UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ENGENHARIA DA COMPUTAÇÃO

FELIPE HAUSCHILD GRINGS

MATCH IO:
UM SISTEMA IOT LPWAN DE UTILIZAÇÃO SIMPLIFICADA PARA O MONITORAMENTO DE GRANDEZAS FÍSICAS

SÃO LEOPOLDO
2021

FELIPE HAUSCHILD GRINGS

MATCH IO:
UM SISTEMA IOT LPWAN DE UTILIZAÇÃO SIMPLIFICADA PARA O MONITORAMENTO DE GRANDEZAS FÍSICAS

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia da Computação, pela Universidade do Vale do Rio dos Sinos – UNISINOS

Orientador: Prof. Me. Armando Leopoldo Keller

SÃO LEOPOLDO

2021

RESUMO

O objetivo deste trabalho é realizar um estudo de forma abrangente sobre o conceito de Internet das Coisas e as tecnologias que a rodeiam. O trabalho explora diversas arquiteturas voltadas para a otimização do conceito de IoT (*Internet of Things* – do inglês Internet das Coisas) usadas para a comunicação de longo alcance, visando um sistema de simples instalação para o usuário final. Para tanto é apresentada uma revisão bibliográfica sobre protocolos, arquiteturas, distribuições de componentes e a importância para o pequeno agricultor e indústria de monitorarem suas variáveis de ambiente para otimização dos seus processos. A partir do que se percebe há uma falta de disponibilidade de produtos que contemplem a facilidade de manuseio e instalação, utilizando destas tecnologias de forma que sejam acessíveis para empresas e agricultores que não dispõem de altos recursos financeiros para tal implementação. Diante deste problema, a proposta do trabalho é desenvolver um protótipo que entregue ao consumidor final um produto *end-to-end*, sem a necessidade de terceiros para configuração e de fácil instalação, independente de internet, e podendo utilizar embarcado em servidores locais ou providos por fornecedores. Para isso utiliza-se uma metodologia de desenvolvimento em etapas, onde é desenvolvido isoladamente o produto de hardware e *firmware*, o produto de software alto nível, como banco de dados, serviços de comunicação HTTP e interfaces de usuários, também como a implementação de *scripts* para a infraestrutura rodar como código. Com isso utiliza-se tecnologias como LoRa, para comunicação de longa distância, MQTT para serviços de mensageiria, Python e Flask para serviços de alto nível, entre outras tecnologias para simplificação do desenvolvimento. Com isso é possível a implementação do protótipo para monitoramento de grandezas físicas, assim como a transmissão para uma base de dados para futura análise e entendimento do ambiente. Para comprovar a eficiência e a confiabilidade do sistema, foram realizados testes contínuos ao longo de 6 meses em situação de uso real em um ambiente de criação de frangos. Os resultados obtidos apresentaram a taxa de mensagens enviadas com sucesso real sobre ideal de 84,56% e uma aprovação de média 4 de 5 entre os usuários em facilidade de uso e implementação do sistema. Estes testes comprovam o correto funcionamento do sistema proposto.

Palavras-chaves: IoT. LPWAN. MQTT. LoRa. Sistemas Embarcados.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

BLE	*Bluetooth Low Energy* (Bluetooth de Baixa Energia)
COAP	*Constrained Application Protocol* (Protocolo de aplicativo restrito)
CSS	*Cascade Style Sheet* (Folha de Estilos Cascade)
GPIO	*General Purpose Input/Output* (Entrada / Saída de Uso Geral)
GPS	*Global Position System* (Sistema de Posicionamento Global)
GSM	*Global System for Mobile Communications* (Sistema Global para Comunicação Móvel)
HDMI	*High Definition Multimedia Interface* (Definição Multimedia de Alta Definição)
HTML	*Hyper Text Markup Language* (Linguagem de marcação de hipertexto)
GW	*Gateway*
IEEE	*Institute of Electrical and Electronics Engineers* (Instituto de Engenheiros Elétricos e Eletrônicos)
IOT	*Internet Of Things* (Internet das Coisas)
IPV4	*Internet Protocol Version 4* (Protocolo de Internet Versão 4)
IPV6	*Internet Protocol Version 6* (Protocolo de Internet Versão 6)
LED	*Light Emitting Diode* (Diodo de Emissão de Luz)
MQTT	*Message Queuing Telemetry Transport protocol* (Protocolo de Transporte de Telemetria de enfileiramento de Mensagens)
PWM	*Pulse Width Modulation* (Modulação de Largura de Pulso)
RAM	*Random Access Memory* (Memória de Acesso Randomizado)
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)
TCP	*Transmission Control Protocol* (Protocolo de Controle de Transmissão)
UDP	*User Datagram Protocol* (Protocolo de Datagrama de Usuário)
WIFI	*Wireless Fidelity* (Fidelidade sem Fio)
WSN	*Wireless Sensor Networks* (Rede de Sensores sem Fio)

SUMÁRIO

1 INTRODUÇÃO

O conceito de Internet das Coisas foi introduzido por Kevin Ashton em 1999. Tema de grande discussão nos dias atuais por sua potencialidade de crescimento e movimentação de capital. A Comissão Mundial sobre Meio Ambiente e Desenvolvimento que afirmou que, até 2030, mais de 26 trilhões de dólares serão gerados globalmente, apenas em razão dos investimentos em tecnologias sustentáveis. A McKinsey, compartilhando da ideia, acredita que o impacto da IoT (*Internet of Things* – do inglês Internet das Coisas) no mundo, até 2025, gerará 11 trilhões de dólares (MCKINSEY, 2019)**.**

A definição do termo IoT gerou grandes divergências ao decorrer dos anos, ainda sendo muito discutido nos dias atuais. Segundo Ashton, “Coisa” é uma entidade ou objeto físico, podendo ser um veículo ou a representação de um conjunto de grandezas abstratas, tais como temperatura, pressão. Ele previu um mundo onde computadores estão capacitados a ver, ouvir e cheirar o ambiente ao seu redor, assim sabendo muito mais sobre coisas do que ideias, pois perante a visão de Ashton ideias são importantes, mas as coisas regem o mundo. Grande parte dos dados armazenados hoje são provenientes de interações humanas. Se tivermos computadores que sabem tudo, que podem saber sobre coisas, utilizando dados que eles mesmos captam, seríamos aptos a rastrear e quantificar tudo, assim trazendo grande eficiência para nosso meio de produção e compreensão do mundo (ASHTON, 1999). Os principais ramos de atuação nessa área de pesquisa destacam-se: indústria, agricultura, mobilidade, transporte e meteorologia. Essas áreas de atuação têm como objetivo a captação de dados para melhor interação com o ambiente.

O monitoramento das variáveis envolvidas na agricultura, acompanhado de um sistema de análise dos dados coletados, é de suma importância para aumentar a qualidade de produção, diminuir o desperdício e assim tornar essa prática economicamente mais viável. Ao conhecer tais pontos é possível manipulá-los. Sendo a agricultura apenas uma das diversas áreas de possível implementação da tecnologia (Guilhermo *et al*., 2018).

Usufruir de todas as vantagens tecnológicas citadas é praticamente impossível para pequenas e médias indústrias e agricultores sem um grande avanço tecnológico. Aqueles que dispõem da necessidade de tal ferramenta acabam por

gastar boa parte de sua receita para tal implementação. Prado e Santo (2014) defendem que a análise de dados obtidos do sistema, mesmo que isolados, são de grande valia para o entendimento e aproveitamento de recursos dentro da indústria.

Com o avanço da tecnologia o mercado se tornou mais competitivo para aqueles que comportam a infraestrutura e capital necessários para o investimento em sua evolução tecnológica. Indústrias e agricultores de todo o globo tem avançando cada vez mais com seus meios de produção, reduzindo desperdício e aumentando a qualidade, assim tornando seus preços mais competitivos.

Segundo Hitachi e Márcia Ogawa (2017), IoT é essencial para a indústria sobreviver, porém o mercado nacional não está pronto para tal tecnologia. Mesmo sabendo que diversas indústrias o necessitam, o empresário nacional investe pouco nessa oportunidade pôr o custo ser elevado de implantação e a complexidade desta tecnologia acaba por acarretar tal conclusão proposta por Hitachi e Márcia Ogawa (2017), mesmo com todo o avanço tecnológico nos últimos anos.

