em nove, conforme a Tabela 2.

Segundo Zanella et al. (2014), aplicações de IoT que são implementadas em ambiente urbano, possibilitam um melhor uso de recursos públicos e melhoria na qualidade de serviço prestada a população, enquanto reduz o custo operacional da administração pública, e isso caracteriza uma *smart city*. A gama de possibilidades e aplicações em *smart city*, despertam também o interesse do setor privado. Em 2018, cerca de 23% dos projetos empresariais na área da IoT, tiveram ênfase no segmento de *smart city*, a maior parte deles aplicados a área de transporte inteligente, com aplicações em sistemas de estacionamento, monitoramento e controle de tráfego, compartilhamento de bicicletas e faixas de ônibus inteligente (SCULLY,

Tabela 2 – Aplicações de IoT

Áreas	**Aplicações**
Indústria	Controle do processo de produção, monitoramento do ambiente industrial, rastrear a cadeia de suprimentos, monitoramento do tempo de vida de produtos, segurança na fabricação, redução de custos de energia e controle de poluição.
Agricultura inteligente	Utilização de recursos agrícolas, gestão de recursos no processo de produção agrícola, produção e monitoramento de cultivos, gestão de qualidade, segurança e rastreabilidade de produtos agrícolas.
Logística inteligente	Controle de inventário, gestão de distribuição, rastreamento e modernização do sistema logístico e e-commerce inteligente.
Transporte inteligente	Percepção de volume e estado do tráfego, guia de tráfego, posicionamento veicular e monitoramento veicular remoto.
Rede inteligente	Monitoramento de instalações elétricas, subestação inteligente e sensoreamento remoto de grandezas elétricas.
Proteção ambiental inteligente	Monitoramento de fontes de poluição, qualidade de água, qualidade do ar e rede de coleta de informações ambientais
Segurança inteligente	Monitoramento de segurança social, monitoramento de cargas perigosas, transporte de cargas químicas e alarmes preventivos.
Medicina inteligente	Controle de medicamento inteligente, gestão de hospitais, coleta e análise da fisiologia humana para parâmetros médico e cuidado médico remoto para famílias e comunidades.
Casa inteligente	Segurança residencial, controle inteligente de eletrodomésticos, economia de energia e medição inteligente.

Fonte: Traduzido e adaptado de Chen et al. (2014).

2018).

2.2 Low Power Wide Area Network

A extração automatizada de dados por meio de sensoriamento, resultou em um crescimento significativo de tecnologias para IoT, juntamente com a necessidade de cumprir alguns requisitos essenciais para o bom funcionamento de uma aplicação IoT em larga escala. Para Mekki et al. (2019), estes requisitos são:

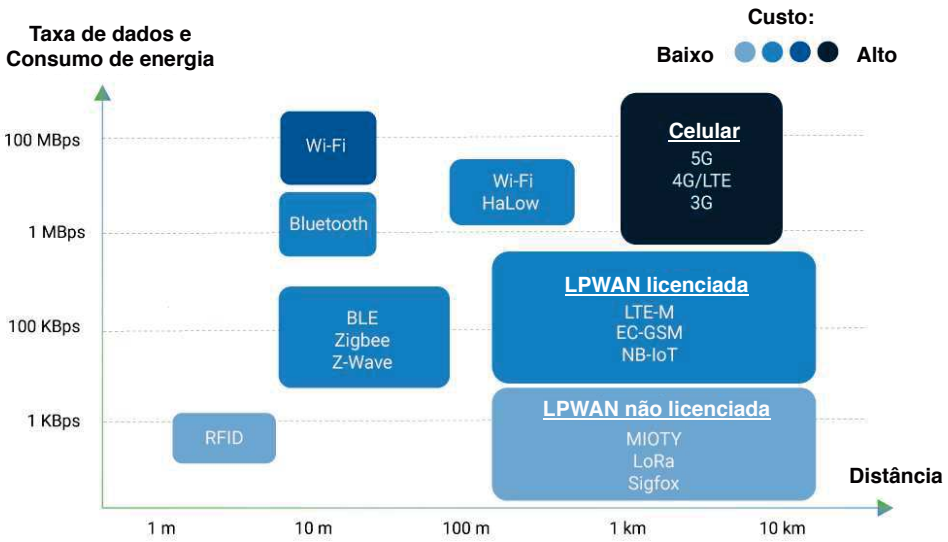
- **Longa distância:** Requisito como alternativa ao Bluetooth e ZigBee, que são tecnologias

de rádio de curta distância.

- **Baixa taxa de dados**: Aplicações em IoT, usualmente envolve sensores e atuadores, para ambos a necessidade de uma alta taxa de dados é dispensável.
- **Baixo consumo de energia:** Viabiliza a utilização de dispositivos IoT mantidos a bateria, aumentando a vida útil e tempo de carga do mesmo.
- **Baixo custo:** O custo para implementar uma rede de comunicação para IoT de ampla escala, deve ser o menor possível para que seja viável.

Com a necessidade de uma nova tecnologia de comunicação *wireless*, impulsionada pelos requisitos de mercado que representaria um crescimento massivo de aplicações em IoT, surgiu o conceito de LPWAN, acrônimo de *Low Power Wide Area Network*, e logo em seguida, algumas soluções tecnológicas foram propostas com base nessa abordagem.

Figura 2 – Tecnologias e suas caracteristicas



Fonte: Adaptado e traduzido de Behr (2018).

As métricas adequadas para quantificar o desempenho de uma rede LPWAN são a eficiência energética, escalabilidade e área de cobertura (SONG et al., 2017). A Figura 2, relaciona diferentes tecnologias aplicadas em IoT com custo de implementação, taxa de dados, consumo de energia e alcance. Dentre as quais, destacam-se nesse trabalho as tecnologias LPWAN não licenciadas LoRa e SigFox, pois a natureza de uma aplicação de monitoramento de transporte publico, requer a utilização de tecnologias de longas distâncias. Caso contrário, o custo de implementação inviabilizaria o negócio.

2.2.1 Sigfox

Sigfox é uma tecnologia proprietária que opera nas bandas *Industrial Sientific and Medical* (ISM) não licenciadas, estabelecidas em 1985 pelo órgão regulador da área de telecomunicações e radiofusão dos Estados Unidos, a *Federal Communications Commission* (FCC) (TELECO, 2020). Segundo Aernouts et al. (2018), a tecnologia permite que dispositivos se comuniquem com baixo consumo energético e baixo custo de implementação, a uma distância de 10 a 50 km, fazendo uso de uma técnica de modulação *Ultra-Narrow Bandwidth* (UNB), chamada *Differential Binary Phase Shift Keying* (DBPSK). Dispositivos Sigfox apresentam uma limitação de *uplink*, transmissão do dispositivo até a antena, e *downlink*, transmissão da antena até a dispositivo, e por consequência da UNB a taxa máxima de transferência alcançada é de 100 bps. Apresenta também frequência de operação na faixa de 868 MHz na Europa e 902 MHz no restante dos continentes.

As restrições de *uplink* e *downlink* por dispositivo, exigem que o tempo máximo de ocupação da rede por hora, seja de 1%. Como uma hora equivale a 3600 segundos, logo o tempo máximo de ocupação da rede será de 36 segundos, ou seja, o dispositivo pode transmitir dados por no máximo 36 segundos no período de uma hora. Uma mensagem Sigfox leva até 6 segundos para ser transmitido, o equivalente a 6 mensagens por hora e 144 mensagens por dia. No entanto, a sigfox separa 4 mensagens para uso de protocolo, que implica em 140 *uplinks* e 4 *downlinks* por dia, de tamanho máximo igual a 12 bytes e 8 bytes, respectivamente (SIGFOX, 2020).

2.2.2 LoRa

LoRa, acrônimo de *Long Range*, é uma tecnologia de modulação de rádio frequência proprietária da Semtech Corporation e pertence a categoria de redes LPWAN, que possui como características chave- o baixo custo de implementação, consumo energético reduzido, baixa taxa de transmissão de dados e o longo alcance. Aplicações envolvendo LoRa, usualmente alcançam 5 km em zona urbana e 15 km em zona rural, sendo assim uma tecnologia com potêncial de aplicação do campo à cidade (SEMTECH, 2019).

Segundo a Semtech (2020a), a tecnologia LoRa foi desenvolvida pela empresa Cycleo em 2009, com objetivo de realizar a comunicação sem fio aplicada a medidores de gás, eletricidade e água. Para tal, e empresa baseou-se na tecnologia de modulação *Chirp Spread Spectrum* (CSS) integrada ao *Forward Error Correction* (FEC), amplamente utilizada em sonares na indústria marítima, e também em radares para a aviação. Contudo, em 2012 a Semtech adqueriu a empresa Cycleo, assim como sua tecnologia.

A tecnologia LoRa atua na camada física, ou *physical layer* do modelo *Open System Interconnection* (OSI), um modelo utilizado na área de telecomunicações para descrever como aplicações e protocolos interagem com dispositivos conectados na rede (BRISCOE, 2000). Conforme os trabalhos de Augustin et al. (2016), Bor e Roedig (2017) e Semtech (2020a),

a modulação do sinal pode ser alterada a fim de obter resultados que favoreçam a aplicação envolvendo LoRa, sejam estes o aumento da distância de transmissão, taxa de dados ou intensidade do sinal. Para tal, é necessário modificar três paramêtros principais:

- **Spreading Factor (SF):** O SF, ou do português, fator de espalhamento, é um parâmetro que pode variar de 6 a 12. A nomenclatura utilizada para se referir aos mesmos valores é SFx, sendo x um valor entre 6 e 12. Ao aumentar o SF em um, a taxa de transmissão de bits ou *bit rate* é reduzida pela metade e, portanto, dobra a duração da transmissão e, por fim, o consumo de energia. O aumento do SF aumenta também a Relação sinal-ruído ou *Signal to Noise Ratio* (SNR), uma métrica na área de telecomunicações que relaciona a potência de um sinal sobre a potência do ruído sobre o sinal. Quanto maior o SNR, menos ruído e portanto a maior a sensibilidade de um receptor em decibéis, que consequentemente implica em um maior alcance de transmissão.
- **Coding Rating (CR):** Parâmetro relacionado ao robustez do *link* de um modem LoRa, correspondendo a camada de enlace de dados, ou *data link* do modelo OSI, camada responsável pela detecção de erros e opcionalmente, correção do mesmo. CR utiliza o FEC, uma técnica que possibilita a identificação e correção de erros, sem necessidade de retransmissão de dados. CR pode assumir um dos valores a seguir: 4/5, 4/6, 4/7 ou 4/8. Um CR maior oferece maior proteção a erros, mas aumenta o tempo de transmissão e, consequentemente o consumo de energia.
- **Bandwidth (BW):** BW é a frequência da banda de transmissão, e varia de 7,8 kHz a 500 kHz. Contudo, tipicamente é utilizado 500 kHz, 250 kHz ou 125 kHz, também conhecido respectivamente por BW500, BW250 e BW125. A escolha do BW reflete no comportamento da rede LoRa, quando maior BW, maior é o *bit rate* e portanto menos Tempo no Ar ou *LoRa packet duration*, ou também *Time on Air* (ToA), o tempo total para transmissão de um pacote de dados do transmissor ao receptor.

Para exemplificar o impacto da escolha dos parâmetros na rede LoRa, a Tabela 3 relaciona diferentes SF com BW125, 11 bytes de dados úteis, ou *payload* e comparou suas alterações sobre o *bit rate*, ToA e alcance.

Tabela 3 – Impacto do SF na rede LoRa para BW125

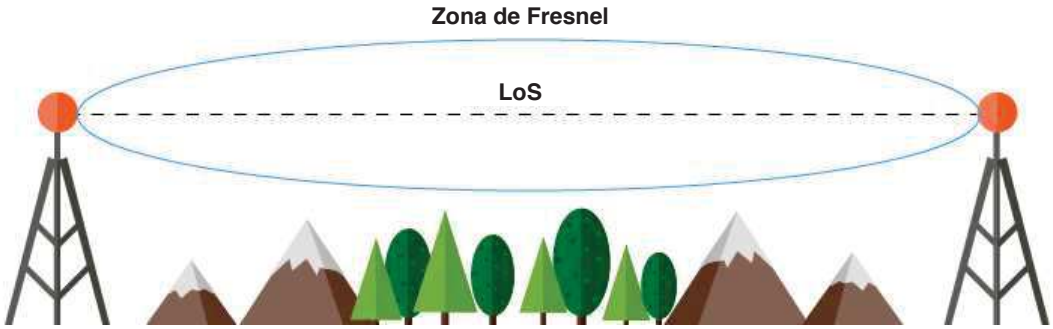
SF (BW125)	**Bit Rate**	**Alcance**	**ToA (11 bytes de payload)**
SF10	980 bps	8 km	371 ms
SF9	1760 bps	6 km	185 ms
SF8	3125 bps	4 km	103 ms
SF7	5470 bps	2 km	61 ms

Fonte: Traduzido de Semtech (2020a).

Apesar da limitação de 15 km de alcance em média para a zona rural, o LoRa possibilita alcances ainda maiores quando há Linha de Visada, ou *Line of Sight* (LoS), linha fictícia em

que dois pontos podem observar um ao outro de forma direta. Associado ao LoS, também deve ser levado em questão a zona de Fresnel, uma área tridimensional de forma elíptica encontrada em torno da LoS entre dois pontos de comunicação, na qual a obstrução completa ou parcial da zona de Fresnel representa o grau de interferência na comunicação entre transmissor e receptor (TEKON, 2020). Em julho de 2019, pesquisadores do *Grupo de Arquitectura de Computadores* da *Universidad de Zaragoza* da Espanha, lançaram 7 balões com *chipset* LoRa da Semtech que alcançaram altitudes superiores a 40 km, reproduzindo as condições da Figura 3 e obteve o recorde mundial de alcance de transmissão LoRa com o marco de 766 km (SLATS, 2019).

Figura 3 – Cenário ideal de comunicação em LoS



Fonte: Tekon (2020).

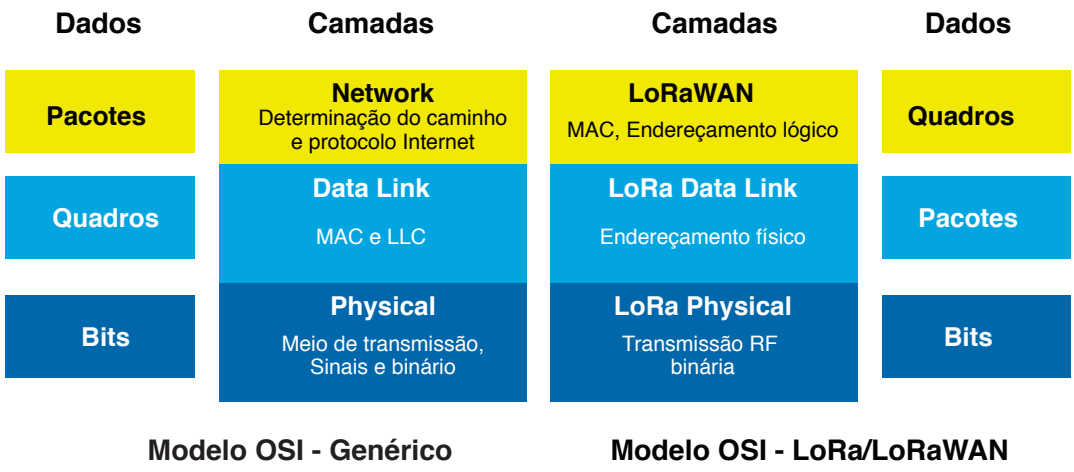
A tecnologia LoRa é responsável especificação do meio físico, e corresponde a forma que dois ou mais dispositivos LoRa irão se comunicar. Contudo, isso não permita a conexão desses dispositivos a uma rede de comunicação, e dessa lacuna foi esbelecido o protocolo de rede LoRaWAN, explicado na Subseção 2.2.3, que utiliza a tecnologia LoRa como meio físico.

2.2.3 LoRaWAN

LoRaWAN é um protocolo de rede aberto que entrega uma comunicação bidirecional entre dispositivos e tem suas especificações mantidas pela LoRa Alliance, uma organização aberta, sem fins lucrativos com objetivo de transformar o protocolo LoRaWAN em um ecosistema seguro, eficiênte e a principal tecnologia para LPWAN para aplicações de IoT (ALLIANCE, 2020). No modelo OSI, o protocolo LoRaWAN representa a camada de rede, ou *network layer*, responsável pelo endereçamento lógico dos dispositivos conectados a rede e pelo Controle de Acesso ao Meio, ou *Media Access Control* (MAC), o elemento que garante que não aconteça colisão de dados (SEMTECH, 2020a). A Figura 4 relaciona as três primeiras camadas do modelo OSI genérico ao modelo equivalente para rede LoRa/LoRaWAN.

A arquitetura de uma rede LoRaWAN é baseada na topologia estrela e composta por quatro elementos principais, o *end-device*, *gateway*, *network server* e *application server*, definidos respectivamente na Subsubseção 2.2.3.1, Subsubseção 2.2.3.2, Subsubseção 2.2.3.3

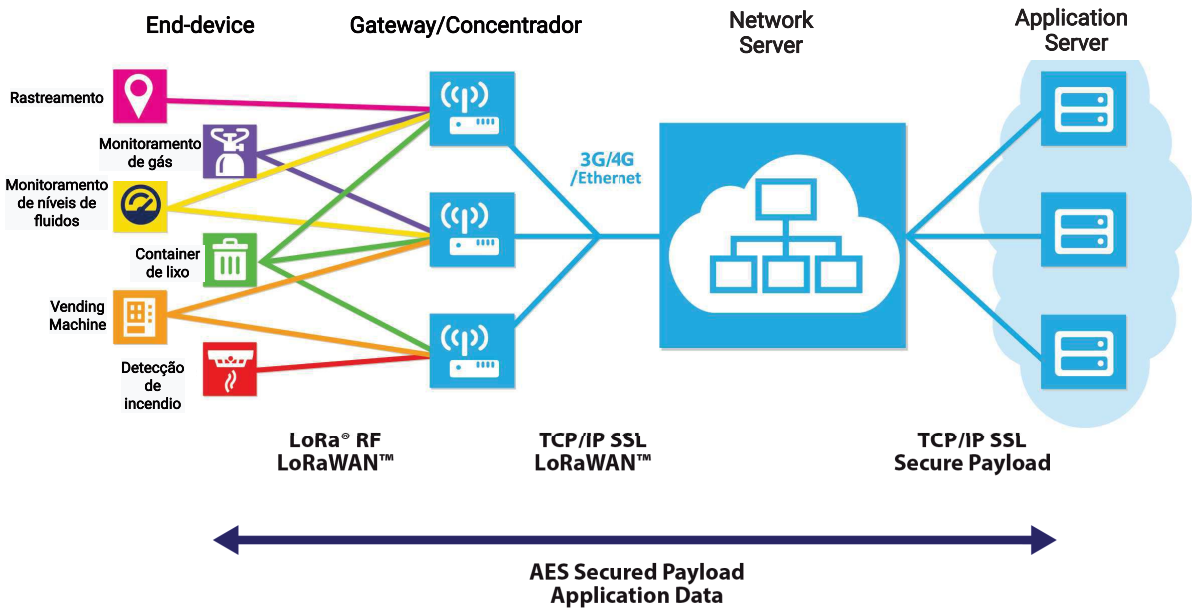
Figura 4 – LoRa/LoRaWAN aplicado ao modelo OSI



Fonte: Traduzido e adaptado de Semtech (2019).

e Subsubseção 2.2.3.4. A Figura 5 representa a arquitetura base de uma rede LoRaWAN, na qual *end-devices* transmitem informações moduladas em LoRa para todos os *gateways*, ou concentradores, que estejam em seu alcance. Posteriormente, a informação é demodulada pelo *gateway* e através do protocolo TCP/IP encaminhada a *internet*, rumo ao servidor LoRaWAN, ou *network server*, localizado na nuvem e acessível a servidores de aplicação, ou *application server*.