Além do cenário citado, outras barreiras também devem ser superadas, como: a integração do projeto no seu todo, oferecendo como produto final o transdutor acoplado à rede e infraestrutura de comunicação; o painel de configurações e monitoramento do usuário final, resultando em uma ferramenta simples e intuitiva desde a implementação dos transdutores até a configuração do sistema via interface de usuário.

A busca por uma entrega de um produto de confiança e fácil usabilidade se mostra crucial para o andamento deste trabalho, de forma que qualquer problema de monitoramento possa ser resolvido de forma simples e economicamente viável, quando comparada com as soluções dispostas no mercado atualmente.

1.1 Objetivo

Nesta seção são apresentados os objetivos do trabalho, uma vez que o problema a ser resolvido foi devidamente apresentado.

Este trabalho tem por principal objetivo a implementação de um sistema de Internet das Coisas utilizando LPWAN *(Low Power Wide Area Network, do inglês – Rede de Área Ampla de Baixa Potência)* de uso simplificado para o monitoramento de grandezas físicas. Essa implementação se dará a partir do desenvolvimento de um sistema completo, contemplando todas as etapas, sendo elas: desenvolvimento

de um protocolo para comunicação entre os dispositivos; construção do *hardware*; programação do *firmware*; escrita das configurações da infraestrutura de servidores como código, também conhecido como IaC (*Infrastructure as Code* – do inglês Infraestrutura como código); programação do *backend*; configuração do banco de dados; e programação da interface de usuário. As grandezas monitoradas são: temperatura; umidade; pressão atmosférica; e dióxido de carbono presente.

O sistema também contempla a fácil instalação e configuração dos equipamentos para o usuário final, sem a necessidade de terceiros especializados, reduzindo consideravelmente os custos de implementação, tornando uma alternativa mais viável. A utilização da tecnologia *LoRa* (*Long Range* – do inglês Longo Alcance) e protocolo *LoRaWAN* (*Long Range Wireless Area Network* – do inglês Rede de Área Sem Fio de Longo Alcance). O protótipo será capaz de comunicar-se utilizando a arquitetura desenvolvida neste trabalho, apresentando todos os dados em um *dashboard* interativo para o usuário final, disponibilizado via uma interface *Web*, onde o usuário poderá realizar todas as configurações de perfil de monitoramento para cada transdutor. O sistema será implementado em ambiente de campo real pelo período de seis meses para sua validação. Serão realizados avaliações das comunicações e sinais a partir da base de dados e, por fim, será realizado um questionário com usuários sobre as percepções de usabilidade do sistema.

1.2 Organização do trabalho

Este trabalho está dividido em 6 capítulos. Sendo este primeiro a parte introdutória, seguindo para o capítulo 2 onde é discutida a fundamentação teórica e tecnologias, que serão de grande valia para o andamento do trabalho, onde é citada a revisão bibliográfica sobre os protocolos de comunicação MQTT e LoRa. O capítulo 2 também expõe trabalhos e sistemas relacionados para melhor embasamento teórico e entendimento das ferramentas disponíveis no mercado. No capítulo 3 é abordada a metodologia utilizada a fim de aprofundar e organizar o desenvolvimento. O capítulo 4 apresenta o sistema proposto com um estudo de caso para obtenção de dados. No capítulo 5 é realizada a análise de resultados,

utilizando dados coletados durante 6 meses. Por fim, o capítulo 6 concluí o trabalho apresentando as considerações finais.

2 REVISÃO BIBLIOGRÁFICA

Para realizar o desenvolvimento do sistema de monitoramento de grandezas físicas é necessário abordar as tecnologias utilizadas no sistema. Dessa forma, o presente capítulo descreverá os conceitos e terminologias das tecnologias utilizadas e que cercam o Match IO.

2.1 Internet das Coisas

Nesta seção serão descritos os principais conceitos de Internet das Coisas, assim como a definição e os exemplos de aplicação.

2.1.1 Definição

O conceito de Internet das Coisas, também conhecido pela abreviação na língua inglesa IoT, é de difícil definição. A organização *IEEE* (*Institute of Electrical and Electronic Engineers* – do inglês Instituto de Engenheiros Eletricistas e Eletrônicos) criou um processo aberto à comunidade para ajudar a desenvolver uma definição que envolva todos os conceitos de Internet das Coisas, ainda sem sucesso absoluto. Hoje os esforços estão mais focados para a definição de subáreas da Internet das Coisas por se tratarem de assuntos mais específicos, como: *WSN* (*Wireless Sensor Network* – do inglês Redes Sem Fio de Sensores)

As múltiplas definições de Internet das Coisas encontradas na comunidade de pesquisa testemunham o forte interesse sobre o assunto e de sua vivacidade. Onde o próprio nome gera uma certa ambiguidade em seu conceito, por ser formado de dois termos, no qual Internet leva a visão de uma rede IoT, enquanto o segundo termo, Coisas, move o foco para objetos genéricos (Atzori *et al*. 2010).

Dentre as diversas definições dispostas na comunidade de pesquisa, as mais interessantes encontradas são dadas por Internet of Things Research e Zhou (2012) onde são explicadas de forma sucinta as definições de grandes grupos e empresas.

Cassagras (apud ZHOU, 2012) define a Internet das Coisas como uma grande infraestrutura de rede global, compatível com a Internet atual, que busca a captura de dados e capacidade de comunicação associando objetos físicos e virtuais. Essa infraestrutura de alto nível e interoperabilidade se utiliza da

capacidade de diferenciação entre sensores, objetos específicos e conectividade para o desenvolvimento de serviços e aplicações (ZHOU, 2012).

O Internet of Things Research define a Internet das Coisas como uma infraestrutura de rede global, dinâmica e com recursos de autoconfiguração, com base em protocolos de comunicação padrão e interoperáveis, nos quais as “coisas” físicas e virtuais têm identidades, atributos físicos e personalidade virtuais, usam interfaces inteligentes e são perfeitamente integradas à rede de informações.

Diante das definições mencionadas ficam evidentes as relações em comum, como a rede ser composta por objetos físicos e virtuais que se comunicam em harmonia com os padrões existentes, no qual se apresentam por um identificador único e alta interoperabilidade.

2.1.2 Aplicações

Grande parte das aplicações de Internet das Coisas ainda se encontram em posse da academia, tendo elas de vasta gama de aplicação, porém comumente se limitando a aquisição de algum dado externo específico, ou fornecendo as informações para um sistema externo, sem buscar a modularidade e adaptação do sistema.

Como exemplo de dispositivo IoT, utilizado para monitoramento de diversas variáveis em campo, o sistema *Digital Matter*, munido de GPS (*Global Positioning System – do inglês Sistema de Posicionamento Global)*, sensor de temperatura e umidade, é utilizado em grandes campos de plantações no qual adquire e transmite todos esses dados para uma central, onde o agricultor responsável pode ter uma melhor compreensão do sistema de chuvas, e a necessidade de irrigação de suas plantações (Digital Matter, 2020).

Automação Residencial é a utilização mais popular nos dias atuais, onde grandes empresas do ramo estão atuando fortemente, alcançando grandes resultados, podendo citar algumas como Amazon e Google com os serviços Alexa e Google Home, respectivamente. Onde é possível controlar e monitorar sistemas como: ventiladores, condicionadores de ar, iluminação, entre outros. Com esses dispositivos conectados grande parte da rotina do dia-a-dia é passível de ser automatizada, além da possível integração com os medidores de energia

inteligentes (Smart Grid), que resulta em grande auxílio no consumo de energia. (XIAO, 2018).

Grande parte dos sistemas de Internet das Coisas estão munidos de ferramentas para controle e compreensão do ambiente ao seu redor, para isso dispõem da captação de dados para melhor entendimento do contexto. Os dispositivos que realizam tal medição são chamados de transdutores.

2.2 Transdutores e interfaces

O transdutor é um dispositivo que realiza a transformação de uma grandeza física para outra. Esses dispositivos são amplamente utilizados nas áreas de automação, medição e controle de sistemas. O processo de transformação de energia é chamado de transdução (Winer, Ethan 2013).

Para realizar a comunicação com os transdutores existem diversos padrões de interfaces. Dentro das mais populares se encontra a interface 4 – 20 mA e I^2C.

A interface 4 – 20 mA é amplamente difundida na indústria desde os anos 80. Isso gerou uma gama de produtos e clientes muito grande, onde grande parte da indústria contém algum desses transdutores. Com a popularização da interface houve um aumento do mercado para esse uso, havendo diversos meios de alimentação, onde normalmente utilizava-se 24 V de alimentação. Nos dias atuais já é comum encontrar transdutores com alimentação de 8 V, pois se adequam melhor aos sistemas computacionais que estão ingressando neste meio.

O protocolo I^2C, inventado pela Philips no início da década de 90, utiliza apenas dois fios para comunicação serial. É um protocolo muito utilizado para comunicar periféricos de baixa velocidade de transmissão. O protocolo é composto pelos fios SDA (*Serial Data Line* – do inglês Linha de Dados Serial), SCL (*Serial Clock Line* – do inglês Linha de Clock Serial) e alimentação (VDD), usualmente 3,3 V ou 5 V. O número de nós é limitado pelo tamanho do endereço, que pode ser 7 bits, 10 bits ou 16 bits. O protocolo é popularmente conhecido e utilizado para ferramentas de IoT (NXP, 2014).

Para a leitura e interpretação dos valores transmitidos pelos transdutores que utilizam protocolos digitais é necessário o processamento lógico do sinal, para isso são utilizados microcontroladores.

2.3 Microcontrolador e módulos IoT

Os microcontroladores foram introduzidos pela Texas Instruments com o modelo TMS1000, em 1974, sendo pequenos computadores em um circuito integrado. Um microcontrolador pode conter um ou mais núcleos de processadores, memória e entradas/saídas programáveis. Memórias programáveis na forma de ferroelétrico RAM (*Random Access Memory* – do inglês Memória de Acesso Aleatório), e PROM (*Programmable read-only memory* – do inglês Memória apenas leitura programável) também estão presentes nos chips. Estes dispositivos são destinados para aplicações embarcadas, agregando as funções de leitura de sinais, processamento lógico e controle de saídas. Esses dispositivos são muito utilizados nas áreas automobilísticas, sistemas de controle, monitoramento, entre outras. Microcontroladores são amplamente utilizados para aplicações que exigem baixo consumo de energia (Shirriff, Ken, 2016).

Dentre tantos microcontroladores existentes no mercado, pode-se destacar dois que popularizaram sua utilização para soluções IoT. O ESP32 e ESP8266, lançados em 2016 e 2014 respectivamente pela empresa Espressif Systems com a finalidade de disponibilizar microcontroladores simples, de baixo custo de aquisição e consumo de energia. Ambos os dispositivos apresentam o módulo WiFi integrado, contudo o ESP32 conta com melhor desempenho computacional. ESP32 também utiliza o módulo BLE (*Bluetooth Low Energy* – do inglês Bluetooth de Baixa Energia) (Maier *et al*. 2016). O Quadro 1 apresenta as especificações de hardware dos modelos comentados.

Quadro 1 - Características ESP32 e ESP8266

Chip	ESP32	ESP8266
CPU	Tensilica Xtensa LX6 32 bit Dual-Core @ 160/240 MHz	Tensilica LX106 32 bit @ 80 MHz (up to 160 MHz)
SRAM	520 kB	36 kB
FLASH	2 MB (max. 64 MB)	4 MB (max. 16 MB)
Tensão	2,2 V to 3,6 V	3,0 V to 3,6 V
Corrente de operação	80 mA	80 mA
Linguagem	Free (C, C++, Lua, etc.)	Free (C, C++, Lua, etc.)
Wifi	802.11 b/g/n	802.11 b/g/n
Bluetooth	4.2 BR/EDR + BLE	-
UART	3	2
GPIO	32	17
SPI	4	2
I²C	2	1
PWM	8	-
ADC	18 (12-bit)	1(10-bit)
DAC	2(8-bit)	-

Fonte: Elaborado pelo autor.

Além da leitura dos dados disponibilizados pelos transdutores e processamento lógico, são necessários módulos específicos para a comunicação. Para solucionar estes problemas empresas, como a Heltec, desenvolveram módulos integrados com microcontroladores onde o produto final é um hardware pronto para desenvolvimento de soluções IoT.

Os hardwares desenvolvidos pela Heltec apresentam o módulo de tecnologia de rádio LoRa, que será abordado detalhadamente no próximo capítulo, integrado

com o microcontrolador ESP32. O módulo disponibiliza conectores para bateria e antena, o que facilita sua operação, além de outras características que estão listadas no Quadro 2.

Quadro 2 - Principais características módulo Heltec LoRa ESP

Processador	ESP32 (80 a 240 MHz Tensilica LX6 dual-core)
Interfaces	UART, SPI, I²C e Micro USB
Entradas/Saídas	Analógicas, Digitais e Display LCD
Memória RAM	448 kb
Memória Flash	8 MB
LoRa Chip	SX1276 ou SX1278
Antena	Conector IPEX
Sensibilidade	-139 dBm (máxima)
Potência de TX	18dB ± 2dB
Modulação	LoRa, FSK, GFSK e OOK
Consumo	10,8 mA (RX) e 130 mA (TX)
Alimentação	5 V ou 3,3 V
Dimensões	50,2 mm x 25,5 mm
Temperatura de funcionamento	-40 °C a 85 °C

Fonte: Adaptado de HELTEC, 2018.

O microcontrolador utiliza a interface SPI (*Serial Peripheral Interface* – do inglês Interface Serial periférica) para a comunicação com o módulo LoRa, assim de forma integrada é disponibilizado o processamento e a modulação utilizando a tecnologia de comunicação.

2.4 Meios de comunicação sem fio

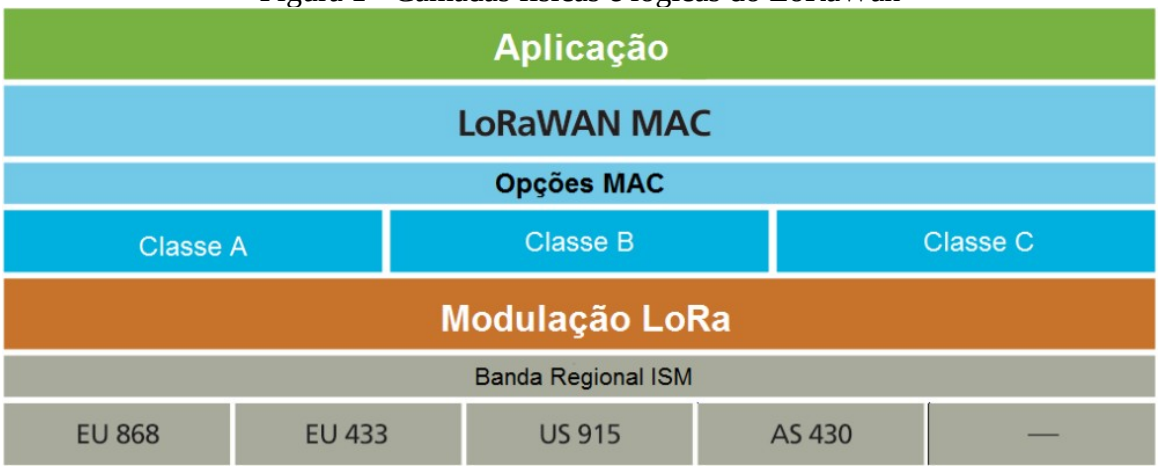
Nesta seção serão abordadas as principais tecnologias de meio de comunicação sem fio, junto com as tecnologias de rede de enlace mais populares e utilizadas na área de Internet das Coisas.

A comunicação sem fio utiliza do princípio de irradiação de ondas eletromagnéticas. Esse tipo de comunicação não necessita de meios físicos e se dispersa diretamente no ambiente (TANENBAUM, 2003).