Figura 5 – Arquitetura base da rede LoRaWAN



Fonte: Traduzido e adaptado de Semtech (2020a).

As especificações LoRaWAN, são documentos desenvolvidos e mantidos pela LoRa Alliance. Variam de acordo com algumas regiões globais seguindo as restrições reguladas por

essas regiões, e o LoRaWAN *Regional Parameters* é o documento que regula as bandas de frequência de operação sobre as regiões globais, nomeadas de acordo com a Tabela 4.

Tabela 4 – Bandas de frequência disponíveis

Bandas de frequência (MHz)	**Nomenclatura**
863-870	EU868
902-928	US915
779-787	CN779
433	EU433
915-928	AU915
470-510	CN470
923	AS923
920-923	KR920
865-867	IN865
864-870	RU864

Fonte: Traduzido de Alliance (2018).

No Brasil, a Anatel é o órgão regulador e fiscalizador sobre serviços de telecomunicações, e permite que equipamentos de radiofrequência Sub-GHz operem nas faixas de frequência 902-907 MHz, 915-928 MHz e 433-435 MHz (ANATEL, 2004). Dentre as faixas de frequência disponíveis, o parâmetro regional AU915 equivale a faixa estabelecida de 915-928 MHz e o parâmetro regional EU433 equivale a faixa de 433-435 MHz, e portanto são os parâmetros regionais LoRaWAN aplicáveis no território brasileiro. Contudo, também é possivel utilizar o parâmetro regional US915, que opera na faixa de 902-928 Mhz, se este for limitado as faixas de frequência permitidas (ALLIANCE, 2020). Dentro de cada banda de frequência, existem canais específicos para *uplink*, que em redes LoRaWAN são mensagens transmitidas do *end-device* ao *gateway* e *downlink*, mensagens enviadas do *gateway* ao *end-device*.

Para determinar a taxa de dados em uma parâmetro regional, é necessário definir o *Data Rate* (DR), parâmetro representado usualmente pela nomenclatura DRx, onde x pode ocupar valores de 0 a 15. O DR é uma composição de diferentes SF e BW que respeitam as restrições dos parâmetros regionais. A exemplo do efeito de DR sobre a rede LoRaWAN, a Tabela 5 relaciona o DR, sua configuração em termos de SF e BW e sua taxa de bits, ou bit rate para o parâmetro regional AU915, que tem o DR7, D14 e D15 reservados para uso futuro, do inglês, *Reserved For Future Use* (RFU).

2.2.3.1 End-device

End-device, *End-node* ou *End-point* são os nomes popularmente atribuídos ao dispositivo embarcado com a tecnologia LoRa, que em uma rede LoRaWAN envia um *broadcast*, uma mensagem enviada simultaneamente a todos os receptores, que pode ser captada por um ou mais *gateways* ou concentrador, conforme ilustrado na Figura 5. A arquitetura de *hardware* genérica para desenvolvimento de um *end-device*, segundo Orange (2016) é:

Tabela 5 – DR e suas implicações para AU915

Data Rate	Configuração	Bit Rate (bps)
0	SF12 / BW125	250
1	SF11 / BW125	440
2	SF10 / BW125	980
3	SF9 / BW125	1760
4	SF8 / BW125	3125
5	SF7 / BW125	5470
6	SF8 / BW500	12500
7	RFU	
8	SF12 / BW500	980
9	SF11 / BW500	1760
10	SF10 / BW500	3900
11	SF9 / BW500	7000
12	SF8 / BW500	12500
13	SF7 / BW500	21900
14	RFU	
15	RFU	

Fonte: Traduzido de Alliance (2018).

- **Fonte de alimentação:** Geralmente é utilizado uma bateria como fonte de alimentação, recarregável ou não, pela facilidade, mobilidade e baixo consumo energético relacionado a aplicações LoRaWAN.
- **Microcontrolador (MCU):** Responsável por gerênciar todas as funcionalidades da aplicação e implementar a pilha de protocolos LoRaWAN.
- **Radio LoRa:** Composto por um transceptor LoRa, dispositivo capaz de transmitir e receber informações, circuito correspondente a antena e a própria antena.
- **Periféricos:** Dispositivos agregados, podendo ser sensores ou atuadores, como acelerômetro, relé, GPS e afins.

Para o desenvolvimento de um *end-device*, é necessário escolher um transceptor LoRa compatível aos parâmetros regionais e adequado aos requisitos de projeto. E dentre as principais alternativas de transceptores LoRa, a Semtech possui sua linha de transceptores SX127x e SX126x, a empresa Microchip com o RN2483 e RN2903 e a empresa HopeRF com a linha RFM9x. A implementação da pilha de protocolo LoRaWAN no MCU, para a linha SX127x por exemplo, requer os requisitos minimos de *hardware* de 8 kB de memória RAM, 128 kB de memória *flash*, 3 entradas de interrupção e conexão com o protocolo *Serial Peripheral Interface* (SPI) (SEMTECH, 2020a).

2.2.3.2 Gateway

Gateway ou concentrador, é um dispositivo embarcado de radiofrequência com multiplos canais e multiplos DR que tem a capacidade de fazer um varredura e detectar pacotes

enviados por *end-devices* e demodulariza-los, para então encaminha-los até o *network server* através de uma conexão Ethernet, WiFi, 3G ou 4G, conforme ilustrado na Figura 5. A arquitetura de um *gateway*, é tipicamente composta de um dispositivo baseado em Linux embarcado, integrado a *hardware* baseado no *chip gateway* da família SX12x ou SX130x da Semtech (SEMTECH, 2020a). Há diversas opções de *gateways*, de uso em ambiente interno e externo, ou *indoor* e *outdoor* respectivamente, e dentre os mais comuns, estão os modelos:

- **Comercial:** Modelos comerciais destacam-se pela sua alta performace e capacidade de aplicações em ambiente externo. Dentre os principais modelos de *outdoor gateways*, estão o Kerlink iBTS da empresa Kerlink, Cisco LoRaWAN Gateway da Cisco e o UG87 da empresa Ursalink.
- **Baseado em Raspberry Pi (RPi):** Solução amplamente encontrada por conta de seu custo benefício e facilidade de implementação, que funciona sobre Linux embarcado. É possível montar um *gateway* acoplando um módulo *HAT*, placa com funções adicionais conectada ao RPi, como por exemplo o RHF0M301 da RisingHF ou o RD43HAT da RadioEnge. Ou também, através de modelos de fabricação caseira, comunmente chamado pelo termo em inglês, *Do it Yourself* (DIY) utilizando um transceptor LoRa como *gateway* de canal único, ou também *gateway single channel*, conforme os trabalhos de Samuelson (2017) e Lie (2016).
- **Baseado em ESP32/ESP8266:** Solução de baixo custo, baseado nos MCU ESP32 e ESP8266 da empresa chinessa Espressif, que é integrado através do protocolo SPI com o transceptor RFM95W. A Espressif (2020a), possui MCUs de baixo custo e fácil conectividade com a internet, com uma ampla área de aplicações em IoT. Com esta integração de *hardware*, é possível aplicar em uma rede LoRaWAN como *end-point*. No entanto, existem projetos que utilizam desta integração para desenvolver um *gateway single channel*, conforme explorado por Westenberg (2020).
- **Open source:** Há também *gateways* comerciais baseado em projetos *open source*, que são projetos de livre distribuição, modificação e totalmente aberto ao público, como é o caso da empresa chinesa Dragino, que desenvolve também outros produtos com o intuito de promover soluções IoT *open source* (DRAGINO, 2020). A exemplo do modelo LG01-N, um *gateway single channel*, baseado em Linux embarcado que pode ser configurado por meio de acesso remoto, com *Secure Socket Shell* (SSH), ou também pelo navegador com uma interface web, tal como um roteador tradicional.

2.2.3.3 Network server

O *network server* é o elemento central da rede LoRaWAN, e carrega todos os recursos necessários para gerênciar a inteligência da rede, e isso é dado através do controle de parâmetros de rede para adaptar o sistema em condições de constante mudança. Além de estabelecer uma conexão segura baseado na ferramenta de criptografia de texto *Advanced Encryption Standard* (AES) de 128 bits, ou simplesmente AES-128, utilizado na conexão do transporte de

dados ponta a ponta, ou seja, do *end-device* ao usuário do *application server* e vice-versa. O *network server* garante a autenticidade de cada *end-device* na rede e a integridade de cada mensagem (SEMTECH, 2019). Segundo Koenen (2019) e Semtech (2019), as responsabilidades do *network server* são:

- **Consolidação de mensagens:** Multiplas cópias de um mesmo pacote podem chegar até o servidor através de diferentes *gateways*, o *network server* garante o rastreamento desses pacotes e a análise de qualidade dos pacotes recebidos.
- **Roteamento:** Decisão de melhor rota para transmissão de *downlinks*, tipicamente baseado na qualidade da conexão, utilizando como métrica o Indicador de intensidade de sinal recebido, ou *Received signal strength indication* (RSSI), SNR de pacotes anteriores e da disponibilidade do *gateway* para transmissão de *downlinks*. Também é realizado o roteamento de *uplinks* ao *application server* correspondente.
- **Controle da rede:** A qualidade da conexão entre o *end-device* e *network server*, permite que o servidor decida o melhor DR para cada um de seus *end-devices*. Contudo, para utilizar esse recurso é necessário permitir que o *network server* tenha controle sobre o DR do *end-device*, e para isso é necessário que o *end-device* tenha ativado o parâmetro ADR, acrônimo de *Adaptative Data Rate*, ou do português, Taxa de dados adaptativa.
- **Controle da downlink:** Ao enviar *downlinks* do *application server*, estes são mantidos em uma fila, até que possam ser encaminhados ao *end-device* desejado (SEMTECH, 2019).

2.2.3.4 Application server

O *application server* é o destino final dos dados enviados pelo *end-device* por mensagens de *uplink*, dentro de uma arquitetura de rede LoRaWAN genérica, conforme ilustrado na Figura 5. Segundo a Semtech (2019), o *application server* lida com a camada de aplicação da rede e provê uma aplicação *web* que permite gerenciar, visualizar e interpretar os dados dos *end-devices* e *gateways*, através de:

- **Formatação de mensagens:** Toda informação da rede gerada por *uplinks* de *end-devices* ou *gateways*, são decodificadas nesta etapa e apresentadas no formato JSON, acrônimo de *JavaScript Object Notation*, um formato de mensagens em forma texto, independente de uma linguagem de programação e leve, representado por 4 tipo de dados primitivos, strings, números, booleanos e nulos. Além de objetos e vetores, que são tipos estruturados (BRAY et al., 2014).
- **Ativação:** Conforme os trabalhos de Butun, Pereira e Gidlund (2018) e Semtech (2019), a especificação da rede LoRaWAN apresenta duas formas de ativação de *end-devices* a rede, o *Over-the-Air Activation* (OTAA) e a *Activation by Personalization* (ABP). O OTAA estabelece uma maneira segura e flexível de ativação, em que o *end-devices* solicita ao *network server* para se juntar a rede e aguarda a resposta de confirmação positiva, para então gerar chaves de segurança. O ABP apresenta-se como uma alternativa simplificada

de ativação de *end-devices* na rede, as chaves de segurança permanece inalterada após serem definidas pelo projetista e adicionadas no *firmware* do *end-device* e no *application server*. Desta forma, o OTAA é a alternativa que fornece mais segurança a rede, pois as chaves de segurança são geradas de forma autônoma e renovam-se regularmente, ao contrário do ABP.

- **Sistema de downlink:** Por meio de camada de abstração de *downlink*, permitindo enviar *payloads* do *application server* ao *gateway* e, por fim ao *end-device* através dos canais específicos para *downlink*, que variam de acordo com o parâmetro regional utilizado (SEMTECH, 2019).

2.3 Banco de dados

Conforme Elmasri et al. (2005), um banco de dados é uma coleção de dados relacionados entre si por alguma relação lógica e coerente com algum significado intrínsico, e que representa certos aspectos do mundo real- e, neste contexto, o mundo real é chamado de minimundo ou universo de discurso. A forma constituída pela abstração de dados em um minimundo é chamada de modelo de dados, um representação conceitual dos dados que utiliza uma formulação lógica entre objetos, propriedades e seus relacionados.

O processo de elaboração de um banco de dados é feito pelo armazenamento dos dados em alguma mídia sob controle de um sistema de gerenciamento de banco de dados (SGBD), um *software* que facilita a construção, definição, manipulação e compartilhamento do banco de dados entre diversos usuários e aplicações. Algumas funções são utilizadas para manipular o banco de dados, possibilitando pesquisar e recuperar um dados específico sob condições explícitadas (ELMASRI et al., 2005).

A utilização de um banco de dados é uma forma amplamente utilizada no armazenamento de dados para soluções em nuvem, e relaciona-se com o *application server* do servidor LoRaWAN, pois toda informação encaminhadas ao servidor LoRaWAN é perdida ao fechar ou reiniciar o *application server*. Disto, se faz necessário a persistência dos dados enviados ao servidor LoRaWAN, e portanto, o mesmo é vinculado a um banco de dados.

O acesso controlado ao informações armazenadas no banco de dados é feito utilizando algum sistema de mensageria, permitindo a troca de mensagens e o trafégo do informações, de ambos os lados, entre sistemas, dispositivos e usuários.

2.4 Microserviços

Conforme Balalaie, Heydarnoori e Jamshidi (2016), microserviços, do inglês *microservices*, é uma padrão de arquitetura de *software*, baseado no desenvolvimento de aplicações a partir da fragmentação de tarefas grandes e que operam de forma independente e sugiu como uma alternativa a arquitetura de sistemas monolíticos. Em um sistema baseado na arquitetura monolítica os serviços são implementados em uma única estrutura que relaciona a interface

gráfica, as regras de negócio e por fim, a camada responsável por acessar os dados armazenados, conforme ilustrado na Figura 6. A arquitetura de microserviços, baseia-se em fragmentar a estrutura de monólito em pequenas tarefas, chamados de microserviços. A Figura 6, ilustra três possibilidades distintas de modelagem de microserviços. No microserviço 1, chamado por uma ação realizada na interface gráfica, que acessa o banco de dados, podendo consultar, alterar ou remover informações do mesmo. Uma segunda possibilidade, representada pelo microserviço 2 é equivalente ao microserviço 1, porém não realiza acesso ao banco de dados. O microserviço 3, não é chamado por uma ação envolvendo a interface gráfica e, pode acesar o banco de dados e também chamar outros microserviços.

Segundo Balalaie, Heydarnoori e Jamshidi (2016), dentre as principais caracteristicas de uma estrutura de microserviços, estão:

- **Facilidade de manutenção:** Ao contrário da estrutura em monólito, as tarefas implementadas por microserviços apresentam tamanho reduzido e facilitam durante a manutenção do sistema. A exemplo disso, considere a situação na qual um microserviço é responsável por buscar informações de uma linha de ônibus no banco de dados, em caso de falha desse serviço, a região do sistema que provavelmente contêm o problema está neste microserviço, ou em demais microserviços associado diretamente a este.
- **Facilidade de implementação:** Os microserviços são independentes entre si, e portanto podem ser implementados em linguagens de programação diferentes e se relacionar com os demais.
- **Escalabilidade:** A medida que a demanda por um ou mais serviços aumenta, a arquitetura de microserviços possibilita que serviços possam ser instalados em servidores diferentes, e desta forma, reduzindo a sobrecarga de servidores.

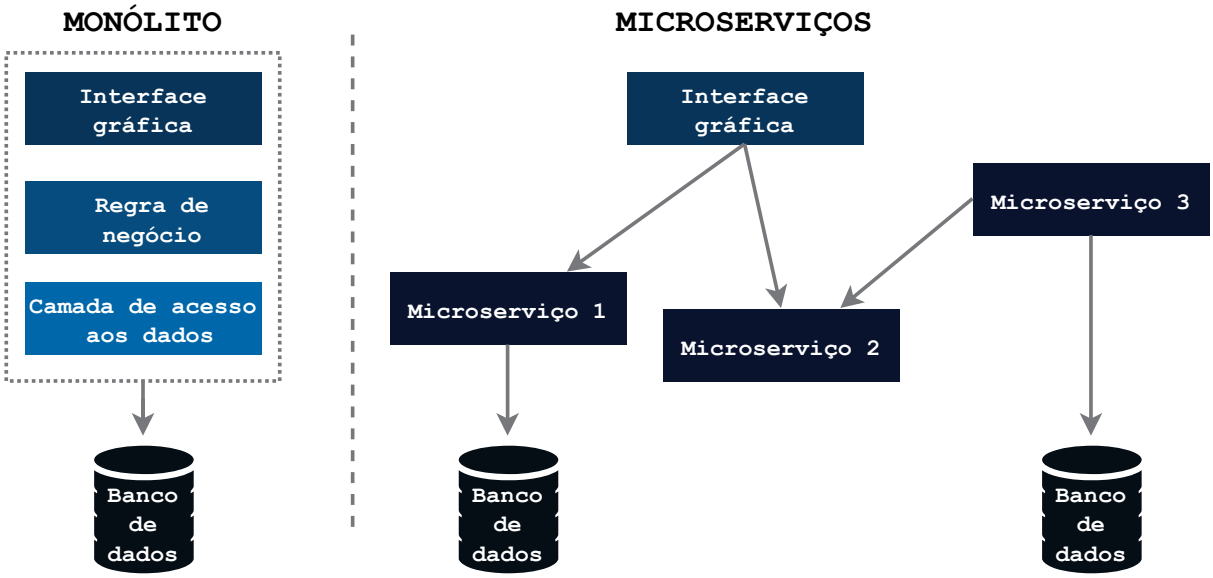
A arquitetura de microserviços tornou-se uma realidade para grandes empresas, por conta das carências de uma implementação baseada na arquitetura monolítica, conforme Evans (2016), descreve as dificuldades encontradas pela empresa que utilizava uma arquitetura monolítica, até sua transição completa para um sistema baseado em mais de 500 microserviços.