LoRa é a camada física, ou modulação sem fio, utilizada para criar a conexão de comunicações de longo alcance. O LoRa se encaixa nos padrões LPWAN que emergiu como grande promissor da tecnologia de Internet das Coisas, possibilitando a integração de locais remotos com o ambiente virtual. A camada física, desenvolvida pela Semtech, derivada da modulação CSS (*Chirp Spread Spectrum – do inglês* Espectro de Propagação de Chirlos) traz a mesma otimização de energia da tecnologia FSK (*Frequency Shift Keying* – do inglês Modulação por Chaveamento de Frequência) porém adicionando grande alcance na comunicação, onde uma ERB (Estação de Rádio Básica) pode chegar à 15 km em visada direta com sinal ideal abaixo de -120 dBm RSSI (*Received signal strength indication* – do inglês Indicação de Força do Sinal Recebido). (SEMTECH, 2019).

A modulação LoRa utiliza a faixa de frequência ISM (*Institute Supply Management* – do inglês Instituto de Gestão do Fornecimento), que não necessita ser licenciada, podendo operar nas frequências 430, 433, 868 e 915 MHz, como mostrado na Figura 1, onde pode-se visualizar as camadas físicas e lógicas da modulação. A baixa frequência de propagação utilizada resulta em uma baixa largura de banda e taxa de transmissão de dados, porém por se tratar de uma tecnologia de uso aberto, não existe nenhum limite ao número diário de mensagens enviadas (SEMTECH, 2019).

Figura 1 - Camadas físicas e lógicas do LoRaWan



Fonte: Adaptado de Semtech, 2019.

Como apresentado na Figura 1, a modulação LoRa é a camada mais inferior dentro da arquitetura do protocolo LoRaWan. As diversas camadas subsequentes são relacionadas aos padrões implementados ao protocolo LoRaWAN buscando padronizações e melhorias para o uso em dispositivos de Internet das Coisas.

2.5 Protocolos de Comunicação IoT

Nesta seção serão abordadas as diferentes tecnologias de protocolos disponíveis para o desenvolvimento de aplicações IoT. Neste projeto foram introduzidos diversos protocolos IoT com o objetivo de fornecer uma comunicação eficiente para aplicações com recursos limitados, contudo cada protocolo contém suas vantagens e desvantagens.

2.5.1 LoRaWAN

Long Range Wide Area Network (LoRaWAN) é o protocolo de comunicação e a arquitetura de rede, enquanto LoRa é a camada física que foi abordada na seção 2.4 deste trabalho. O protocolo é responsável pela transmissão dos dados dos dispositivos finais, até os dispositivos chamados de *gateways*, que tem por objetivo centralizar os diversos nós disponíveis em uma rede LoRa (SENEVIRATNE, 2019).

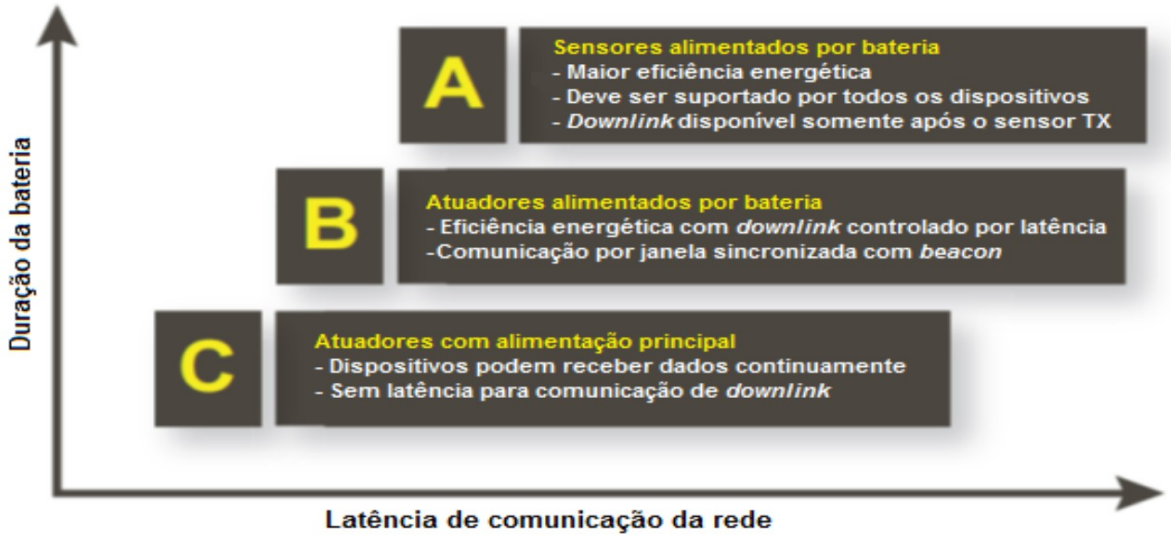
A arquitetura proposta para a utilização de LoRaWAN é constituída por, nós, *gateways*, servidor de rede e servidor de aplicação. Tipicamente quando um dado é enviado por um nó, ele é recebido por diversos *gateways que devem dar*

prosseguimento a transmissão dos dados. Os *gateways* utilizam da topologia de estrela para estrela, transmitindo por redes como: GSM *(Global System for Mobile Comunication – do inglês Sistema Global para comunicação de móvel)*, Ethernet, WiFi ou satélite. Os servidores de rede ao receberem esses dados repassam aos servidores de aplicação, que dão utilidade aos dados, assim disponibilizando-os para uso (SENEVIRATNE, 2019)

Outro fato importante sobre o protocolo LoRaWAN é sobre seu consumo de energia. O protocolo utiliza do modo de comunicação assíncrona, comunicando-se apenas quando há dados (evento) ou por tempo pré-determinado (agendamento). O método é conhecido por ALOHA, onde os nós sincronizam-se em intervalos de tempo. O método se mostra vantajoso quando comparado com redes síncronas convencionais, que realizam de tempos em tempos a verificação de novas informações, e, consequentemente, utilizam mais energia (LORA ALLIANCE, 2019.).

O LoRaWAN classifica seus dispositivos em três classes de operação dentro de determinados requisitos de funcionamento, como apresentado na Figura 2.

Figura 2 - Definições de classes LoRaWAN



Fonte: Adaptado de LORA ALLIANCE, 2019

Classe A: Tem como objetivo principal a economia de energia, operando de modo bidirecional. A cada evento o dispositivo envia a mensagem ao *gateway* entrando em uma janela de curto período de tempo no qual poderá receber dados do *gateway*, caso isso não ocorra o dispositivo entra em estado de hibernação, podendo ser comunicável novamente no próximo evento (LORA ALLIANCE, 2019).

Classe B: É a classe intermediária entre consumo e disponibilidade. Ele também opera de forma bidirecional assíncrona. Essa classe tem o diferencial de contudo operar de forma síncrona, uma vez que sincroniza com o *gateway* através de *beacons* periódicos, indicando a presença de rede sem fio, sincronizando com os dispositivos, assim abrindo janelas de comunicação programadas, podendo desta forma disponibilizar uma latência probabilística, de até 128 segundos. Em contrapartida há o aumento de consumo de energia, pois estará com a recepção ativa por mais tempo (LORA ALLIANCE, 2019).

Classe C: Por fim esta classe opera com a maior disponibilidade encontrada. A menor latência é resultado das mesmas especificações de comunicação bidirecional encontrada nas outras classes, porém esta sempre utiliza a recepção dos dispositivos ligados, permitindo o *gateway* e os servidores se comunicarem com os dispositivos remotos sem a necessidade de sincronização. Esta especificação por contrapartida trabalha com um gasto energético consideravelmente elevado, tornando inviável a utilização com baterias como fonte de alimentação (LORA ALLIANCE, 2019).

No quesito segurança o protocolo LoRaWAN utiliza de duas camadas, sendo uma em sua rede, garantindo a autenticidade dos dispositivos da rede; outra na parte de aplicação, garantindo a inviolabilidade dos dados. Utiliza a criptografia AES (*Advenced Encryption Standard – do inglês Padrão de Criptografia Avançado*), e como identificador o IEE EUI-64 (*Extended Unique Identifier 64 bits* – do inglês Identificador Único Extendido 64 bits) para troca de chaves, sendo o mesmo algoritmo utilizado em redes que utilizam o protocolo IPV6. (LORA ALLIANCE, 2019).

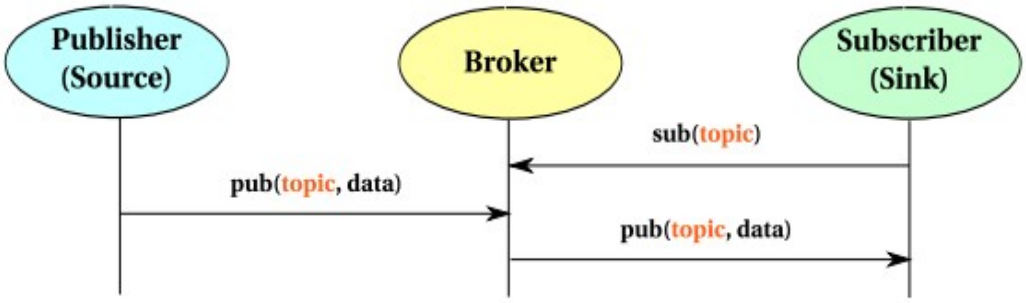
2.5.2 MQ Telemetry Transport

MQTT, do termo inglês *MQ Telemetry Transport*, foi introduzido por Andy Standford-Clard e Alrne Nipper em 1999. É um protocolo de base assinantes/publicador, extremamente simples e leve, voltado para aplicações de desempenho restrito e baixa largura de banda. Os princípios do design focam em minimizar o consumo de rede e os recursos exigidos pelos dispositivos. O protocolo

visa, também, garantir confiabilidade e diversos níveis de garantia de entrega. Estas características o tornam extremamente útil para o emergente mundo dos dispositivos conectados como Internet das Coisas, máquina-para-máquina, e para aplicações móveis onde a largura de banda e bateria são escassos (MQTT Org, 2020).

O processo publicador/assinante é ilustrado na Figura 3, onde os componentes interessados em consumir um determinado conteúdo devem realizar a assinatura de um determinado tópico, assim sendo chamado de *subscriber*. Componentes que estão interessados em divulgar suas informações, assim publicando-as, são chamados de *publishers*. A entidade responsável pela entrega dos dados entre os publicadores e os assinantes é chamado de *broker*. O *broker* coordena as inscrições, normalmente os inscritos contatam o *broker* diretamente (Hunkeler *et al*., 2020).

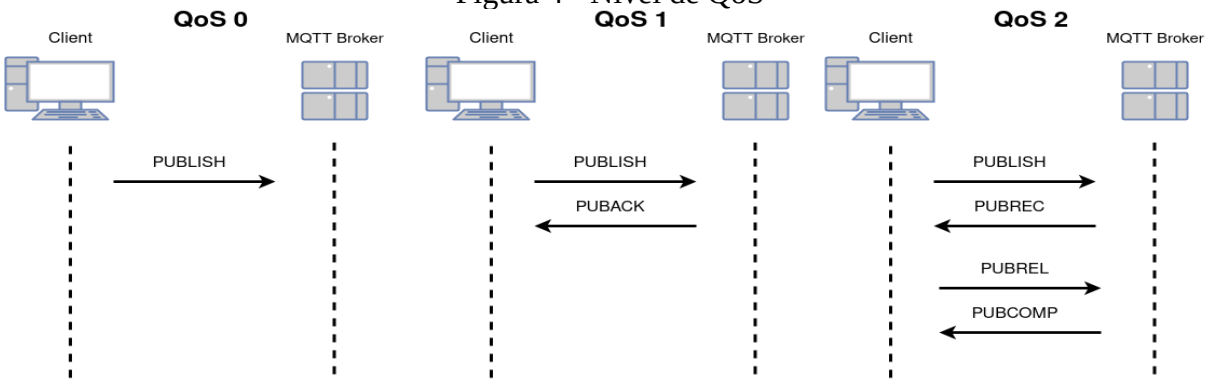
Figura 3 - Funcionamento do modelo publicador/assinante



Fonte: Hunkeler *et al*., 2020.

O protocolo é definido em 3 níveis de qualidade de serviço sob a troca de mensagens. A Figura 4 apresenta a troca de pacotes referentes aos 3 níveis de QoS. O nível 0 envia a mensagem apenas uma vez sem garantia de recebimento pelo *broker*. O nível 1 envia a mensagem pelo menos uma vez, realizando a verificação pela resposta de PUBACK. No entanto, se o PUBACK for perdido no caminho pode ocorrer de um segundo recebimento da mensagem original aos assinantes, pois não houve a primeira confirmação de recebimento. O nível 2 de QoS envia a mensagem exatamente uma vez. Por utilizar o método de *handsake* de 4 direções não é possível haver perdas nesse nível, porém devido à complexidade é possível que haja atrasos de ponto a ponto nas entregas.

Figura 4 - Nível de QoS



Fonte: Adaptado de Hunkeler *et al*., 2020.

No tópico seguinte serão apresentados os trabalhos relacionados que serviram de base para motivação dos estudos, levantamentos das tecnologias e exemplos de aplicações.

2.6 TRABALHOS RELACIONADOS

Nesta subseção são analisados e comparados sistemas, trabalhos e arquiteturas nas áreas de desenvolvimento do Match IO. Sistemas completos abrangendo todas as funcionalidades propostas neste trabalho são escassos na área de pesquisa. Alguns produtos como MATCHX e Seeed IoT foram muito utilizados como parâmetros para esse desenvolvimento. Devido ao fato deste trabalho envolver diversas áreas e a dificuldade de encontrar trabalhos, produtos ou sistemas que contenham a mesma abrangência foram selecionados trabalhos que contém no mínimo uma tecnologia de interesse, dentro de todas as utilizadas (LoRa, LoRaWAN, MQTT, IoT).

Em Nor *et al*. (2017) no segmento de Smart City utilizou-se do Seed LoRaWAN com GPS para o nó e o Dragino Lora-GPSHAT com Raspberry Pi 3 para o *gateway*. O objetivo geral era aperfeiçoar o monitoramento de tráfego em áreas urbanas mais densas da Malásia, como conclusão tiveram uma ótima percepção do uso da LoRaWAN, destacando-a para o sucesso do trabalho.

Já Grión *et al*. (2017) utilizaram tanto para o nó, quanto para o *gateway* o módulo Globalsat LM130. Como o objetivo do trabalho deles era realizar a análise do comportamento deste conjunto de componentes conectado a uma rede existente em Buenos Aires. Concluiu-se que o longo alcance obtido de 5 km em áreas urbanas é suficiente para o uso do conjunto como dispositivo de IoT, porém com consumo de energia relativamente elevado (500 mA durante os testes).

O artigo de Maksudjon e Francesco (2017) tem como intento o monitoramento de plantações em Torino, Itália, onde já são monitoradas por redes implementadas por BLE e ZigBee, porém ambas as tecnologias têm baixo alcance. Já redes de celulares tem maior alcance, porém precisam de painéis solares para alimentar os módulos e dependem, ainda, da disponibilidade dos sistemas de celular. Utilizando o módulo Ra-02 (baseado em SX1278), que foi prototipado para o nó e o *gateway* utilizando um Atmega328P com o objetivo de transmitir dados de 9 bytes para mil nós, enquanto realizam o monitoramento de irrigação da plantação. No projeto os objetivos foram atingidos com sucesso, além de estimar a durabilidade da bateria para até 2 anos.