Desta forma, o acesso aos microserviços são feitos por meio de uma API, desenvolvida com o *micro framework* Flask, que encaminha as requisições ao sistema de mensageria e então direciona ao microserviço requisitado. Com isso, toda requisição à API é vinculada a um microserviço desenvolvido na linguagem Python com o *framework* Nameko, que possui suporte ao protocolo AMQP, e portanto pode ser vinculado ao servidor de mensagem RabbitMQ.

2.5 Mensageria

Para Naik (2017), ao contrário de aplicações *web* que estão padronizadas sobre o protocolo de troca de mensagens HTTP, no cenário de IoT, centenas de protocolos de troca de mensagens estão disponíveis. Consequentemente, é necessário buscar um protocolo que possa se adaptar as necessidades da aplicação e atender todos os seus requisitos. Dentre as alternativas relacionadas à protocolos de troca de mensagens, destacam-se:

Figura 6 – Arquitetura monolítica e de microserviços



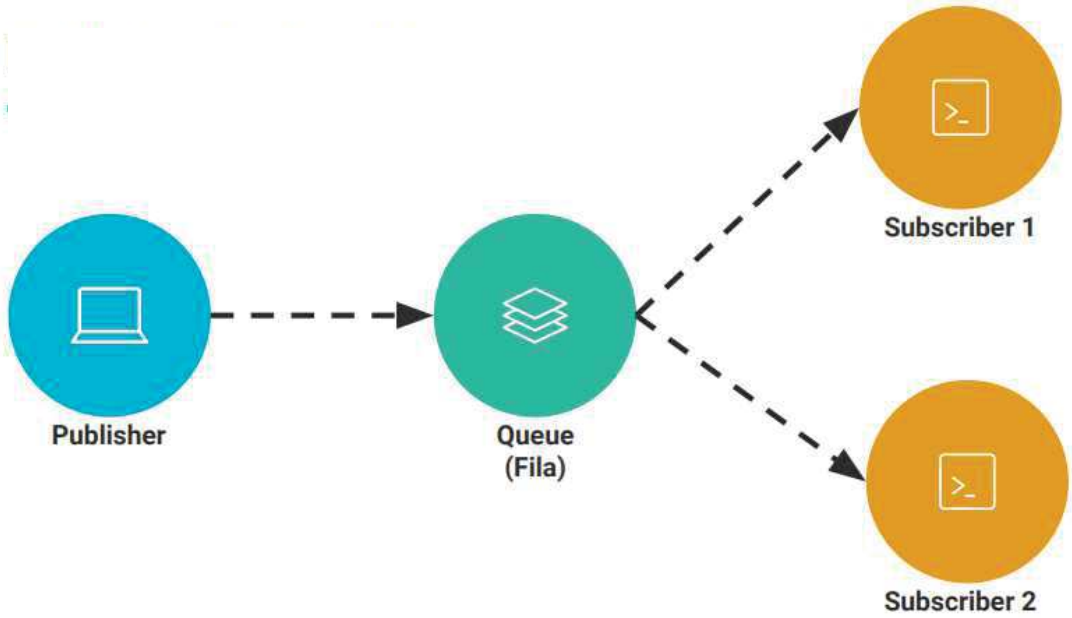
Fonte: Autoria própia.

- **MQTT:** Desenvolvido pela IBM para comunicações leves entre máquinas, comumente chamado de *Machine-to-Machine* (M2M), nasceu o *Message Queue Telemetry Transport* (MQTT). Baseado em um modelo assíncrono de publicação e assinatura, mais conhecido por *publish* e *subscribe*, sobre a rede TCP. O modelo de *publish*/*subscribe* para aplicações IoT, não necessita que o cliente solicite por atualizações do servidor, e portanto, reduzindo a largura de banda e recursos computacionais necessários. Essencialmente, o MQTT baseia-se em um servidor, chamado de *broker* que contêm tópicos, no qual cada cliente pode publicar em um *broker*, ação nomeada de *publish*, e também pode assinar um tópico, ou *subscribe*. Ao assinar um tópico, o cliente recebe automáticamente a messengem publicada, assim que houver atualização no tópico assinado (KARAGIANNIS et al., 2015).
- **HTTP:** Acrônimo de *Hyper Text Transport Protocol*, é um protocolo de mensagens utilizado predominantemente para aplicações *web* e suporta a arquitetura de mensagens e serviços *Representational State Transfer* (REST). O REST é uma arquitetura de comunicação entre computadores na *web*, que facilita a comunicação M2M, e baseia-se no modelo cliente/servidor. REST utiliza os recursos do protocolo HTTP para prover um sistema baseado em mensageria, onde todas as ações são assíncronas e realizadas sobre requisição e resposta de um servidor, comunmente chamado de *request* e *response*, respectivamente (KARAGIANNIS et al., 2015).
- **AMQP:** Um protocolo M2M, desenvolvido como protocolo de mensageria do setor coorporativo projetado para fornecer confiabilidade, provisionamento e interoperabilidade. AMQP é acrônimo de *Advanced Message Queuing Protocol*, e este protocolo suporta as arquiteturas de *request*/*response* e *publish*/*subscribe*, que oferece uma vasta gama de características, como flexibilidade na arquitetura do roteamento de mensagens, publicações

baseadas em tópicos e confiabilidade em filas de mensagens (NAIK, 2017).

A Figura 7 ilustra a estrutura básica de mensagens aplicada ao servidor de mensageria RabbitMQ, baseado no protocolo AMQP, na imagem temos um elemento que encaminha mensagens (*publisher*) para uma fila acessada por duas fontes (*subscriber*). Nessa arquitetura, por exemplo, se o *publisher* for dispositivo IoT que coleta e publica medições de temperatura na fila do RabbitMQ, esta informação será notificada aos *subscribers* dessa mesma fila. Esse se diferencia de uma arquitetura baseada em requisições HTTP, pois a parte interessada na informação não precisa solicitar para que a fonte retorne esta informação, e caso aconteça alguma falha durante a requisição, um código de erro é retornado. Contudo, se a mesma requisição fosse aplicada em um servidor de mensageria, essa requisição seria armazenada em um fila e processada quando possível para garantir que a parte solicitante teve uma resposta.

Figura 7 – Arquitetura básica de mensagens no RabbitMQ

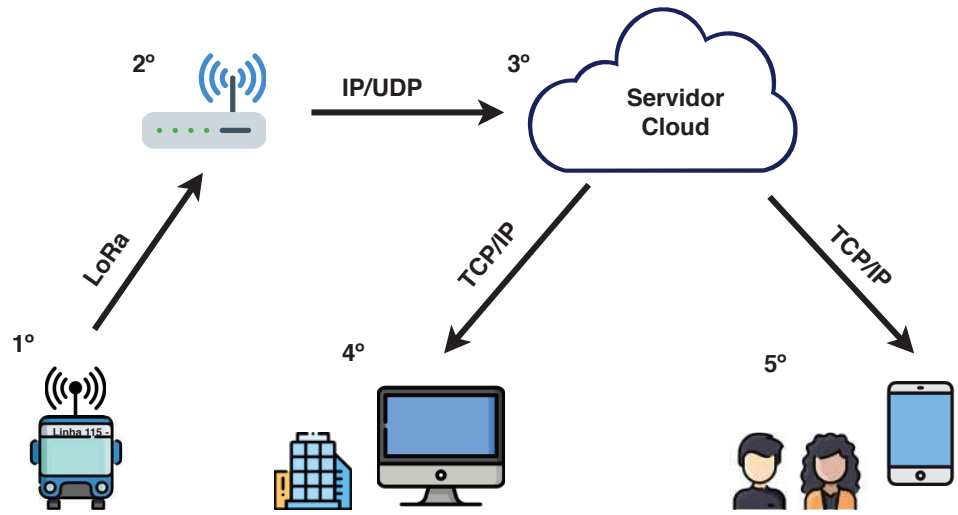


Fonte: Adaptado de Cramer (2018).

3 PROPOSTA PARA O MONITORAMENTO DE TRANSPORTE PÚBLICO

A arquitetura básica para desenvolvimento do sistema proposto, ilustrada na Figura 8, pode ser abstraída em 5 partes fundamentais que compõem um sistema de monitoramento veicular aplicado ao transporte coletivo.

Figura 8 – Arquitetura básica do sistema proposto



Fonte: Autoria própria

1. Um rastreador instalado no ônibus, que obtêm as coordenadas do veículo e transmite essa informação utilizando a tecnologia LoRa.
2. Os dados são recebidos por um *gateway*, e encaminhados até um servidor *cloud*, através de uma conexão IP/UDP.
3. O servidor *cloud* recebe as informações do *gateway*, e então as processa e armazena em um banco de dados.
4. A empresa de transporte coletivo, acessa a partir de um computador com conexão a internet, a localização de seus ônibus em tempo real.
5. Os usuários de transporte coletivo, através de um aplicativo para dispositivos móveis, podem acompanhar os ônibus monitorados em tempo real e verificar os pontos da parada desse ônibus, assim como sua previsão de chegada.

3.1 Arquitetura

Esta seção contêm a sequência de decisões de projeto para o desenvolvimento do sistema proposto, e capazes de cumprir os objetivos gerais e específicos, deste trabalho, definidos em Seção 1.1 e Seção 1.2, respectivamente. E com isso, implementar a arquitetura básico do sistema proposto, ilustrada na Figura 8.

3.1.1 Desenvolvimento do *end-device* - *Hardware*

A faixa de frequência permitida no Brasil é estipulada pela Anatel, conforme a Subseção 2.2.3, e se enquadra no parâmetro regional AU915, entre 915-928 MHz. Dentre as opções de transceptor LoRa, a linha SX127x da Semtech pode operar entre 137-1020 MHz. Analisando o *datasheet* dos transceptores SX127x, temos que a faixa de operação do modelo SX1278 está fora da especificação do AU915, enquanto o restante dos modelos encontram-se na faixa de frequência desejada. Contudo, dentre os demais, o modelo SX1277 apresenta uma faixa de SF inferior ao restante dos transceptores, e consequentemente apresenta uma sensibilidade menor, conforme explicado na Subseção 2.2.2 e relacionado na Tabela 6.

Portanto, dentro da família SX127x, temos o SX1276 ou SX1279 como opções mais viáveis para utilização nesse trabalho.

Tabela 6 – Comparação entre transceptores da família SX127x

Part Number	Frequência (MHz)	SF	BW (kHz)	Bit Rate (kbps)	Sensibilidade (dBm)
SX1276	137 - 1020	6 - 12	7,8 - 500	0,018 - 37,5	-111 a -148
SX1277	137 - 1020	6 - 9	7,8 - 500	0,011 - 37,5	-111 a -139
SX1278	137 - 525	6 - 12	7,8 - 500	0,018 - 37,5	-111 a -148
SX1279	137 - 960	6 - 12	7,8 - 500	0,018 - 37,5	-111 a -148

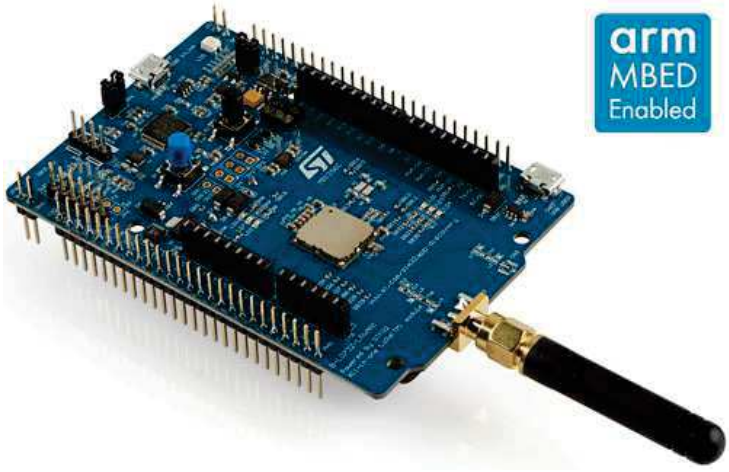
Fonte: Traduzido de Semtech (2020b)

A fim de facilitar a implementação de rede LoRaWAN e desenvolver um *end-device*, propõe-se utilizar um *kit* de desenvolvimento com transceptor LoRa integrado. Existem diversos *kits* de desenvolvimento baseados no transceptor SX1276, dentre eles:

- **B-L072Z-LRWAN1:** O *Discovery kit* LoRa, ou *kit* de descoberta da empresa STMicroelectronics, embarcado com o MCU STM32L072CZ, baseado na arquitetura Arm Cortex-M0+, e acompanha um conector SMA para antena. É vendido na faixa de 45 dólares americanos (STMICROELECTRONICS, 2020)
- **WiFi LoRa 32 v2:** *Kit* de desenvolvimento da empresa chinesa Heltec, baseado no MCU ESP32 da Espressif, sendo um *kit* de alta conectividade, com tecnologia *WiFi* e *Bluetooth* inclusa. O *kit* também possui um *display* OLED, que facilita testes de campo e acompanha um conector SMA para antena. É vendido na faixa de 19 dólares americanos (HELTEC, 2020).
- **TTgo LoRa32 v1:** *Kit* de desenvolvimento da empresa LILYgo, baseado modelo anterior, porém de menor custo, vendido na faixa de 13 dólares americanos (LILYGO, 2020).

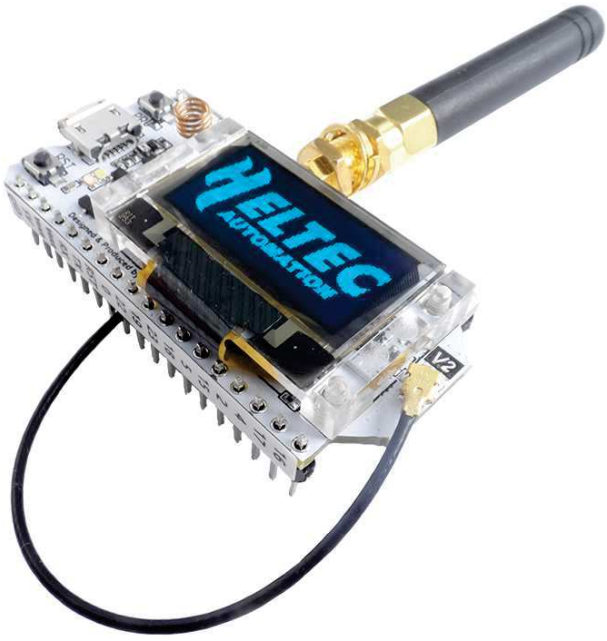
Por conta do baixo custo e alta conectivadade, propõe-se utilizar o modelo TTGO LoRa32 v1, baseado em ESP32. Com algumas alterações no *firmware* a ser desenvolvido para o *kit* TTGO, é possível fazer a portabilidade do mesmo para outros *kits* baseado em ESP32 e SX1276.

Figura 9 – *Discovery kit* LoRa B-L072Z-LRWAN1



Fonte: STMicroelectronics (2020)

Figura 10 – *Kit* de desenvolvimento WiFi LoRa 32 v2



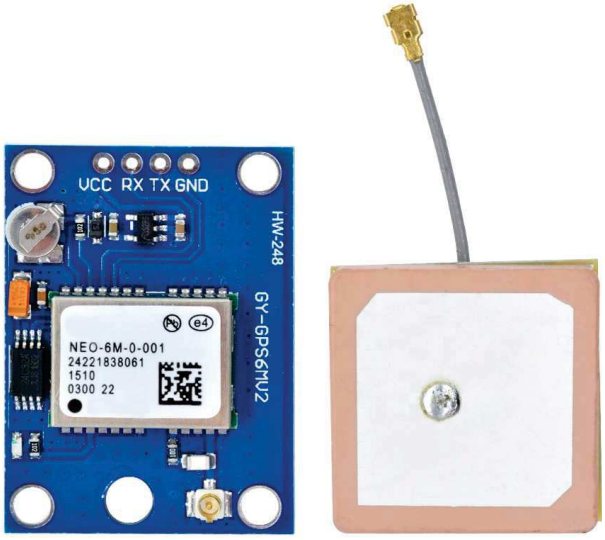
Fonte: Heltec (2020)

O modelo TTgo LoRa32 v1, não possui um módulo GPS embutido no *kit*. Portanto, propõe-se a utilização do módulo GPS GY-NEO6MV2, baseado na linha NEO-6 da empresa Ublox, um módulo GPS com conector UFL para antena e bastante acessível em custo, sendo encontrado na faixa de 3 dólares americanos (UBLOX, 2020). A integração do módulo GPS com o *kit* de desenvolvimento proposto, é dada por comunicação serial *Universal asynchronous receiver/transmitter* (UART), permitindo transmitir informação do GPS ao MCU, e vice-versa.

Figura 11 – *Kit* de desenvolvimento TTGO LoRa32 v1

Fonte: LILYgo (2020)

Figura 12 – Módulo GPS GY-NEO6MV2



Fonte: Ublox (2020)

3.1.2 Desenvolvimento do *end-device* - *Firmware*

O ESP32 popularizou-se por seu baixo custo, alta conectividade, processamento e possibilidade de ser programado utilizando o mesmo *framework* do Arduino. Contudo, o ESP32 pode ser programado por outros *frameworks*, como Lua, Micropython e Espressif-IDF. O Espressif-IDF (ESP-IDF), é o *framework* de desenvolvimento para IoT oficial da Espressif, para a linha de MCU ESP32 e ESP32-S, disponível para Windows, macOS e distribuições Linux (ESPRESSIF, 2020b).

O desenvolvimento do *firmware* do *end-device* foi desenvolvido utilizando o ESP-IDF, que posui *drivers* para utilização de periféricos da família ESP32, dentre eles a UART. Ainda,

possui uma grande variedade de bibliotecas, incluindo uma específica para decodificação de mensagens no protocolo NMEA utilizado pelos módulos GPS. Ao utilizar o driver UART e a biblioteca do protocolo NMEA pode-se obter a localização do *end-device*.

Todo o *framework* e *drivers* do ESP-IDF utilizam como base o FreeRTOS, que é líder de mercado em RTOS para MCU e processadores pequenos, um projeto *open source* que suporta mais de 40 arquiteturas de MCU. Possui portabilidade para processadores da Xtensa, utilizado na família ESP32 (FREERTOS, 2020).

A pilha de protocolo LoRaWAN para desenvolvimento de *end-devices*, é mantida no Github da Semtech e nomeada de LoRaMAC. Segundo Lora-net (2020), o LoRaMAC não possui portabilidade para processadores da Xtensa. Porém, o autor disponibiliza um *Porting Guide*, documento para realizar a portabilidade da pilha de protocolos LoRaWAN, realizando a interface com o transceptor SX1276 por uma porta serial SPI.

3.1.3 Determinar o *network* e *application server*

O *network server* mais popular e responsável por parte da disseminação da tecnologia LoRa e IoT, é o *The Things Network* (TTN), um projeto global, *open source* e mantido por entusiastas de IoT. O modo de funcionamento do TTN é baseado em uma rede LoRaWAN comunitária, que em 2020 conta com mais de 16 mil *gateways* espalhados por todo o mundo, permitindo que os usuários instalem *gateways* e *end-devices* em sua rede. Caso o usuário não tenha um *gateway*, mas esteja em uma área coberta por um ou mais *gateways* registrados no TTN, ele pode utilizar de *gateways* já instalados. O TTN não necessita de instalação, pois todo seu sistema é hospedado em nuvem, e esse modelo de funcionamento colaborativo do TTN, pode ser um fator desinteressante para a utilização no setor privado, como solução a empresa lançou o *The Things Industries* (TTI), um *network server* privado baseado no TTN (TTI, 2020).

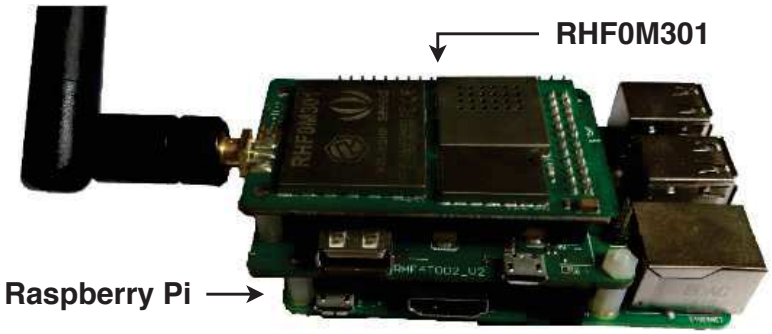
Outra alternativa popular entre as opções de *network server* é o Chirpstack, é um projeto *open source* que dispõe de componentes para montar uma rede LoRaWAN, por meio de uma arquitetura modular que torna possível a integração com infraestruturas já existentes. O Chirpstack possui seu próprio *application server* como um módulo a parte do *network server*, que permite a integração diversas tecnologias, dentre elas o RabbitMQ, HTTP, InfluxDB, MQTT e PostgreSQL. Está disponível para Windows, macOS e distribuições Linux, baseada em Debian/Ubuntu e Redhat/Centos (BROOCAR, 2020).

Este projeto tem interesse em utilizar majoritariamente ou se possível completamente, por soluções *open source*, escaláveis e sem o custeio de serviço de terceiros, com a finalidade de produzir uma solução que possa ser empregada em fins comerciais. Portanto, utilizou-se no projeto a pilha de protocolos LoRaWAN disponibilizada pelo ChirpStack.

3.1.4 Definir e instalar o *gateway*

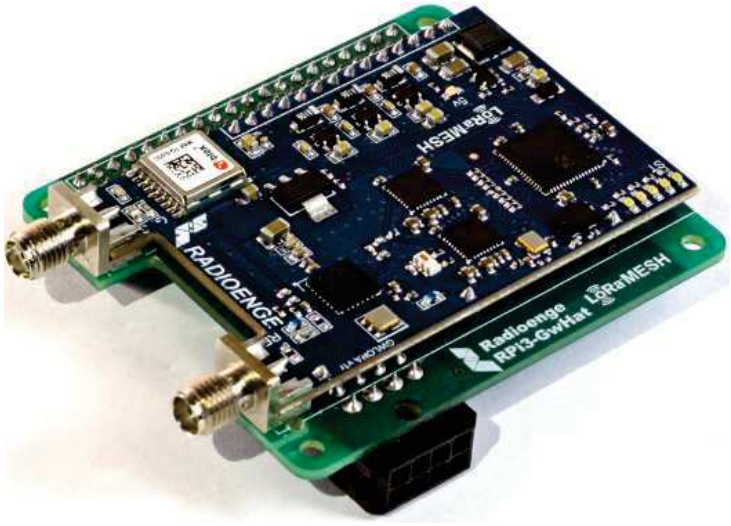
Gateways baseado em RPi, são opções interessantes considerando seu custo benefício e acessibilidade para aquisição, a exemplo dos modelos HAT baseado no *gateway* SX1301 da Semtech, o RHF0M301 e RD43HATGPS, *gateway* de 8 canais, encontrado em território nacional na faixa de 1200 reais, e portanto, propõe-se a utilização de tais módulos nesse trabalho. Esta alternativa tipicamente é executada sobre Linux embarcado. Como necessitamos do Chirpstack e esse possui um módulo para o *gateway*, é preciso utilizar uma distribuição baseada em Ubuntu/Debian ou Redhat/Centos.

Figura 13 – *Gateway* RHF0M301 com Raspberry Pi 3 Model B+



Fonte: Autoria própria

Figura 14 – *Gateway* RD43HATGPS



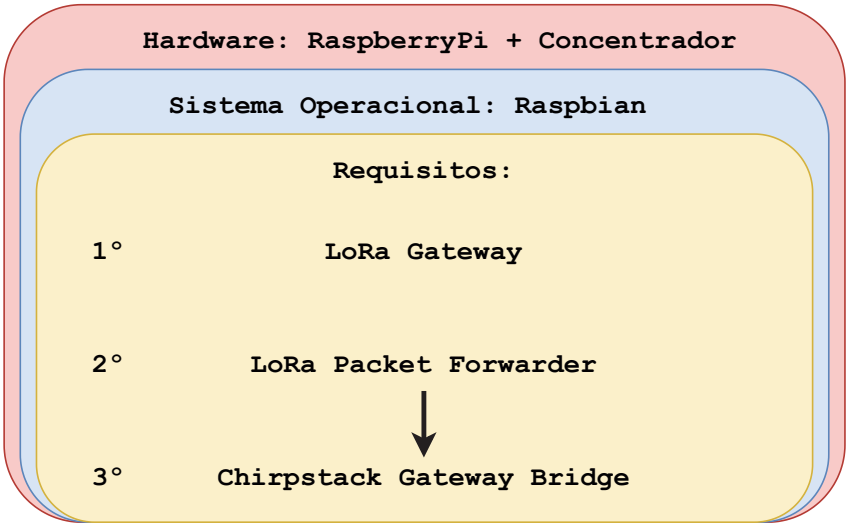
Fonte: Radioenge (2020)

Existem diversos sistemas operacionais baseado em Linux para RPi, dentre os mais populares estão Raspbian, Ubuntu Core, OpenELEC, RaspBSD e Linutop. Raspbian, acrônimo de Raspberry e Debian, é um sistema operacional baseado em Debian mais difundido para Raspberry Pi, que em maio de 2020 foi adotado pela Raspberry como o sistema operacional

oficial da empresa (RASPBERRY, 2020a). E portanto, propõe-se a utilização do Raspbian como sistema operacional.

A Figura 15 apresenta a arquitetura utilizada para a implementação do *gateway*, que utiliza uma RPi 3 Model B+ com o HAT RHF0M301 ou RD43HATGPS utilizando o sistema operacional Raspbian. Para que esse conjunto de componentes possa operar como um *gateway* LoRaWAN, primeiramente é necessário utilizar os *drivers* e uma camada de abstração de hardware, ou do inglês, *Hardware Abstraction Layer* (HAL), em *gateways* baseado em *chip* SX1301, conforme Lora-net (2017a). Posteriormente, é necessário a execução de um programa chamado *packet forwarder*, ou encaminhador de pacotes, mantido pela Lora-net (2017b). Como o nome sugere, o papel do *packet forwarder* é encaminhar os pacotes recebidos pela modulação LoRa para o *network server*, através de uma conexão IP/UDP. Além disso, o *packet forwarder* também encaminha os pacotes do *network server* ao *end-device*. Por fim, temos o *gateway bridge*, uma consequência da utilização do Chirpstack e responsável por fazer uma ponte entre o *gateway* e o *network server* da Chirpstack. Dessa forma, o *packet forwarder* encaminhará os pacotes para o *gateway bridge*, que os encaminhará para o *network server* da Chirpstack, através do protocolo MQTT.

Figura 15 – Arquitetura do *gateway*



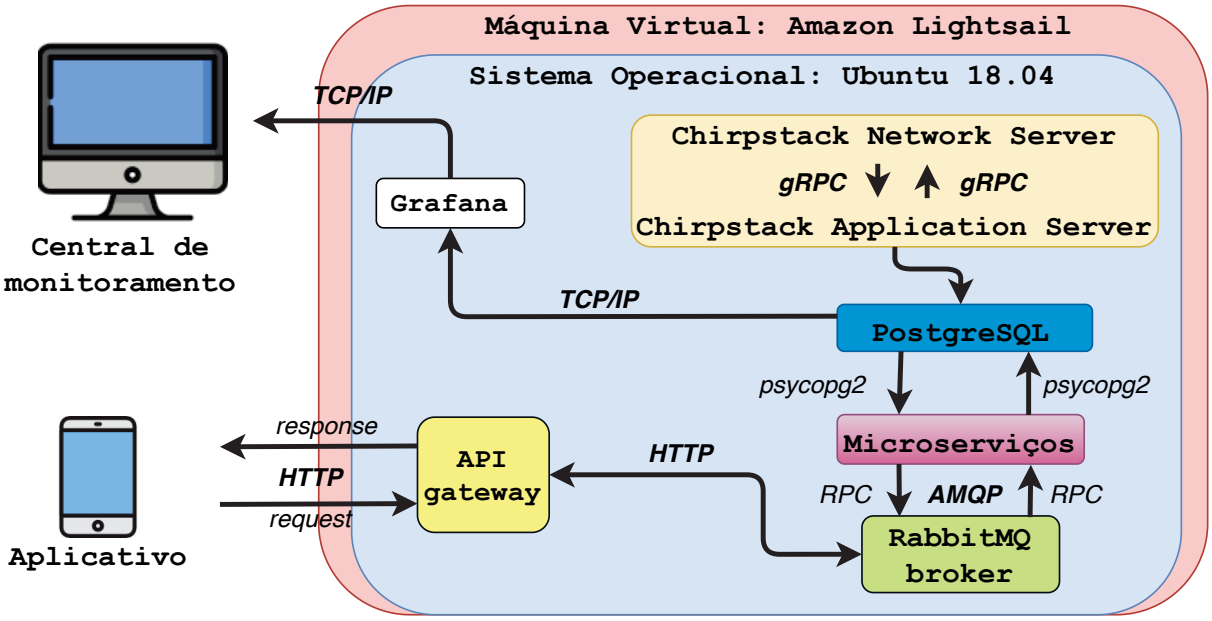
Fonte: Autoria própria

3.1.5 Servidor *Cloud*

Todo o Chirpstack pode ser instalado no próprio *gateway*, ou em outro dispositivo conectado na mesma rede. Contudo, propõe-se a utilização de um servidor *cloud* da Amazon para hospedar o *network server*, *application server*, banco de dados, sistema de mensageria e a central de monitoramento, pois dessa forma é viabilizado a utilização da central de monitoramento e do aplicativo em qualquer lugar que tenha conexão a internet, seja via WiFi, 3G/4G ou cabeada (AWS, 2020). Para o projeto do servidor *cloud*, propõe-se uma arquitetura

baseado em tecnologias que permitam implementar uma arquitetura distribuída. Contudo, a estrutura será implementada em um único servidor *cloud*, devido ao custo de manter um servidor e, pelo fato de ser um projeto em fase experimental.

Figura 16 – Proposta de arquitetura do servidor *cloud*



Fonte: Autoria própria

A Figura 16 ilustra a arquitetura proposta para a implementação do servidor *cloud*. Utilizando uma máquina virtual baseado na distribuição Ubuntu 18.04, da Amazon Lightsail, um serviço de *cloud* que oferece servidores virtuais, ideal para implementações simples, acessível pelo valor mínimo de 3,50 dólares americanos ao mês e que possibilita escalabilidade do servidor (AWS, 2020).

A partir do Chirpstack *network server*, que recebe a informação do Chirpstack *gateway bridge*, é integrado internamente com o *application server* da Chirpstack, por meio de uma conexão gRPC, conforme Chirpstack (2020). O Chirpstack *application server*, possibilita a integração com o banco de dados PostgreSQL, dessa forma fazemos a persistência dos dados enviados ao *application server*.

A central de monitoramento, utilizada para visualização centralizada de dados, foi implementada com base no Grafana, um servidor *open source* para visualização de dados em forma de gráficos, *dashboards*, tabelas e correlato. O Grafana pode ser instalado em Windows, macOS e distribuições Linux, e é acessado pelo navegador *web*. Com o Grafana, é possível conectar a mais diversos bancos de dados e associar os dados a diferentes tipos de visualização, incluindo mapas. Portanto, propõe-se a instalação do mesmo no servidor *cloud*, para que seja acessível de qualquer mídia com acesso a internet (GRAFANA, 2020).

O aplicativo, batizado de PatoBus, foi desenvolvido para dispositivos Android e iOS utilizando o *framework* Flutter, e utiliza a linguagem Dart para o desenvolvimento, que permite

a utilização de bibliotecas para acessar recursos do Google Maps, utilizado para exibir a localização dos rastreadores, pontos de parada e rota dos ônibus no mapa.

Para atender as necessidades de milhares de usuarios que utilizariam o serviço de informação de transporte público por meio do aplicativo desenvolvido para *smartphones*, propõe-se o desenvolvimento de uma estrutura baseada em um sistema de mensageria com RabbitMQ e microserviços com Nameko, acessíveis por uma API *gateway* desenvolvida com Flask, baseado nos trabalhos de Hong, Yang e Kim (2018), Neto (2019) e Rocha (2016). Dessa forma, o aplicativo realiza requisições através o protocolo HTTP, ou *resquests*, a um API *gateway*. O RabbitMQ utiliza o protocolo AMPQ, nele chegam os *requests* e *responses*, que serão enfileirados no RabbitMQ *broker* e consumido pelos microserviços vinculado a essas filas, que realização a conexão com o banco de dados através do psycopg2, um biblioteca para Python que permite realizar operações no PostgreSQL.

3.2 Material

É apresentado a seguir os materiais propostos, de *software* e *hardware*, utilizados para desenvolvimento do trabalho.

3.2.1 Software

- **Espressif-IDF:** É o *framework* de desenvolvimento para IoT oficial da Espressif, *open source* e destinado para a linha de MCU ESP32 e ESP32-S, disponível para Windows, macOS e distribuições Linux. Permite o desenvolvimento de aplicações nas linguagens C e C++ (ESPRESSIF, 2020b).
- **Chirpstack:** Um *network server* LoRaWAN, criado pelo engenheiro de *software* Orne Brocaar em 2015, é um projeto *open source* que dispõe de componentes para montar uma rede LoRaWAN, por meio de uma arquitetura modular que torna possível a integração com infraestruturas já existentes. Inclui uma interface *web* para gestão de *end-devices*, *gateways* e até mesmo outros *network servers*, com ferramentas de integração para outras tecnologias (CHIRPSTACK, 2020).
- **PostgreSQL:** Um SGBD relacional lançado no ano de 1986 como projeto da Universidade da California. É *open source*, e possui recursos adicionais que permitem a adição de outras linguagens, além da linguagem SQL (POSTGRESQL, 2020).
- **pgAdmin 4:** Ferramenta gráfica *open source*, utilizada para administrar e desenvolver no SGBD PostgreSQL e outros bancos de dados relacionais derivados do PostgreSQL. Acessada pelo navegador *web* e disponível para Linux, macOS e Windows (PGADMIN, 2020).
- **Grafana:** O projeto Grafana foi criado em 2014, por Torkel Odengaard, e tornou-se bastante popular dentre os projetos *open source* do Github. Ele permite realizar consultas a banco de dados e outras fontes, visualizar os dados, adicionar alertas sobre métricas e

gerar relatórios. Possui suporte para diversos bancos de dados, como InfluxDB, Graphite, Prometheus e MySQL, PostgreSQL (GRAFANA, 2020).

- **Flutter:** Flutter é um *kit* de ferramenta para desenvolvimento *web*, *desktop* e *mobile* de Android/IoS, gratuíto e *open source*, criado pelo Google. O Flutter possui suporte aos serviços do Google, como Google Agenda, Google Maps e demais serviços. Utiliza a linguagem de programação Dart, desenvolvida pelo Google, para desenvolver suas aplicações (FLUTTER, 2020).
- **RabbitMQ:** RabbitMQ é um serviço de mensageria *open source* baseado no protocolo AMQP, que garante a escalabilidade, confiabilidade e flexibilidade as suas aplicações. É o serviço de mensageria *open source* mais difundido e implementado do mundo, que possibilita a utilização de pequenos negócios a grandes empresas (RABBITMQ, 2020).
- **Nameko:** Nameko é um *framework open source* para desenvolvimento de microserviços em Python, que possui suporte para protocolo RPC sobre AMQP, HTTP e eventos assíncronos sobre AMQP. Como utiliza o protocolo AMQP, tipicamente é a solução utilizada para trabalhar com o desenvolvimento de microserviços integrados ao RabbitMQ (NAMEKO, 2020).
- **Flask:** Flask é um *micro framework* para desenvolvimento *web*, escrito em Python, *open source* e que provê um modelo simples para o desenvolvimento de aplicações *web* (FLASK, 2020).
- **Psycopg2:** Psycopg2 é uma biblioteca que permite a conexão com o banco de dados PostgreSQL utilizando Python. Desenvolvida majoritariamente na linguagem C, é um projeto *open source* que possibilita inserir, ler, atualizar e deletar dados no PostgreSQL (PSYCOPG2, 2021).