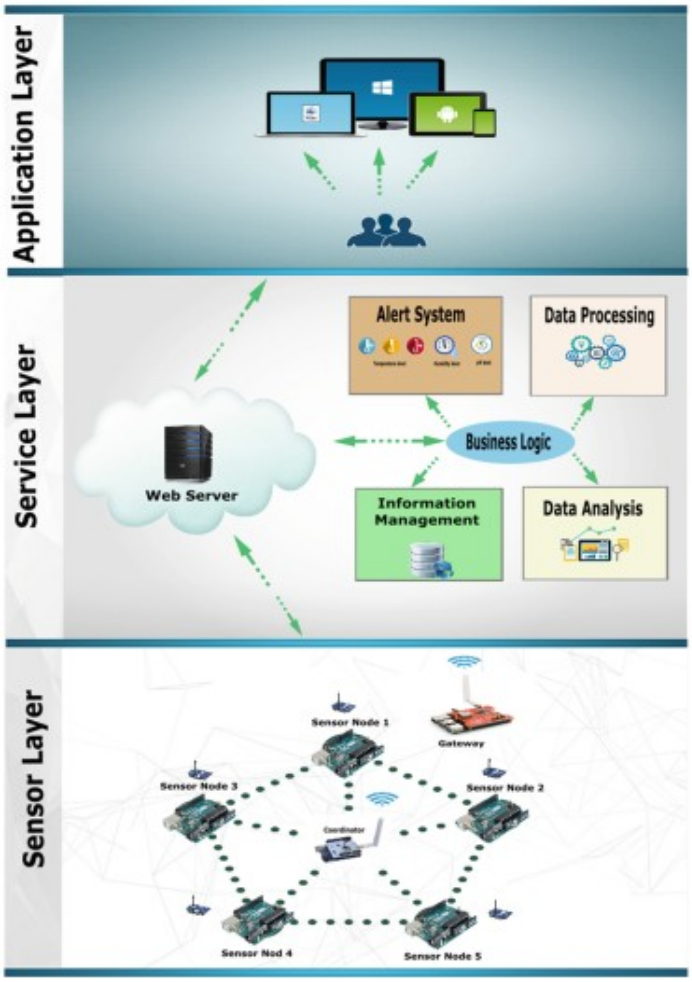
Em Alassio e Alfredo (2017) no segmento de baixo consumo de energia desenvolveram uma plataforma utilizando LoRaWan com consumo médio de 5 µA para deteção de temperatura de gados otimizando a classe A do protocolo LoRaWAN. Utilizaram a estrutura de FIFO (*First In First Out* – do inglês primeiro dentro primeiro fora) armazenando uma sequência de dados e descarregando o conjunto de forma sequencial quando carregar a memória a um nível determinado. O modulo comunicador utilizado foi SX1276 junto com o microcontrolador MS 430FR5969.

2.6.1 Sistema Mocca

Em Guilermo *et al*. (2020) desenvolve-se o sistema nomeado de MOCCA. Sua principal finalidade é o desenvolvimento de um sistema integrado entre sensores e usuário final, para o monitoramento do plantio de cacau na região do Equador, assim acompanhando em tempo real a temperatura, umidade, pH e luminosidade incidente no solo. O foco de usuários são os pequenos e médios agricultores de cacau no Equador.

O sistema é distribuído em três camadas, sendo elas: camada de aplicação, camada de serviço e camada de sensores, como mostra a Figura 5.

Figura 5 - Arquitetura sistema Mocca



Fonte: Guilhermo *et al*., 2020

Através desta arquitetura e interação entre as camadas, pequenos e médios agricultores são capazes de monitorar e coletar as informações captadas pelos sensores distribuídos, armazenar e posteriormente processar e visualizar, utilizando ferramentas de mineração de dados.

A camada de sensores é composta por três diferentes conjuntos de *hardware*, onde o mais externo é o nó de sensores. O nó de sensores é composto pelos sensores citados anteriormente, um Arduíno UNO e um módulo de comunicação ZigBee, assim realizando a comunicação com o nó coordenador. O nó coordenador é responsável por receber os dados do nó de sensores utilizando o protocolo ZigBee e enviá-los para o *gateway* através do protocolo LoRaWAN. Como processamento é utilizado o Arduino UNO. Para comunicação ZigBee e LoRaWAN são usados Xbee S2C e Dragino LoRa GPS Shield para Arduino, respectivamente. Por fim encontra-se o *gateway* que é responsável por receber os dados via LoRaWAN e realizar a

interação com a próxima camada, enviando os dados por Wi-Fi. Para isso foram utilizados Raspberry Pi e Dragino LoRa GPS HAT para Raspberry Pi.

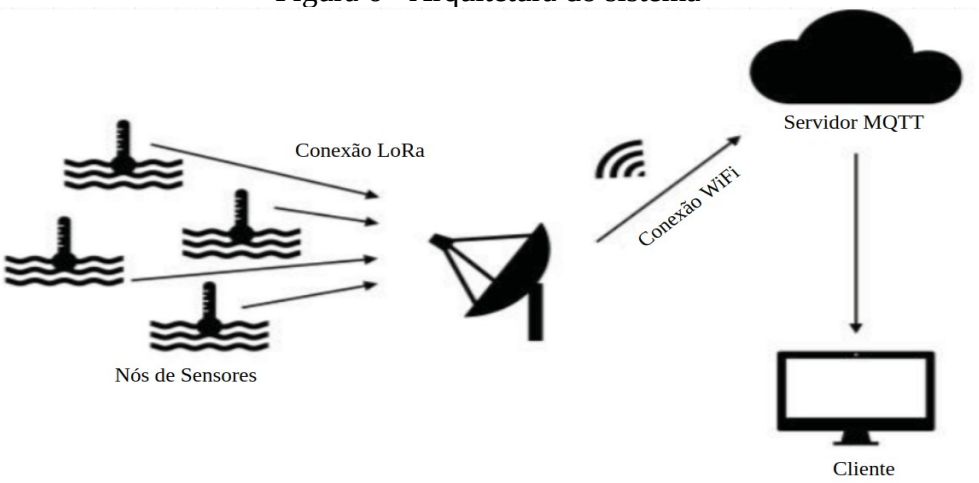
Após todos os dados devidamente adquiridos e enviados a internet, dá-se início ao processamento da camada de serviço.

A Camada de serviço fornece as ferramentas necessárias para gerenciar, processar e visualizar os dados da rede e gerar alertas imediatamente. A camada é composta por quatro módulos: gerenciamento de informações, gerenciamento de alertas e análise e processamento de dados. Todos os serviços são disponibilizados na nuvem, assim qualquer pessoa pode acessar de qualquer lugar do globo. As tecnologias utilizadas foram PHP, Angular, JavaScript e MySQL. Para processar os dados foi utilizado o Apache Spark Framework.

2.6.2 Rede WSN utilizando LoRaWAN

Em Huang *et al*. (2012), é proposto como exemplo de aplicação WSN um sistema de monitoramento de rede marinha utilizando LoRa e MQTT. O trabalho parte do princípio da comparação entre LoRa e um módulo de transmissão RF, onde os dois trabalham na frequência de 433 Mhz. A comparação visa identificar qual das duas tecnologias teria a menor taxa de perdas de pacotes e menor consumo de energia dentro da arquitetura proposta mostrada na Figura 6.

Figura 6 - Arquitetura do sistema

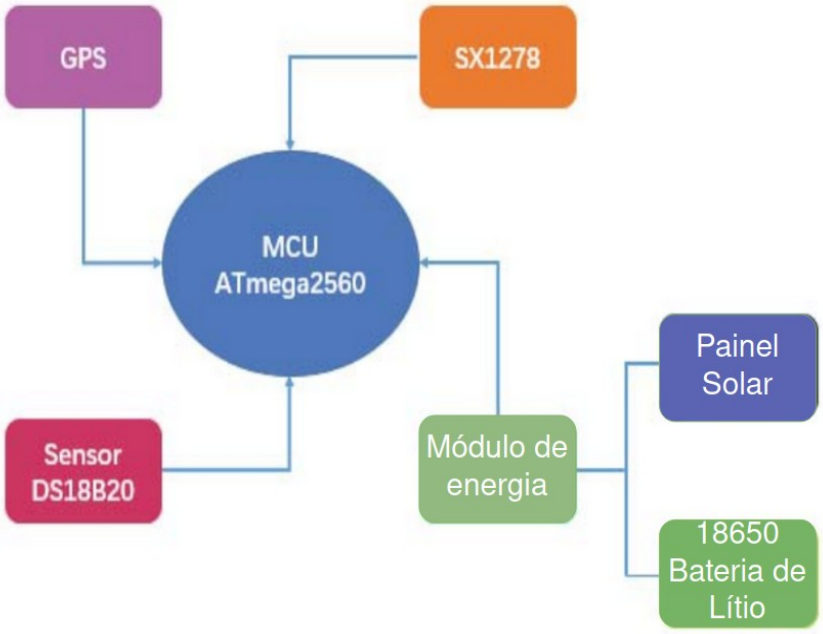


Fonte: Adaptado de Huang *et al*. 2017

O trabalho em sua fase de testes concluiu que a rede LoRa é superior utilizando o módulo SX1278 sobre o comunicador RF HC-12, onde o SX1278 obteve

até 5 km de alcance, enquanto o HC-12 obteve o máximo de 600 m. Como nó, foi utilizado o processador ATmega2560 comunicando com o módulo SX1278 via SPI, conectado com outros dispositivos necessários para a operação, como mostra a Figura 7.