3.2.2 Hardware

- **TTgo LoRa32 v1:** *Kit* de desenvolvimento baseado no MCU ESP32, ilustrado na Figura 11. Apresenta uma alta conectividade, com as tecnologias embarcadas, WiFi, Blueetooth e LoRa. Suporta as faixas de frequência EU433, CN470, IN868, EU868, US915, AU915, KR920 e AS923 (LILYGO, 2020).
- **Módulo GPS GY-NEO6MV2:** Módulo GPS baseado linha de GPS 6-M da empresa Ublox, com ênfase em baixo custo e baixo consumo energético. Possui apenas 4 pinos, e apresenta facilidade para a conexão UART com o MCU, conforme ilustrado na Figura 12. Opera entre os níveis de tensão de 2,7 - 3,6 V e possui 3 metros de acurácia (UBLOX, 2020).
- **Raspberry Pi 3 Model B+:** Um pequeno computador, barato, acessível e com conectividade WiFi, Blueetooth e conexão cabeada. Acompanha um processador Broadcom BCM2837B0, de arquitetura ARMv8 64-bit, 1 GB de RAM e *clock* de 1,4 GHz,(RASPBERRY, 2020b).
- **RHF0M301:** Módulo HAT para RPi3, sendo um *Gateway* LoRaWAN embarcado com

o *chip gateway* de 8 canais da Semtech, o SX1301. Ilustrado na Figura 13, acoplado com uma RPi3 Model B+. Suporta as faixas de frequência EU868, US915 e AU915 (RISINGHF, 2020).

- **RD43HATGPS:** Similar as funcionalidades do RHF0M301, contudo esse modelo de *Gateway* da a empresa RadioEnge, possui um GPS embutido, é um produto de fabricação nacional e homologado pela Anatel. Ilustrado na Figura 14, este modelo também possui um *case* sob medida, vendido separadamente pela RadioEnge, que permite a utilização do mesmo em ambiente externo (RADIOENGE, 2020).
- **AWS Lightsail:** Um serviço de *cloud* que oferece servidores virtuais, ideal para implementações simples, acessível pelo valor mínimo de 3,50 dólares americanos ao mês e que possibilita escalabilidade do servidor. O AWS Lightsail oferece máquinas virtuais para os sistemas operacionais Windows Server, Ubuntu, Amazon Linux, Debian, FreeBSD e OpenSUSE (AWS, 2020).

4 IMPLANTAÇÃO E VALIDAÇÃO

A fim de explicar como a localização dos ônibus são obtidas, processadas, armazenadas e disponibilizadas no app e na central de monitoramento, este capítulo dedica-se a descrever e apresentar os resultados obtidos em cada uma das 5 partes fundamentais da arquitetura do sistema proposto, conforme ilustrado na Figura 8 e discutido no Capítulo 3.

4.1 Rastreador

O *end-device* foi projetado e construído sobre o *kit* de desenvolvimento TTGO LoRa32 v1, que possui o transceptor LoRa SX1276 conectado a porta SPI do microcontrolador ESP32.

4.1.1 Requisitos de projeto

Os requisitos de projeto propostos para o sistema embarcado foram definidos para atender algumas necessidades decorrêntes da aplicação de um sistema de rastreamento de ônibus, esses requisitos foram definidos com base em informações repassadas em 2020 pela empresa responsável pelo transporte urbano de Pato Branco e também pelo Departamento municipal de transito de Pato Branco (DEPATRAN) durante uma reunião de apresentação do projeto e proposta de implementação na cidade de Pato Branco. Para exemplificar e justifcar os requisitos adotados no sistema embarcado, seguem os principais pontos, sugestões e informações relatadas nesta reunião:

- **Identificadores:** Todo ônibus possui um identificador único associado a frota de ônibus representado por uma sequência numérica visível na parte traseira do veículo. Além disso, os ônibus apresentam um identificador relacionado ao seu itinerário, chamado de linha. Outros identificadores podem ser utilizados para definir um veículo, como a placa de identificação ou o chassi. Contudo, como a natureza da aplicação visa principalmente atender os usuários de transporte coletivo, será utilizado neste projeto apenas os identificadores relacionados a frota e ao itinerário, pois essas informações são mais acessíveis aos passageiros.
- **Rotação de linhas:** Uma característica presente no identificador de itinerário de um ônibus é de que esse é passível de mudança em casos de necessidade. Por exemplo, um evento na cidade pode gerar um aumento no número de passageiros que desejam ir até lá, e portanto, mais ônibus realizando a mesma linha serão necessários para atender essa demanda momentânea. Para tal, o sistema deve ser capaz de suportar a mudança de linha de um ou mais ônibus.
- **Instalação:** Atualmente todos os ônibus possuem rastreamento e o sistema de bilhetagem eletrônica fornecidos por uma empresa terceirizada, e por fatores contratuais foi sugerido adotar uma forma de instalação que não interferisse no sistema existente. Portanto,

a instalação idealizada utiliza a alimentação do *end-device* através de um cabo USB conectado ao painel do motorista e um extensor de antena para instalação de uma antena LoRa com base magnética na parte superior do veículo.

- **Configuração:** Após a instalação de um *end-device*, deve ser possível configura-lo para garantir que esse ônibus possa fazer a rotação de linhas. Conforme repassado pelo DEPATRAN, a definição do itinerário no letreiro do ônibus e a integração dessa nova informação com o sistema da empresa tercerizada é realizada por meio da interação do motorista ou cobrador do ônibus com o sistema da empresa terceirizada. Baseado no sistema atual, o poder de configurar um *end-device* foi atribuído ao motorista ou cobrador do ônibus, que poderá acessar o dispositivo protegido por senha e alterar o itinerário do ônibus.

Todos os pontos e informações levantadas foram utilizadas para a elaboração dos requisitos de projeto, com excessão dos requisitos associados a instalação do *end-device* que foram desconsiderados para o desenvolvimento desse trabalho. Dessa forma, a análise dos demais pontos foi a base para os requisitos propostos do projeto do *software* embarcado, descritos a seguir:

- **Conectividade LoRaWAN:** O *end-device* precisa ser capaz de transmitir dados utilizando o módulo transceptor LoRa SX1276 presente no *kit* de desenvolvimento, utilizar a faixa de frequência do padrão regional AU915 e ser projetado levando em consideração a escolha do DR. A interface entre o MCU e o módulo transceptor é feita através do protocolo SPI, sendo este um requisito implícito para a conectividade LoRaWAN.
- **GPS:** Outro requisito fundamental do *end-device* é a capacidade de coletar as coordenadas do veículo utilizando o módulo GPS GY-NEO6MV2. Apesar do protocolo NMEA retornar diversas informações, por exemplo a quantidade de satélites visíveis, os dados necessários para a aplicação são: latitude, longitude, altitude e velocidade. Essas informações devem ser encapsuladas em todo *payload* do pacote LoRa transmitido e ser mantida atualizada a medida que novos pacotes do protocolo NMEA são recebidos, para possibilitar o rastreamento em tempo real do veículo. A interface entre o MCU e o módulo GPS é feita através do protocolo UART, sendo este um requisito implícito para o funcionamento do GPS.
- **Display OLED:** A fim de aproveitar o recurso presente no *kit* de desenvolvimento, o *display* deve ser utilizado para depuração de testes em campo. Podendo assim, identificar o momento de transmissão de um pacote LoRa, o estado da conexão com o GPS e demais necessidades. A interface entre o MCU e o *display* OLED é feita através do protocolo I2C, que é também um requisito implícito para a utilização do recurso.
- **Conectividade WiFi:** Recurso nativo dos MCUs da família ESP32, podendo atuar em dois modos simultâneamente: *Access Point* (AP) e *station*. Em modo AP, o dispositivo cria uma rede local e permite a conexão de outros dispositivos *WiFi* na frequência de 2,4 GHz. Em contrapartida, o modo *station* permite com que o dispositivo se conecte a

outras redes locais. Ao fim, deve ser gerada uma rede local baseada nos últimos 4 digitos do endereço MAC do MCU acrescido de um identificador textual, por exemplo, para o MCU cujo endereço MAC seja **7c:64:a5:3b:f9:cf**, deve ser gerada uma rede nomeada **BUS-F9CF**.

- **Página web:** Destinada a cobradores, motoristas, gestores de frota e responsáveis em geral, tem o objetivo de viabilizar uma interface para configuração do dispositivo, caminho para a central de monitoramento e conexão à uma rede *WiFi*. A configuração do dispositivo é destinada principalmente aos motoristas e cobradores que precisam alterar o itinerário do ônibus para atender a necessidade de rotação de linhas. O caminho para a central de monitoramento é de interesse de gestores e responsáveis por tropas, e este deve direcionar para a o endereço da central de monitoramento, que solicitará suas credenciais em uma tela de validação de usuário. Por fim, deve ser possível que e o usuário conecte o dispositivo à uma rede *WiFi*, este recurso será utilizado em versões futuras do dispositivo para permitir uma atualização remota da lista de itinerários armazenadas em memória *flash* do MCU, na qual por meio de uma requisição HTTP à API que encaminhará a um microserviço e retornará a lista de itinerários atualizada.

4.1.2 Resultados

Tanto o desenvolvimento, quanto os experimentos de teste e validação do rastreador em conjunto com o sistema, foram realizados utilizando uma prototipação montada em *protoboard*, conforme a Figura 17. A montagem do protótipo é bastante simples e consiste do *kit* de desenvolvimento TTGO LoRa32 v1 associado ao módulo GPS GY-NEO6MV2 pelo mapeamento de pinos da Tabela 7. Além disso, é possível observar o funcionamento módulo GPS e a frequência da transmissão dos dados pelo transceptor LoRa através do *display* OLED, utilizado para depuração visual e uma validação parcial do funcionamento dos requisítos de conectividade LoRaWAN e GPS. Contudo, essa informação não é suficiente para garantir que o sistema está operando corretamente, uma vez que o funcionamento do GPS e a frequência da transmissão não garantem que o *payload* encaminhado via LoRaWAN foi entregue ao *gateway*.

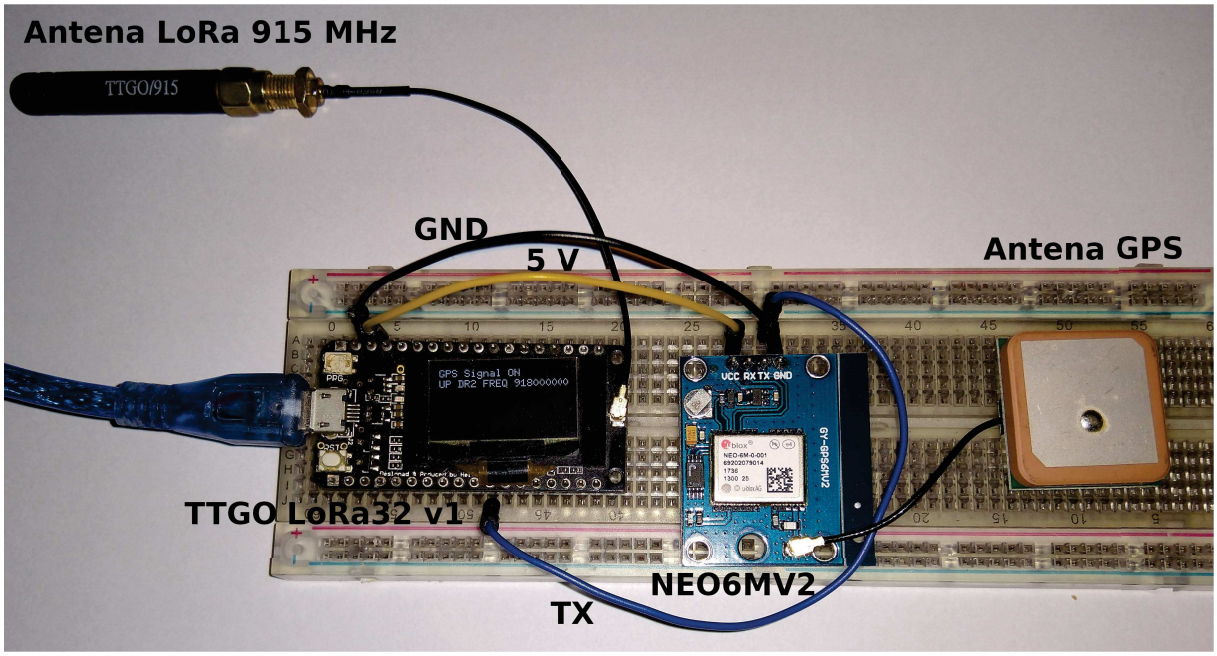
Tabela 7 – Mapeamento de pinos do protótipo

TTGO LoRa32 v1	**GY-NEO6MV2**
GPIO 23	TX
5 V	VCC
GND	GND

Fonte: Autoria própria

A Figura 18 representa o fluxograma de operação do *firmware* para transmissão de um pacote LoRa, além da organização e relação entre as tarefas do sistema, gerenciadas pelo FreeRTOS. Conforme o fluxograma, para transmitir um pacote LoRa foram criadas quatro tarefas descritas a seguir:

Figura 17 – Protótipo de rastreador montado em *protoboard*

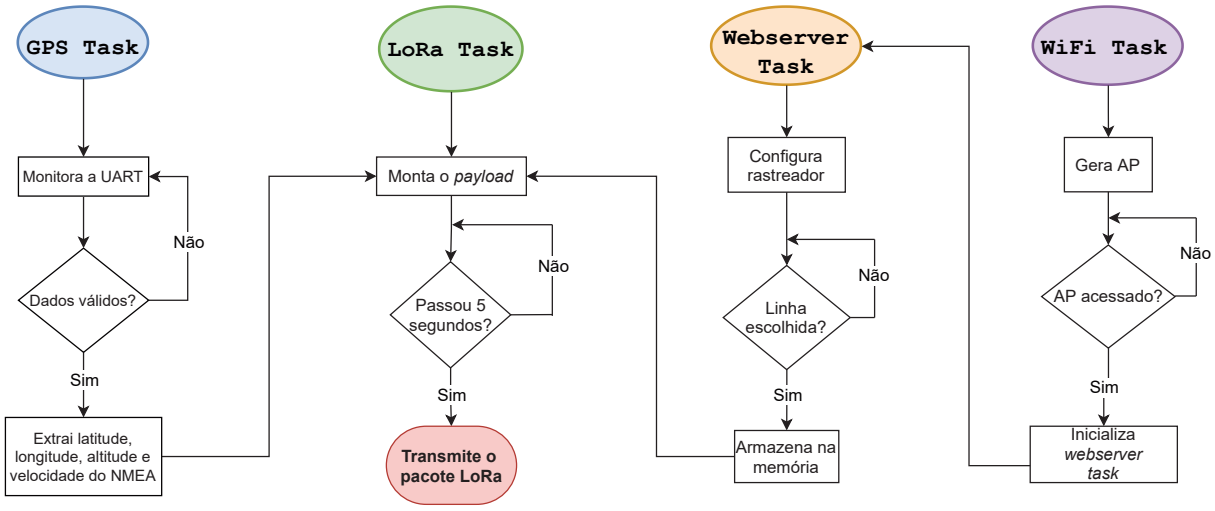


Fonte: Autoria própria.

- **GPS Task:** Tarefa responsável por monitorar os dados encaminhados pelo módulo GPS ao microcontrolador através da UART, decodificar esses dados da especificação NMEA para extrair a latitude, longitude, altitude e velocidade utilizadas na aplicação. Além disso, a instalação e configuração da UART em 9600 *baud rate* é necessária para o recebimento correto dos dados do módulo GPS NEO6MV2 e é responsabilidade dessa tarefa.
- **WiFi Task:** Responsável por toda a conectividade *WiFi* do dispositivo, incluindo a inicialização do microcontrolador em modo AP e *station*. Possibilita os recursos para que o usuário conecte-se a uma rede gerada pelo microcontrolador em modo AP, inicializando assim a tarefa responsável pelo *webserver*.
- **Webserver Task:** Inicializa o servidor HTTP para responder as requisições do *webserver*, permitindo ao usuário interagir com a página *web* e trocar informações com o microcontrolador. Com isso, o usuário pode conector o dispositivo a uma rede WiFi e configurar o rastreador para atribuir uma linha de ônibus ao mesmo, informação utilizada para formar o *payload*.
- **LoRa Task:** Inicializa a pilha de protocolo LoRaWAN e faz a interface entre o microcontrolador e o transceptor LoRa SX1276 através do protocolo SPI. Também é responsável pelo gerenciamento de pacotes, formados com base em informações recebidas pelo GPS e *Webserver task*. Desta forma, é transmitido pelo transceptor LoRa um pacote com *payload* formado respectivamente pela latitude, longitude, altitude, velocidade e a linha que representa o itinerário, conforme o exemplo: -**26.215861,-52.674160,707.00,44.0,102**.

Na Figura 20 foi utilizado a ferramenta nmap para escanear o endereço de rede 192.168.1.0/24 e listar todos os dispositivos conectados a rede local 2, pode-se validar que o

Figura 18 – Fluxograma de operação do *firmware* para transmissão de um pacote LoRa



Fonte: Autoria própria.