Figura 7 - Diagrama de blocos



Fonte: Adaptado de Huang *et al*. 2019

Para o *gateway* foi proposta a utilização do processador ESP32 conectado via SPI ao SX1278, seguindo a mesma definição de comunicação do nó. No momento que o *gateway* recebe novos dados via o módulo, ele realiza uma chamada HTTP (*HyperText Transfer Protocol* – do inglês Protocolo de Transferência de Hiper Texto) a um serviço RESTful alocado em servidores na nuvem para o armazenamento dos dados para posteriormente serem analisados.

2.7 Relação entre trabalhos

As diversas soluções propostas e problemas a serem solucionados são muitas. Muitas chegaram ao sucesso por caminhos diferentes. Esta área ainda é pouco explorada perto da sua potencialidade de implementação e poder de solucionar problemas. No Quadro 3 é possível analisar as relações entre os trabalhos descritos neste capítulo e o trabalho desenvolvido.

Quadro 3 - Trabalhos relacionados

Autores	**Trabalho realizado**	**Relação com o trabalho**
Nor *et al*. (2017)	Monitoramento em cidade com LoRaWAN	Definição de tecnologias
Grión *et al*. (2017)	Simulação LoRa em meio urbano denso e testes de validação	
Maksudjon e Francesco (2017)	Design para implementação LoRa	
Bruno e Alfredo (2017)	Otimização LoRaWAN Classe A	
Guilermo *et al*. (2020)	Estudo de caso LoRaWAN	Motivação de aplicação
Huang *et al*. (2012)	LPWAN utilizando MQTT	Exemplo de aplicação

Fonte: Elaborado pelo Autor

Para fins de comparação foi levantado o custo dos materiais utilizados nos trabalhos correlatos, onde o Quadro 4 apresentada a relação de preços médio de

mercado do dia 4 de junho de 2021. Os preços foram analisados em buscas na internet com base em compras unitárias.

Quadro 4 - Preços de mercado dos dispositivos

Hardware	Preço
ESP32	R$ 41,00
Arduino Uno	R$ 69,90
Raspberry Pi 3	R$ 339,00
Arduino Mega	R$ 139,90
SX127x	R$ 39,90
Dragino LoRa GPS HAT	R$ 450,00
Dragino LoRa GPS Shield	R$ 300,00

Fonte: Elaborado pelo Autor

No capítulo seguinte é abordada a metodologia utilizada para guiar o desenvolvimento do trabalho em questão.

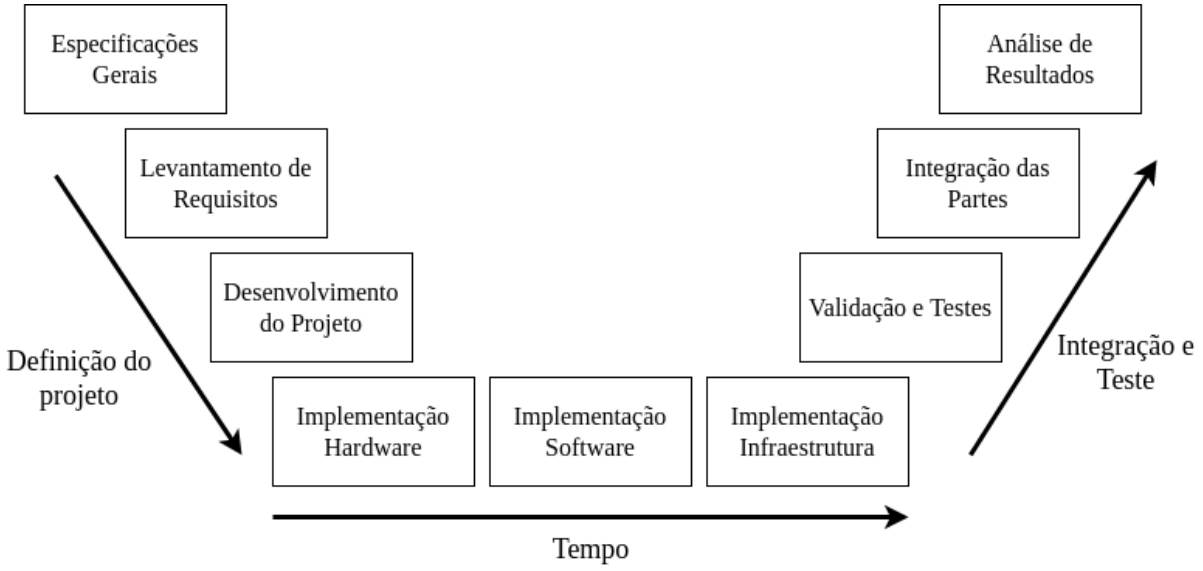
3 METODOLOGIA

Neste capítulo são apresentadas as etapas e ferramentas que foram utilizadas para o desenvolvimento deste trabalho e como o mesmo foi executado. Também são apresentados o ambiente e as características de onde o protótipo foi instalado.

Para reger a metodologia de desenvolvimento deste trabalho optou-se por utilizar o modelo V de desenvolvimento. Truytsa *et al*. (2012) diz que este modelo é um dos principais utilizados no ramo de desenvolvimento de sistemas de engenharia, dentro de diferentes tipos de projeto.

O desenvolvimento do trabalho foi dividido em 9 etapas, sendo 3 de implementação do projeto, como é apresentado na Figura 8, começando com a especificação geral, seguindo para o levantamento dos requisitos, então para o desenvolvimento do projeto. Feito isto, foram implementadas as partes de *hardware* e *firmware*, *software* e infraestrutura. As partes foram validadas em seus escopos e sequencialmente integradas. Após a integração o protótipo foi instalado e configurado em um ambiente rural, onde foi validado em uso real por 6 meses. Esta instalação serve de base para a análise dos resultados, levantamento de melhorias e validação do uso do sistema em casos reais.

Figura 8 - Modelo V adaptado



Fonte: Elaborado pelo autor.

Os requisitos do sistema são um aprofundamento dos conceitos gerais do projeto, de forma a apontar os requisitos funcionais e não funcionais. Abaixo estão listados os requisitos do sistema utilizados para análise de resultados e conclusão deste trabalho.

a) Monitorar grandezas de temperatura, umidade, pressão atmosférica e dióxido de carbono;
b) Comunicar os nós com o *gateway* utilizando protocolo LoRa;
c) Configurar um *broker* MQTT com capacidade mínima de 200 nós;
d) Visualização dos dados a partir de uma interface Web;
e) Configuração de frequência de leitura dos dados;
f) Armazenar os dados de leitura dos sensores em base de dados;
g) Acessar estes dados por meio do aplicativo desenvolvido.

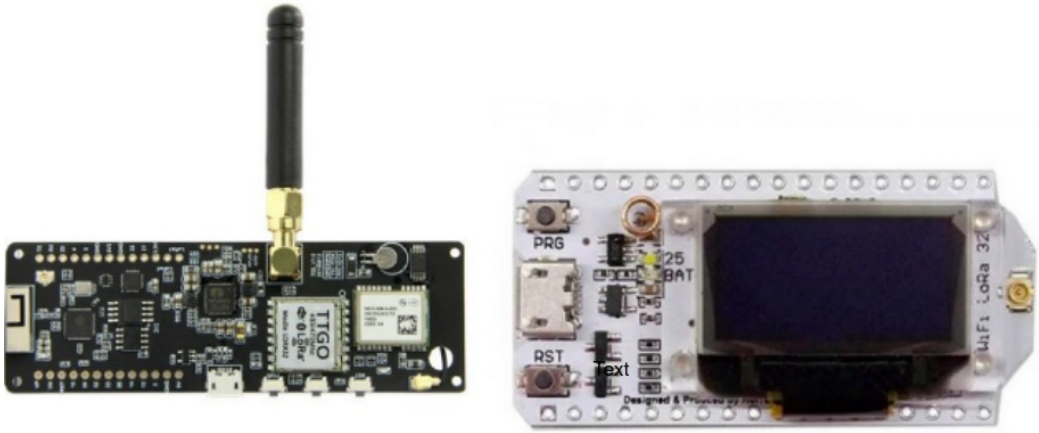
Ao finalizar o cumprimento desses requisitos o usuário deve ser capaz de utilizar o protótipo de forma simples e intuitiva, podendo, o próprio, realizar a instalação e configuração de sua rede de monitoramento com nós e *gateways*.