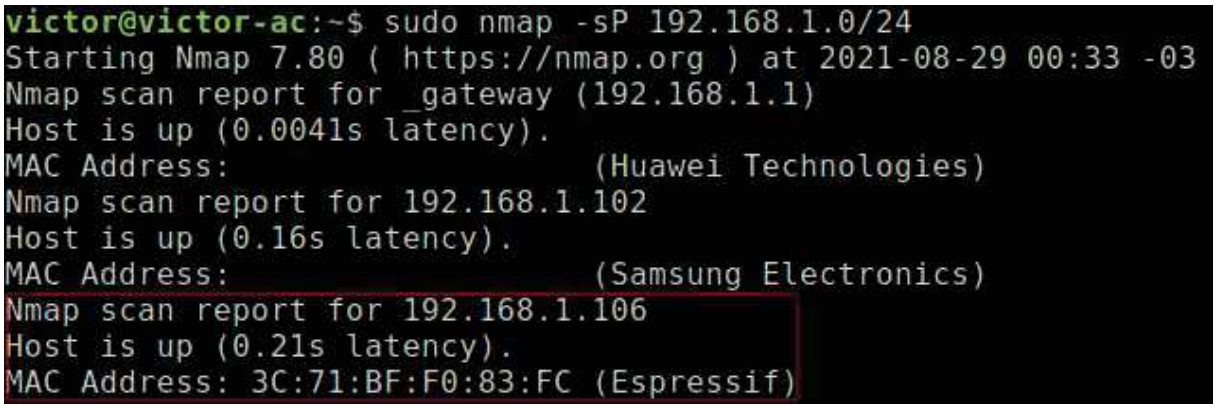
MCU foi capaz de se conectar a rede local com o endereço IP 192.168.1.106, e também foi capaz de gerar uma rede local baseada nos ultimos 4 dígitos do seu endereço MAC, de acordo com a rede local 1 da Figura 19. Dessa forma, validamos que o MCU está operando em modo *station* e AP, que por padrão cria uma rede local com o endereço de rede 192.168.4.1.

Figura 19 – Redes locais escaneadas

Fonte: Autoria própria.

A ESPIDF também permite a criação de um servidor HTTP utilizando o *framework* que possibilita a execução de um servidor *web* simples, comumente utilizado como interface de configuração de roteadores. Quando um dispositivo *WiFi* é conectado a rede local do MCU, ele pode acessar o servidor *web* ao acessar o endereço http://192.168.4.1, que retornará uma

Figura 20 – MCU conectado a rede



Fonte: Autoria própria.

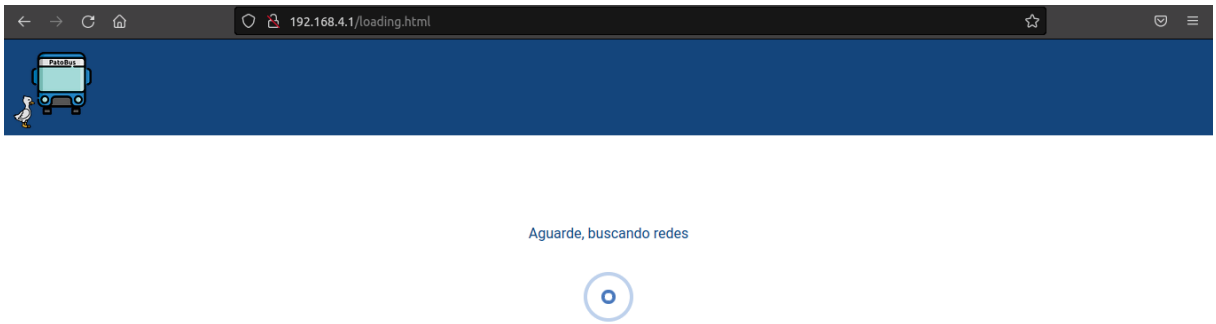
página inicial ao dispositivo *WiFi*. A Figura 21 ilustra o retorno a requisição da página inicial, na imagem é possível identificar 4 marcadores em destaque, que serão descritos a seguir:

1. Opção destinada para a configuração da rede *WiFi*, o usuário será redirecionado a tela de carregamento, ilustrada pela Figura 22. A tela de carregamento é utilizada como uma sala de espera enquanto o MCU executa uma varredura das redes locais, sem esta tela intermediária o usuário não teria como saber se o MCU está executando algo e poderia induzi-lo ao erro de clicar continuadamente na opção 1.
2. Utilizada para redirecionar o usuário a tela de configuração do rastreador, ilustrada pela Figura 24.
3. Redireciona o usuário para página de acesso da central de monitoramento.
4. Imagem presente em todas as telas, serve de também de atalho para a tela inicial.

Figura 21 – Página inicial do serivdor *web*

Fonte: Autoria própria.

Figura 22 – Tela de carregamento da página de configuração



Fonte: Autoria própria.

A Figura 23 ilustra o retorno a requisição da página configuração de rede *WiFi*, nessa imagem é possível identificar 5 campos em destaque, descritos a seguir:

1. Enquanto a tela de carregamento é executada, as redes são escaneadas e alocadas dinamicamente em memória para formar em tempo execução, uma lista contendo o SSID das redes encontradas. Desta forma, o usuário pode expandir a lista clicando nesta opção e selecionar pelo nome a rede que deseja conectar o MCU.
2. Campo para digitar a senha da rede escolhida.
3. Ao marcar este botão de verificação, uma função em javascript será chamada para substituir a lista de nomes por um campo do tipo texto, permitindo ao usuário digitar o nome da rede.
4. Após o termino da configuração deve-se pressionar este botão para armazenar as credenciais da rede na memória *flash* do microcontrolador e tentar acessar a rede selecionada. Caso o usuário entre com um tamanho de SSID ou senha fora do intervalo válido, uma notificação será exibida para instrui-lo.
5. Volta para a página inicial.

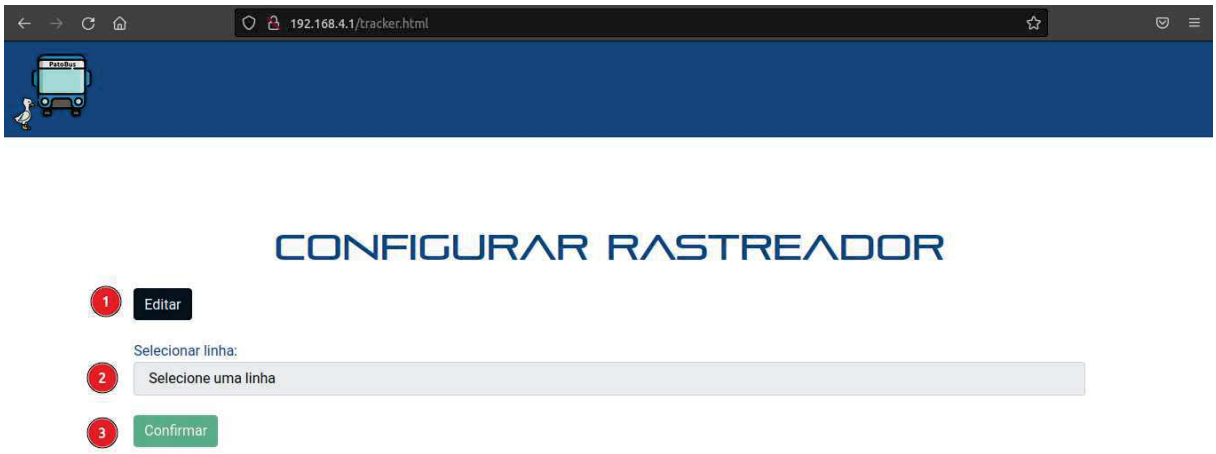
A Figura 24 expõe a página de configuração do rastreador, que possui 3 marcadores em destaque e são explicados a seguir:

1. Enquanto esta opção não for selecionada, não é possível fazer a configuração do rastreador. Quando selecionada, o texto **Editar** é substituido por **Voltar** e o usuário pode integrar com as opções 2 e 3.
2. Lista formada pelos identificador de itinerários.
3. Confirma as alterações realizadas e armazena a configuração na memória *flash* e também altera o campo de itinerário do *payload* enviado.

Figura 23 – Tela de configuração de rede *WiFi*

Fonte: Autoria própria.

Figura 24 – Tela de configuração do rastreador



Fonte: Autoria própria.

4.2 Gateway LoRaWAN

Como elemento responsável por receber os pacotes LoRa e encaminha-los ao servidor LoRaWAN pela internet, esse é um ponto de depuração importante para identificação de falhas na tecnologia LoRa e também na integração com o servidor LoRaWAN, pois o *gateway* deve respeitar alguns requisitos elencados abaixo:

1. A configuração do parâmetro regional do *gateway* deve ser a mesma da rede LoRaWAN

implementada, pois na configuração do dispositivo é necessário informar as faixas de frequência, ganho da antena, potência de transmissão, quantidade de canais e demais parâmetros. Sendo esses aplicados no *packet forwarder* levando em consideração o *hardware* utilizado, que possui uma antena de 3 dBi e 8 canais compartilhados para *uplink* e *downlink*.

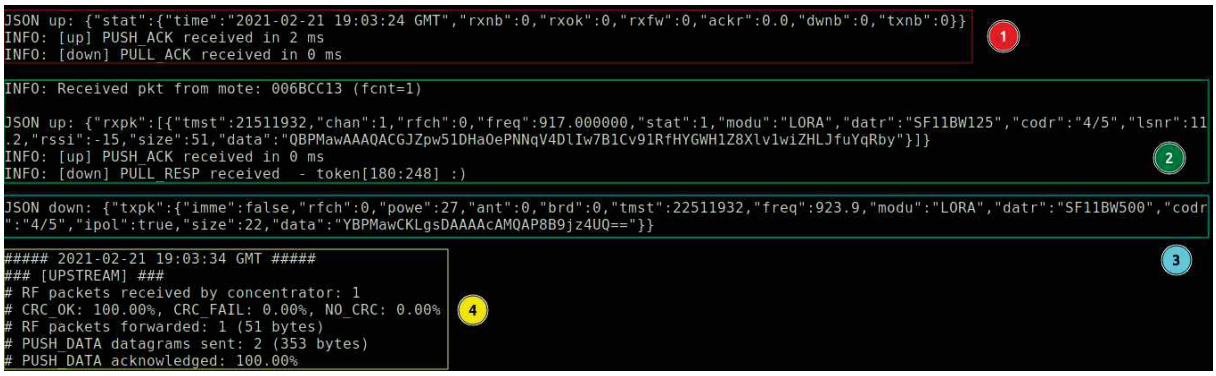
2. Cumprindo os requisítos de instação detalhados na Subseção 3.1.4, deve-se definir o endereço IP para qual os dados serão encaminhados, a porta da aplicação, intervalo de tempo de notificação do *gateway* no servidor e outros.
3. Garantir que a conexão cabeada e/ou *wireless* está funcionando e que o dispositivo está conectado a rede local. Deve-se também se certificar que a alimentação e instalação fisíca estão corretas, a exemplo da Figura 15.

Utilizando uma RPi3 e o módulo concentrador RHF0M301, é necessário a instalação do *driver*/HAL para construir um *gateway* LoRaWAN e acessar o *hardware* do módulo concentrador a partir da RPi3, configurar o *packet forwarder* e Chirpstack *gateway bridge*, conforme descrito em Costa (2021a). Para validar as partes associadas ao *gateway*, podemos utilizar o *gateway* como depurador, nos conectamos a RPi3 via SSH, acessamos a aplicação que abstrai o *hardware* para inicialização do mesmo e executamos o *packet forwarder*. Após isso, temos o *log* em tempo real da aplicação, conforme a Figura 25, e podemos validar as mensagens enumadas na imagem, conforme a explicação abaixo:

1. Representa o *uplink* da notificação de estado do *gateway* ao servidor, essa mensagem valida o envio e a confirmação do recebimento por parte do servidor. Não está relacionado a rede LoRa, mas pode garantir que a comunicação com o servidor LoRaWAN está estabelecida.
2. Demonstra o recebimento do pacote LoRa encaminhado pelo rastreador foi recebido, decodificado, encaminhado e confirmado o recebimento por parte do servidor. Nesse quadro temos algumas informações conhecidas, como RSSI, SNR e frequência, que são informações adicionadas ao pacote encaminhado servidor. O *payload* é encriptado em Base64, uma criptografia comumente utilizada para transmissão de dados do tipo texto que necessita ser convertido para o formato binário, podendo ser visualizado no campo data do JSON.
3. Mensagem de *downlink* encaminhada do *gateway* ao *end-device* após o recebimento de um *uplink*.
4. Mensagem de diagnóstico do pacote recebido pelo rastreador, informando o resultado discutido nos 3 itens acima, classificando o sucesso do recebimento, decodificação, envio e recebimento do pacote.

Por fim, podemos concluir que o *gateway* está com *uplink* e *downlink* operacionais, o que permite a recepção e transmissão de dados entre o *end-device* e o *gateway*. Além disso, pela frequência e largura de banda podemos verificar que o parâmetro regional AU915 está sendo respeitado.

Figura 25 – Registro em tempo real do *packet forwarder*



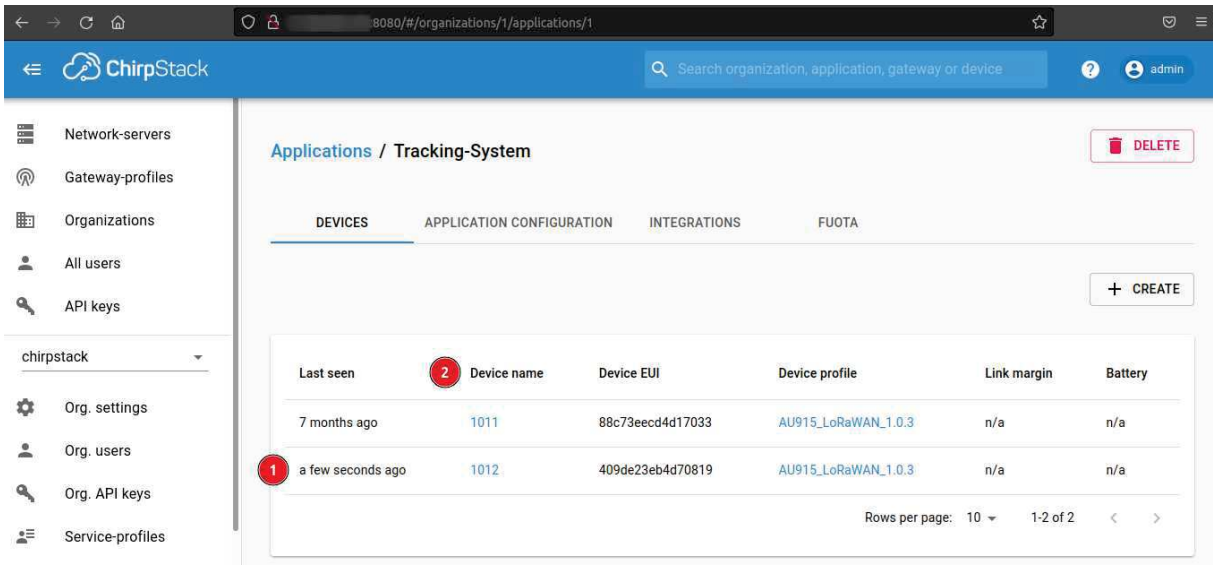
Fonte: Autoria própria.

4.3 Servidor *cloud*

Através do servidor de aplicação do Chirpstack, configuramos a rede LoRaWAN respeitando o parâmetro regional AU915 baseado em Alliance (2018) e seguindo o processo de instalação e utilização de Chirpstack (2020). Também podemos criar diferentes aplicações para a rede LoRaWAN e centraliza-las no mesmo servidor, adicionar *end-devices* e *gateways*. Sendo assim, uma ferramenta utilizada na gestão de toda a infraestrutura de rede LoRaWAN. A Figura 26 exibe a interface do servidor de aplicação, na qual podemos verificar no marcador 1 que os *end-device* nomeado como 1012 transmitiu um pacote do *gateway* a alguns segundos atrás. No marcador 2 temos o nome dos *end-devices* registrados, a padronização de nomenclatura definida aqui foi utilizada baseando-se no identificador de frota dos ônibus, desta forma é mantido a relação lógica entre os rastreadores instalados nos veículos e o restante do sistema. O servidor de aplicação armazena todos os seus dados no banco de dados postgreSQL, juntamente com o banco de dados descrito em Subseção 4.3.1 e respeitando a arquitetura proposta da Figura 16.

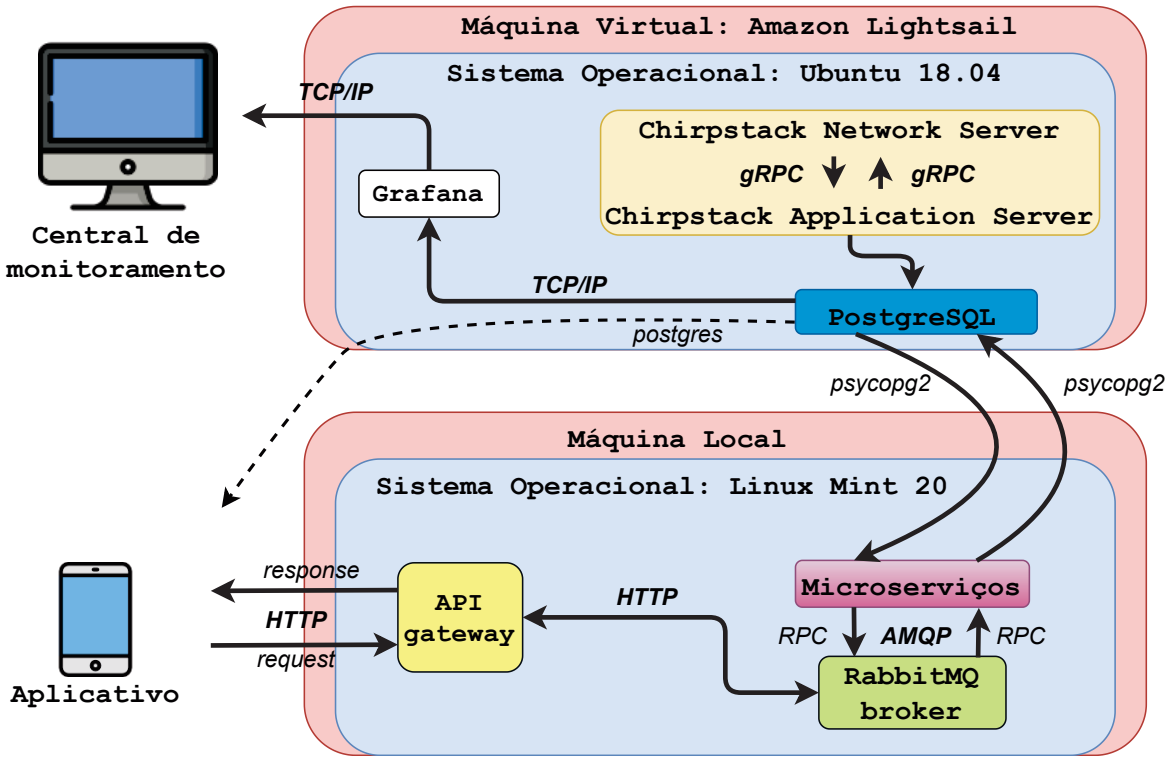
Contudo, a implementação real do sistema proposto em Figura 16 precisou ser modificada devido a falta de recursos computacionais da máquina virtual hospedado na Amazon. Ao contrário de outros planos e contratos de serviços *cloud* da Amazon que possuem um custo variável a depender da quantidade de recursos computacionais necesários para executar as aplicações instaladas, o Amazon Lightsail tem um custo fixo e uma capacidade de processamento limitada. Portanto, para garantir um desempenho satisfatório para prova de conceito e demonstração do sistema em funcionamento, conforme descrito em Subseção 3.1.5, foi necessários implementar a arquitetura distruibuída vista na Figura 27. A parte em *cloud* da solução implementada apresenta diversas aplicações e o armazenamento em banco de dados em um único servidor, esta arquitetura foi adotada em função da limitação de servidores disponíveis para o desenvolvimento desse projeto. No entanto, uma solução real baseada em *cloud* comumente utiliza-se vários servidores para distribuir as aplicações e o processamento, além de manter a redundância de dados para aumentar a resiliência da solução.

Figura 26 – Interface do servidor de aplicação do Chirpstack



Fonte: Autoria própria.

Figura 27 – Arquitetura distribuída implementada



Fonte: Autoria própria.

4.3.1 Banco de dados

O servidor de aplicação do Chirpstack possui integração para o banco de dados postgreSQL, e disto são povoadas automáticamente as tabelas geradas pela integração sempre

que um pacote enviado pelo rastreador alcança o servidor de aplicação. Contudo, as tabelas possuem apenas atributos relacionados a rede LoRaWAN e os dados encaminhados pelo rastreador, e por isso não é suficiente para representar um sistema de monitoramento veícular. Consequentemente, é necessário modelar um banco de dados capaz de representar do sistema proposto.

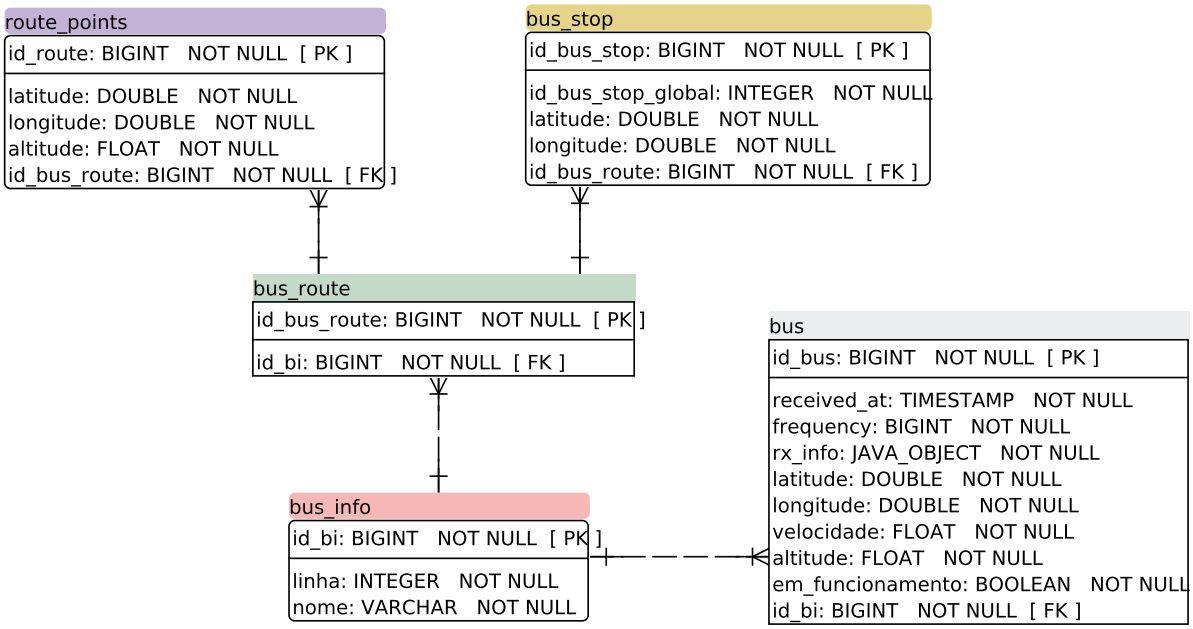
O banco de dados modelado parte de algumas premissas necesárias para utilização das funcionalidades do aplicativo, discutidas na Seção 4.4, e que foram mapeadas após os requisitos de projetos elencados na Subseção 4.1.1. O banco de dados modelado é representado por um diagrama entidade relacionamento, de acordo com a Figura 28. E as premissas que conduziram essa modelagem são as seguintes:

- **Multiplas rotas:** Uma linha de ônibus possui uma ou mais rotas, a exemplo da **Linha 102 - Av. Tupi - UTFPR/FADED** que tem duas rotas diferentes, conforme a cartilha de linhas e horários de Tupa (2018). Isso é, a rota tradicional pode ser alterada para uma rota alternativa e com isso as linhas que identificam o trajeto no mapa do aplicativo podem sofrem alterações. Portanto, a tabela **bus_route** é necessária para relacionar uma ou mais rotas à uma linha de ônibus, identificada pela tabela **bus_info**.
- **Geração de rotas:** Uma rota de ônibus é um circuito com pontos de paradas ao longo de sua extensão. Portanto, varios pontos de paradas precisam ser registrados, e um ponto de parada é representado pela tabela **bus_stop**. Dessa forma, para formar uma rota é preciso associar uma ou mais elementos da tabela **bus_stop** à tabela **bus_route**. Analogamente, isso é aplicado também a tabela **route_points**, que é formada pelo conjunto de pontos que formam uma rota, e utilizada no aplicativo para formar o desenho da rota percorrida pela linha de ônibus.
- **Rotação de linhas:** Conforme o requisito de rotação de linhas descrito em Subseção 4.1.1, a **bus_info** é associada a um ou mais ônibus, mapeados pela tabela **bus**.
- **Ônibus:** Tabela relacionada a integração do Chirpstack com o postgreSQL, sempre que o rastreador envia com sucesso um pacote para o servidor LoRaWAN, esse é armazenado na tabela de integração do Chirpstack. Sempre que um elemento é adicionado nessa tabela, uma *trigger* é disparada e chama uma função que filtra as informações do *payload*, divindo o dado bruto **-26.215861,-52.674160,707.00,44.0,102** pelo caractere separador e povoa a tabela **bus** com as coordenadas e demais paramêtros relacionando-se com as chaves da tabela **bus**.

4.3.2 Microserviços

Conforme ilustrado em Figura 27, a arquitetura distribuída implementada mantém o armazenamento dos dados no servidor *cloud*, para que o aplicativo e a central de monitoramento montada sobre o Grafana possam ser acessadas de qualquer lugar. A estrutura de microserviços associada ao *broker* do RabbitMQ e ao API *gateway* foram removidas do servidor *cloud* e migradas para uma máquina local, e portanto não podem ser acessadas fora da rede local

Figura 28 – Diagrama entidade relacionamento do banco de dados



Fonte: Autoria própria.

dessa máquina. Por esse motivo, a validação do funcionamento da estrutura de microserviços associada ao aplicativo, foi explicado, demonstrado e validado com detalhes em Costa (2021b).

4.4 Aplicativo

Desenvolvido para servir aos usuários de transporte coletivo da cidade de Pato Branco, como uma ferramenta de auxílio aos usuários de transporte coletivo, o aplicativo foi batizado de PatoBus. De acordo com a Figura 27, o aplicativo foi desenvolvido antes da definição da arquitetura do sistema, e portanto o seu desenvolvimento foi inteiramente construído sem a utilização da estrutura de microserviços. Sendo assim, as informações utilizadas para cumprir suas funcionalidades, explicadas a seguir nesta mesma seção, foi implementada utilizando a conexão diretamente com o banco de dados postgreSQL e foi realizada utilizando a biblioteca postgres para a linguagem Dart.

Sabendo disso, após a adição da estrutura de microserviços, a conexão via postgres deve ser substituída por requisições HTTP à API *gateway*. No entanto, a migração causaria muito retrabalho no aplicativo para obter o mesmo resultado visual, e portanto foi aplicado apenas na tela inicial como forma de validação de conceito da estrutura de microserviços. A identificador 2 na Figura 29 é obtido ao utilizar o aplicativo fora da rede local da máquina com a estrutura de microserviços implementada, e é exibida quando a tela inicial é carregada e a requisição à API falha. Caso contrário, se obter sucesso na requisição da informação à API, será formado um quadro informativo do preço das passagens, conforme a Figura 30. Conforme explicado em Subseção 4.3.2, isto pode ser verificado em Costa (2021b).

Para demonstração do funcionamento, a seguir será exibido e comentado cada uma das telas e funcionalidades do aplicativo. Abrindo o aplicativo, o usuário será direcionado à tela inicial, ilustrada na Figura 29. Esta tela possui um menu de opções, representado pelo indicador 1, que ao ser pressionado irá abrir um menu lateral, de acordo com a Figura 31.

Figura 29 – Tela inicial sem a estrutura de microserviços

Fonte: Autoria própria.

Exaltados na Figura 31, o menu lateral possui três icones funcionais, descritos a seguir:

1. Redireciona o usuário para a tela inicial.
2. Acessa a tela da funcionalidade **Linha**, resultando na Figura 32, que ao ser carregada irá fazer uma consulta ao banco de dados e recuperar todas as linhas de ônibus da tabela **bus_info** para formar a tela.
3. Acessa a tela da funcionalidade **Escolha sua parada**, resultando na Figura 33. Analogamente ao item anterior, este irá requisitar todos os pontos de ônibus da tabela **bus_stop**.

Figura 30 – Tela inicial com a estrutura de microserviços

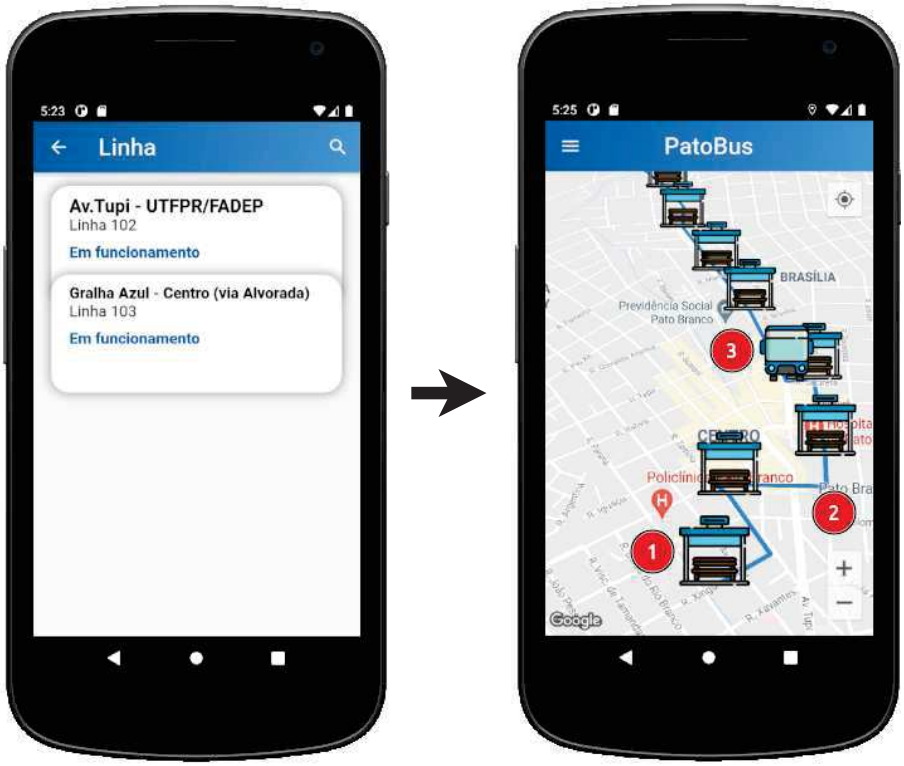
Fonte: Autoria própria.

Figura 31 – Menu lateral

Fonte: Autoria própria.

A funcionalidade **Linha** permite ao usuário escolher entre uma lista de itinerários, e ao selecionar uma delas ele será direcionado para um mapa formado pelos pontos de parada associados a esta linha, o trajeto formado pela rota do mesmo e o posicionamento em tempo real do ônibus associado a linha escolhida, a exemplo da Figura 32, que ressalta os três pontos citados.

Figura 32 – Seleção do itinerário



Fonte: Autoria própria.

A Figura 33 mostra a funcionalidade **Escolha sua parada**, desenvolvida para permitir que os usuários possam escolher o ponto de parada mais próximo do seu destino, e visualizar o trajeto de todas as linhas que passam pelo ponto de parada escolhido. Desta forma, o usuário pode conhecer as linhas de ônibus que o levão ao ponto de parada escolhido.

Figura 33 – Ponto de parada selecionado

Fonte: Autoria própria.

4.5 Central de monitoramento

Ferramenta utilizada para permitir o monitoramento de todo o sistema a partir de uma única interface. Isto é, acompanhar todos os ônibus em circulação, observar a qualidade do sinal da rede LoRaWAN, identificar a localização dos *gateways* instalados, monitorar a velocidade dos ônibus e afins. De forma geral, esta ferramenta é um concentrador de informações que podem auxíliar gestores de frota, e principalmente fornecer suporte à instalação da infraestrutura de rede LoRaWAN. Afinal, sabendo onde os *gateways* estão instalados o usuário pode, por exemplo, utilizar a ferramenta para monitorar o RSSI e SNR de um ônibus localizado em uma coordenada específica, e com isso determinar as causas da perda de sinal, e atuar para adicionar novos *gateways* que garantam a cobertura desta região.

Para demonstração da funcionalidade da central de monitoramento, o mesmo foi utilizado como ferramenta de depuração e configuração dos parâmetros de configuração do *end-device*, conforme descrito na Subseção 4.1.2. Desta forma, alterou-se o parâmetro DR no *end-device* e acompanhou-se as alterações dos sinais recebidos na central de monitoramento. Pela observação do comportamento da relação sinal-ruído, intensidade do sinal recebido e quantidade de pacotes perdidos através do RSSI, SNR e o instante de recebimento dos pacotes é possível inferir o melhor *data rate* para a aplicação.

Para iniciar o experimento, o *end-device* foi configurado para desabilitar o parâmetro

Adaptative Data Rate (ADR), pois este é recomendado para objetos estáticos, já que realiza o envio de vários pacotes com diferentes configurações de DR, e define automáticamente o DR para reduzir a perda de pacotes. Portanto, dada a natureza da aplicação, este parâmetro é mantido desabilitado. A fim de testar a ferramenta, foi aplicado diferentes *data rates* a um *end-device* posicionado a poucos metros do *gateway* e observado o comportamento dos pacotes recebidos. Esses *data rates* variaram entre DR0 à DR5, com isso notou-se pouca diferença entre DR0 e DR1, enquanto a partir de DR2 a diferença foi perceptível e para destacar a diferença entre DR1 e DR2, conforme ilustradas na Figura 34 e Figura 35.

Figura 34 – Relação entre os 10 últimos pacotes recebidos para DR1

Tempo	SNR	RSSI	Frequência	Datarate	Adaptative Datarate
08/29/21 8:57:28 pm	11.50	-19	917.8 MHz	1	0
08/29/21 8:57:23 pm	10.20	-23	916.8 MHz	1	0
08/29/21 8:57:18 pm	8.80	-17	917.2 MHz	1	0
08/29/21 8:57:13 pm	9.80	-19	917.0 MHz	1	0
08/29/21 8:57:08 pm	10.00	-17	917.6 MHz	1	0
08/29/21 8:57:03 pm	10.50	-21	918.0 MHz	1	0
08/29/21 8:56:57 pm	10.80	-17	917.6 MHz	1	0
08/29/21 8:56:52 pm	10.80	-19	917.4 MHz	1	0
08/29/21 8:56:47 pm	8.50	-17	917.2 MHz	1	0
08/29/21 8:56:42 pm	10.80	-23	916.8 MHz	1	0

Fonte: Autoria própria.

Figura 35 – Relação entre os 10 últimos pacotes recebidos para DR2

Tempo	SNR	RSSI	Frequência	Datarate	Adaptative Datarate
08/29/21 9:00:02 pm	7.80	-22	918.2 MHz	2	0
08/29/21 8:59:57 pm	10.00	-20	917.8 MHz	2	0
08/29/21 8:59:52 pm	-7.20	-59	918.2 MHz	2	0
08/29/21 8:59:47 pm	-5.20	-73	917.0 MHz	2	0
08/29/21 8:59:42 pm	-7.80	-71	917.6 MHz	2	0
08/29/21 8:59:38 pm	-7.00	-72	917.8 MHz	2	0
08/29/21 8:59:32 pm	-7.20	-72	917.8 MHz	2	0
08/29/21 8:59:28 pm	10.80	-20	917.8 MHz	2	0
08/29/21 8:59:22 pm	-5.50	-73	917.2 MHz	2	0
08/29/21 8:59:16 pm	-6.80	-73	918.2 MHz	2	0

Fonte: Autoria própria.

Com auxílio da ferramenta, avaliamos que a configuração do *end-device* em DR0 e DR1 mantiveram o SSID e SNR estáveis. Em contrapartida, *data rates* entre DR2 e DR5 gerou instabilidade na qualidade do RSSI e SNR. Também constatamos que nenhum pacote foi perdido com o *gateway* a alguns metros do *end-device*, contudo, o DR1 apresentou uma menor atenuação no sinal recebido e também um menor ruído associado aos pacotes recebidos, além de limitar o tempo no ar a aproximadamente 1 segundo para os 38 *bytes* de *payload*, mostrando-se mais adequado entre os *data rates* avaliados.

Além da tabela de relação entre pacotes, vários outros *dashboards* foram desenvolvidos para a central de monitoramento, e serão explicados conforme Costa (2021c), para validação de funcionamento e usabilidade. Esses *dashboards* podem auxiliar principalmente na implementação de uma infraestrutura de rede LoRaWAN, pois utilizando os dados de coordenada, altitude, velocidade, SSID e SNR, a tomada de decisão para distribuição dos *gateways* e configuração dos *end-devices* são amparadas por meio dessa ferramenta.

5 CONCLUSÃO

Este trabalho teve como objetivo base a implementação de um sistema de monitoramente aplicado ao transporte coletivo utilizando da rede LoRaWAN como meio de comunicação sem fio entre os ônibus rastreados e a internet. Para concretização deste trabalho, como prova de conceito foi implementado uma arquitetura proposta para o desenvolvimento do dispositivo embarcado munido com a GPS e tecnologia LoRa para comunicação com um *gateway* LoRaWAN, possibilitando que os dados sejam processados e armazenados em um banco de dados hospedado em um servidor *cloud* instalado e configurado manualmente. Além disto, foi desenvolvido formas de visualização das informações, com uma ferramenta de utilização para empresas e responsáveis pela infraestrutura LoRaWAN, e também um aplicativo com utilidade prática para a população que depende do transporte público de Pato Branco.

A implantação da arquitetura proposta foi inviabilizada por uma limitação de recursos computacionais do servidor *cloud*, e portanto precisou ser distribuída parte da carga associada a estrutura de microserviços para uma máquina local, pois apenas uma máquina na nuvem foi obtida para a execução desse trabalho. Dessa forma, os objetivos para validação de conceito foram concluídos com êxito. Com excessão da estrutura de microserviços, que é acessada apenas da rede local dessa máquina, todo o restante dos recursos podem ser utilizados a partir de uma conexão externa, isto é, central de monitoramento e aplicado sem a informação do preço das passagens.

Além disso, observou-se que a tecnologia LoRa não se comportou adequadamente para essa aplicação, em função do relevo acidentado da cidade de Pato Branco influenciar na redução da linha de visada e criação áreas sem cobertura de rede, e também pela ação do efeito doppler na transmissão de dados com *end-devices* em movimento. Para obter resultados satisfatórios nessa aplicação, seria necessário um investimento elevado na implantação da infraestrutura de rede LoRaWAN através da instalação de diversos *gateways* distribuídos em pontos estratégicos da cidade para aumentar a linha de visada e reduzir as áreas sem cobertura de rede LoRaWAN. Sendo assim, acredita-se que a utilização de uma rede LoRaWAN nessa aplicação poderia ser utilizada para complementar as falhas de cobertura de uma solução baseada na rede GSM, desta forma o custo para implantação da infraestrutura de rede LoRaWAN seria reduzido, se comparado ao custo para fornecer uma cobertura completa da cidade. Outra possibilidade está associada ao monitoramento de transporte coletivo, ou de veículos em geral em áreas urbanas, pois a tecnologia LoRa apresenta um alcance maior em áreas urbanas, que geralmente não possuí cobertura de qualidade por redes GSM.

O objetivo principal do autor ao desenvolver este trabalho, foi sua aplicabilidade no mundo real pelo seu viés prático e pela capacidade de ajudar as pessoas durante o dia-a-dia no transporte público. Contudo, apesar do projeto permanecer como prova de conceito e não ser implantado na cidade de Pato Branco, este trabalho possui uma proposta de implementação

de um sistema de monitoramento veícular, além de um exemplo de aplicação de várias de tecnologias em conjunto que podem auxiliar aqueles que tenham interesse em alguma dessas partes. Por essa razão, o projeto será documentado e preservado como código aberto para que todo conhecimento e experiência adquirida possa ser compartilhada.

Referências

AERNOUTS, M. et al. Sigfox and lorawan datasets for fingerprint localization in large urban and rural areas. **Data**, Multidisciplinary Digital Publishing Institute, v. 3, n. 2, p. 13, 2018. Citado na página 8.

ALLIANCE, L. Lorawan™ 1.0.3 regional parameters. 2018. Disponível em: <https://lora-alliance.org/sites/default/files/2018-07/lorawan_regional_parameters_v1.0.3reva_0.pdf>. Citado 3 vezes nas páginas 12, 13 e 40.

ALLIANCE, L. 2020. Disponível em: <https://lora-alliance.org/>. Citado 2 vezes nas páginas 10 e 12.

ANATEL. Regulamento sobre equipamentos de radiocomunicação de radiação restrita. 2004. Disponível em: <https://www.anatel.gov.br/hotsites/coletanea_normas/TextoIntegral/ANE/res/anatel_20040510_365_regulamento.pdf>. Citado na página 12.

ANDRADE, J. N.; GALVAO, D. C. **O conceito de smart cities aliado à mobilidade urbana**. 2016. Disponível em: <http://humanae.esuda.com.br/index.php/humanae/article/view/478>. Acesso em: 1 de setembro de 2020. Citado na página 1.

ASHTON, K. et al. That 'internet of things' thing. **RFID journal**, v. 22, n. 7, p. 97–114, 2009. Citado na página 4.

AUGUSTIN, A. et al. A study of lora: Long range & low power networks for the internet of things. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 16, n. 9, p. 1466, 2016. Citado na página 8.

AWS. Aws lightsail. 2020. Disponível em: <https://aws.amazon.com/pt/lightsail/>. Citado 3 vezes nas páginas 26, 27 e 30.

BALALAIE, A.; HEYDARNOORI, A.; JAMSHIDI, P. Microservices architecture enables devops: Migration to a cloud-native architecture. **Ieee Software**, IEEE, v. 33, n. 3, p. 42–52, 2016. Citado 2 vezes nas páginas 16 e 17.

BEHR, T. I. **Best Uses of Wireless IoT Communication Technology**. 2018. Disponível em: <https://industrytoday.com/best-uses-of-wireless-iot-communication-technology/>. Acesso em: 10 de novembro de 2020. Citado na página 7.

BOR, M.; ROEDIG, U. Lora transmission parameter selection. In: IEEE. **2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)**. [S.l.], 2017. p. 27–34. Citado na página 8.

BRASIL. **Lei Nº 12.587**. 2012. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2011-2014/2012/Lei/L12587.htm>. Acesso em: 31 de agosto de 2020. Citado 2 vezes nas páginas 1 e 2.

BRAY, T. et al. The javascript object notation (json) data interchange format. RFC 7159, DOI 10.17487/RFC7159, March 2014,< http://www. rfc-editor. org . . . , 2014. Citado na página 15.

BRISCOE, N. Understanding the osi 7-layer model. **PC Network Advisor**, v. 120, n. 2, p. 13–15, 2000. Citado na página 8.

BROOCAR, O. Chirpstack, open-source lorawan® network server stack. 2020. Disponível em: <https://www.chirpstack.io/>. Citado na página 24.

BUTUN, I.; PEREIRA, N.; GIDLUND, M. Analysis of lorawan v1. 1 security. In: **Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects**. [S.l.: s.n.], 2018. p. 1–6. Citado na página 15.

CARVALHO, C. H. R. et al. **A mobilidade urbana no Brasil**. 2011. Disponível em: <http://repositorio.ipea.gov.br/handle/11058/3494>. Acesso em: 28 de agosto de 2020. Citado na página 1.

CENTENARO, M. et al. **Long-Range Communications in Unlicensed Bands: the Rising Stars in the IoT and Smart City Scenarios**. 2016. Disponível em: <https://arxiv.org/pdf/1510.00620.pdf>. Acesso em: 26 de março de 2020. Citado na página 2.

CHEN, P.; LIU, S. Intelligent vehicle monitoring system based on gps, gsm and gis. In: IEEE. **2010 WASE International Conference on Information Engineering**. [S.l.], 2010. v. 1, p. 38–40. Citado na página 2.

CHEN, S. et al. A vision of iot: Applications, challenges, and opportunities with china perspective. **IEEE Internet of Things journal**, IEEE, v. 1, n. 4, p. 349–359, 2014. Citado 2 vezes nas páginas 5 e 6.

CHIRPSTACK. 2020. Disponível em: <https://www.chirpstack.io/>. Citado 3 vezes nas páginas 27, 28 e 40.

COSTA, V. A. Rhf0m301-chirpstack. 2021. Disponível em: <https://github.com/costa-victor/RHF0M301-Chirpstack>. Citado na página 39.

COSTA, V. A. Validação de funcionamento da estrutura de microserviços - tcc 2. 2021. Disponível em: <https://youtu.be/_NZMIn_pCD0>. Citado na página 43.

COSTA, V. A. Validação e instrução de usabilidade da central de monitoramento - tcc 2. 2021. Disponível em: <https://youtu.be/2z3BaFR8L2A>. Citado na página 49.

CRAMER, D. A. M. Filas e mensageria com rabbitmq. 2018. Disponível em: <https://www.slideshare.net/danielmarquescramer/filas-e-mensageria-com-rabbitmq>. Citado na página 19.

DRAGINO. 2020. Disponível em: <https://www.dragino.com/>. Citado na página 14.

ELMASRI, R. et al. **Sistemas de banco de dados**. [S.l.]: Pearson Addison Wesley São Paulo, 2005. v. 6. Citado na página 16.

ESPRESSIF. 2020. Disponível em: <https://www.espressif.com/>. Citado na página 14.

ESPRESSIF. Esp-idf programming guide. 2020. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>. Acesso em: 27 mar. 2020. Citado 2 vezes nas páginas 23 e 28.

EVANS, J. Mastering chaos: A netflix guide to microservices. 2016. Disponível em: <https://www.infoq.com/presentations/netflix-chaos-microservices/>. Acesso em: 30 de outubro de 2020. Citado na página 17.

FLASK. 2020. Disponível em: <https://flask.palletsprojects.com/en/1.1.x/>. Citado na página 29.

FLUTTER. 2020. Disponível em: <https://flutter.dev/>. Citado na página 29.

FORMISANO, C. et al. The advantages of iot and cloud applied to smart cities. **IEEE**, 2015. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7300835>. Acesso em: 1 de setembro de 2020. Citado na página 1.

FREERTOS. Real-time operating system for microcontrollers. 2020. Disponível em: <https://www.freertos.org/>. Citado na página 24.

GOMIDE, A. de Ávila; GALINDO, E. P. **A mobilidade urbana: uma agenda inconclusa ou o retorno daquilo que não foi**. 2013. Disponível em: <https://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-40142013000300003>. Acesso em: 30 de agosto de 2020. Citado na página 1.

GRAFANA. 2020. Disponível em: <https://grafana.com/>. Citado 2 vezes nas páginas 27 e 29.

HELTEC. Wifi lora 32 v2. 2020. Disponível em: <https://heltec.org/project/wifi-lora-32/>. Citado 2 vezes nas páginas 21 e 22.

HONG, X. J.; YANG, H. S.; KIM, Y. H. Performance analysis of restful api and rabbitmq for microservice web application. In: IEEE. **2018 International Conference on Information and Communication Technology Convergence (ICTC)**. [S.l.], 2018. p. 257–259. Citado na página 28.

JYOTHI, P.; HARISH, G. Design and implementation of real time vehicle monitoring, tracking and controlling system. In: IEEE. **2016 international conference on communication and electronics systems (ICCES)**. [S.l.], 2016. p. 1–4. Citado na página 2.

KARAGIANNIS, V. et al. A survey on application layer protocols for the internet of things. **Transaction on IoT and Cloud computing**, v. 3, n. 1, p. 11–17, 2015. Citado na página 18.

KOENEN, K. **Understanding the LoRaWAN® Architecture**. 2019. Disponível em: <https://tech-journal.semtech.com/understanding-the-lorawan-architecture>. Acesso em: 20 de novembro de 2020. Citado na página 15.

LI, S.; XU, L. D.; ZHAO, S. The internet of things: a survey. **Information Systems Frontiers**, Springer, v. 17, n. 2, p. 243–259, 2015. Citado 2 vezes nas páginas 4 e 5.

LIE, R. Build lora gateway using raspberry pi 3 model b and dragino lora shield v95. 2016. Disponível em: <https://www.hackster.io/ChrisSamuelson/lora-raspberry-pi-single-channel-gateway-cheap-d57d36>. Citado na página 14.

LILYGO. Ttgo lora32 v1. 2020. Disponível em: <http://www.lilygo.cn/prod_view.aspx?TypeId=50003&Id=1134&FId=t3:50003:3>. Citado 3 vezes nas páginas 21, 23 e 29.

LORA-NET. lora_gateway. 2017. Disponível em: <https://github.com/Lora-net/lora_gateway>. Citado na página 26.

LORA-NET. packet_forwarder. 2017. Disponível em: <https://github.com/Lora-net/packet_forwarder>. Citado na página 26.

LORA-NET. Loramac-node. 2020. Disponível em: <https://github.com/Lora-net/LoRaMac-node>. Citado na página 24.

LUETH, K. L. **Why the Internet of Things is called Internet of Things: Definition, history, disambiguation**. 2014. Disponível em: <https://iot-analytics.com/internet-of-things-definition/>. Acesso em: 07 de novembro de 2020. Citado na página 4.

MEKKI, K. et al. A comparative study of lpwan technologies for large-scale iot deployment. **ICT express**, Elsevier, v. 5, n. 1, p. 1–7, 2019. Citado na página 6.

MOURA, L. C. B. Avaliação do impacto do sistema de rastreamento de veículos na logística. **Pontifícia Universidade Católica. Rio de Janeiro**, 2004. Citado na página 1.

NAIK, N. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In: IEEE. **2017 IEEE international systems engineering symposium (ISSE)**. [S.l.], 2017. p. 1–7. Citado 2 vezes nas páginas 17 e 19.

NAMEKO. 2020. Disponível em: <https://www.nameko.io/>. Citado na página 29.

NETO, A. P. 2019. Disponível em: <https://medium.com/@alvaropaconeto/python-micro-services-nameko-flash-rabbitmq-docker-f85339abf42a>. Acesso em: 28 de novembro de 2020. Citado na página 28.

ORANGE. Lora device developer guide. 2016. Disponível em: <https://developer.orange.com/od-uploads/LoRa-Device-Developer-Guide-Orange.pdf>. Citado na página 12.

PGADMIN. 2020. Disponível em: <https://www.pgadmin.org/>. Citado na página 28.

POSTGRESQL. 2020. Disponível em: <https://www.postgresql.org/>. Citado na página 28.

PSYCOPG2. 2021. Disponível em: <https://pypi.org/project/psycopg2/>. Citado na página 29.

RABBITMQ. 2020. Disponível em: <https://www.rabbitmq.com/>. Citado na página 29.

RADIOENGE. 2020. Disponível em: <https://www.radioenge.com.br>. Citado 2 vezes nas páginas 25 e 30.

RASPBERRY. 2020. Disponível em: <https://www.raspberrypi.org>. Acesso em: 28 de setembro de 2020. Citado na página 26.

RASPBERRY. 2020. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Citado na página 29.

RISINGHF. 2020. Disponível em: <http://www.risinghf.com/product/detail/5>. Citado na página 30.

ROCHA, B. 2016. Disponível em: <http://brunorocha.org/python/microservices-with-python-rabbitmq-and-nameko.html>. Acesso em: 28 de novembro de 2020. Citado na página 28.

RODRIGUES, M.; CUGNASCA, C. E.; FILHO, A. P. de Q. **Rastreamento de veículos**. [S.l.]: Oficina de Textos, 2009. Citado na página 2.

SAMUELSON, C. Single channel gateway. 2017. Disponível em: <https://www.hackster.io/ChrisSamuelson/lora-raspberry-pi-single-channel-gateway-cheap-d57d36>. Citado na página 14.

SCULLY, P. **The Top 10 IoT Segments in 2018 – based on 1,600 real IoT projects**. 2018. Disponível em: <https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/>. Acesso em: 03 de novembro de 2020. Citado na página 6.

SEMTECH. Lora® and lorawan®: A technical overview. 2019. Disponível em: <https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/>. Citado 4 vezes nas páginas 8, 11, 15 e 16.

SEMTECH. 2020. Disponível em: <https://www.semtech.com>. Citado 6 vezes nas páginas 8, 9, 10, 11, 13 e 14.

SEMTECH. Sx1276-7-8-9 datasheet. p. 10, 2020. Disponível em: <https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276>. Citado na página 21.

SIGFOX. **Assess your project's needs**. 2020. Disponível em: <https://build.sigfox.com/study>. Acesso em: 27 de outubro de 2020. Citado na página 8.

SINGER, T. Tudo conectado: conceitos e representações da internet das coisas. **Simpósio em tecnologias digitais e sociabilidade**, v. 2, p. 1–15, 2012. Disponível em: <http://www.academia.edu/download/37718997/SimSocial-44965.pdf>. Acesso em: 1 de setembro de 2020. Citado na página 1.

SLATS, L. **University of Zaragoza Breaks Long-Range LoRaWAN®-based Signal Record**. 2019. Disponível em: <https://tech-journal.semtech.com/university-of-zaragoza-breaks-long-range-lorawan-based-signal-record>. Acesso em: 10 de outubro de 2020. Citado na página 10.

SONG, Y. et al. An internet of energy things based on wireless lpwan. **Engineering**, Elsevier, v. 3, n. 4, p. 460–466, 2017. Citado na página 7.

STATISTA, R. D. **Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025**. 2016. Disponível em: <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>. Acesso em: 08 de novembro de 2020. Citado na página 5.

STMICROELECTRONICS. B-l072z-lrwan1. 2020. Disponível em: <https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html>. Citado 2 vezes nas páginas 21 e 22.

TEKON. **Line of Sight - O verdadeiro alcance das comunicações sem fio**. 2020. Disponível em: <https://www.tekonelectronics.com/download.php?fd=61&l=pt&key=350e47f96d4e34728077ff8b2ef25416>. Acesso em: 20 de novembro de 2020. Citado na página 10.

TELECO, I. em T. **Espectro de Frequência ISM**. 2020. Disponível em: <https://www.teleco.com.br/tutoriais/tutorialredeswifi1/pagina_5.asp>. Acesso em: 22 de outubro de 2020. Citado na página 8.

TTI. **The Things Industries**. 2020. Disponível em: <https://thethingsindustries.com/>. Acesso em: 20 de novembro de 2020. Citado na página 24.

TUPA. **Linhas e itinerários**. 2018. 18 p. Disponível em: <http://www.patobranco.pr.gov.br/wp-content/uploads/2018/05/TUPA_Cartilha_Linhas_hor%C3%A1rios_ED-01_web.pdf>. Acesso em: 22 de agosto de 2021. Citado na página 42.

UBLOX. Neo-6 series. 2020. Disponível em: <https://www.u-blox.com/en/product/neo-6-series>. Citado 3 vezes nas páginas 22, 23 e 29.

WESTENBERG, M. Esp-1ch-gateway. 2020. Disponível em: <https://github.com/things4u/ESP-1ch-Gateway>. Citado na página 14.

ZANELLA, A. et al. Internet of things for smart cities. **IEEE Internet of Things journal**, IEEE, v. 1, n. 1, p. 22–32, 2014. Citado na página 5.