O desenvolvimento do projeto foi realizado em um laboratório de propriedade do autor, dispondo de equipamentos básicos para manipulação dos componentes eletrônicos, um notebook para desenvolvimento dos algoritmos e um servidor XEON E5 2678 v3, 64 Gb RAM com o sistema operacional Ubuntu 18.04, sendo conectado a internet utilizando a rede sem fio IEEE 802.11 para hospedagem dos serviços, e bancos de dados necessários.

A implementação de hardware para o nó e para o *gateway* foi realizada utilizando os módulos Heltec LoRa ESP32 V2, com custo de R$ 89,00, e Ttgo T-Beam ESP32 LoRa com custo de R$ 179,00, que utilizam o módulo ESP32 que apresenta excelente performance computacional em relação ao seu preço de mercado abaixo dos demais componentes utilizados nos trabalhos relacionados no capítulo 2, e display *OLED* (*Organic Light Emitting Diode* – do inglês Diodo Orgânico de Leve Emissão) para realizar a interface com o usuário, conforme a Figura 9. Os módulos foram escolhidos pela facilidade de integração, preço e desempenho satisfatório para redes com baixo tráfego, conforme descrito por Huang, *et al* (2017). Esse microcontrolador suporta a programação pelo ambiente de desenvolvimento

IDE Arduíno, em linguagem de programação C++, sendo compatível com diversas bibliotecas disponibilizadas pela ferramenta.

Figura 9 - Microcontroladores escolhidos

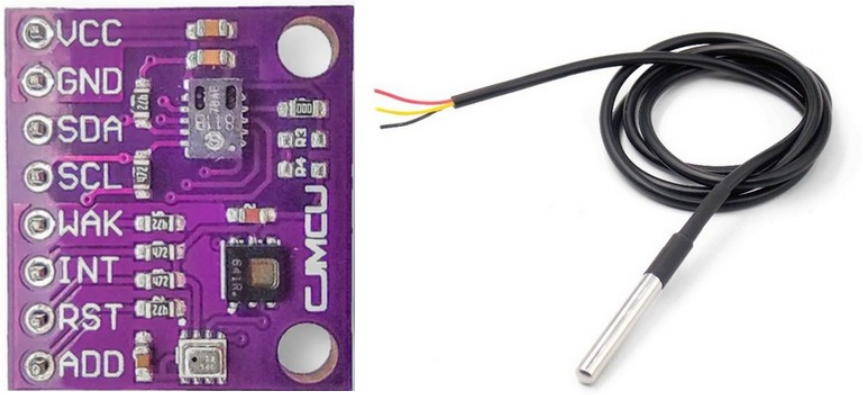


Ttgo T-Beam ESP32 Heltec LoRa ESP32

Fonte: Adaptado de HELTEC (2021) e TTGO (2021).

As grandezas físicas são obtidas pelo módulo CJMCU-8128, que agrega os transdutores CCS811, HDC1080 e BMP280, para medição de dióxido de carbono, temperatura, umidade e pressão atmosférica, respectivamente; e o transdutor DS18B20 para medição de temperatura, podendo alterná-los de acordo com a preferência do usuário. Os transdutores utilizam o protocolo I²C e OneWire para comunicação com o microcontrolador.

Figura 10 - Sensores escolhidos



CJMCU-8128 DS18B20

Fonte: Elaborado pelo autor.

Para comunicação entre os nós e o *gateway*, foi utilizado o chip SX1278, embarcado no microcontrolador escolhido. Os testes foram realizados utilizando o canal fixo de comunicação e potência de envio igual a 9 de 12 na escala disponível. Os dispositivos de nó contam com uma bateria de lítio 1800 mAh para seu fornecimento próprio de energia.

Para a comunicação entre os nós e o *gateway* foi desenvolvido um protocolo interno com quatro estados, onde cada estado representa uma ação. Este protocolo visa padronizar as mensagens enviadas e recebidas assim evitando erros e descartando mensagens com interferências. Os tópicos contém quatro padrões, sendo eles: Nó enviando dados lidos; Nó recebendo nova taxa de atualização; Configuração de nó na rede; e pedido de para envio de mensagens de teste. O padrão é apresentado no Quadro 5.

Quadro 5 - Protocolo interno de comunicação LoRa

Estado Protocolo	**Ação**	**Acionamento**
101	Enviar Leituras	Relógio
201	Nova Taxa Atualização	Usuário
300	Pedido de configuração	Usuário
301	Mensagens de configuração	Após receber a mensagem 200

Fonte: Elaborado pelo Autor

As Figuras 11 e 12 apresentam os padrões de mensagens de ponta a ponta. A estrutura apresentada na Figura 11 é enviada do nó ao MQTT Parser onde é reestruturada e armazenada no banco de dados não relacional. A Figura 12 mostra a estrutura que é gerada a partir da interface do usuário com a nova taxa de atualização do dispositivo, passando pelo *broker* MQTT e recebida pelo dispositivo.

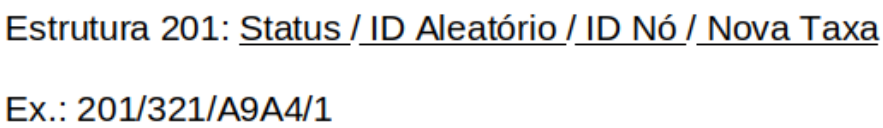
Figura 11 - Exemplo estrutura 101

Estrutura 101: Status / ID Aleatório / ID Nó / Valores / RSSI
Valores: Bateria, Temperatura, Umidade, CO^2, Pressão

Ex.: 101/321/A9A4/89,23.5,88,400,1004/-103

Fonte: Elaborado pelo autor

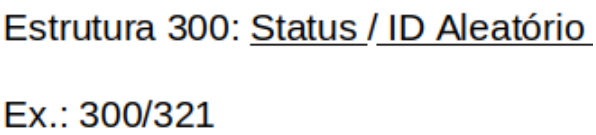
Figura 12 - Exemplo estrutura 201



Fonte: Elaborado pelo autor

As Figuras 13 e 14 apresentam as estruturas de mensagens utilizadas apenas em âmbito de comunicação entre os nós e o *gateway*.

Figura 13 - Exemplo estrutura 300



Fonte: Elaborado pelo autor

Figura 14 - Exemplo estrutura 301

Fonte: Elaborado pelo autor

A comunicação entre o *gateway* e o servidor MQTT é realizada utilizando o protocolo MQTT, seguindo o padrão de tópicos “/grupo/identificador unico/data” para publicação dos valores lidos pelo nó, onde “data” é uma palavra-chave específica para o mapemanto do *WebHook* e “/grupo/identificador unico/201” onde 201 é uma palavra-chave para identificação do intuito da mensagem. Esta definição de tópicos permite maior facilidade na instalação dos equipamentos por conter um identificador no início da requisição específico para instalação, configuração e leitura da potência do sinal.

Os serviços foram implementados utilizando a linguagem de programação Python 3.7, uma linguagem de *script* de alto nível interpretada, ideal para a prototipação de sistemas que apesar de ser nativa em sistemas operacionais baseados em Unix como o Linux, também pode ter o seu interpretador instalado em sistemas como Windows, permitido assim o desenvolvimento multi-plataforma. Foi

utilizado o microframework web, Flask, escrito em Python. Classificado como um microframework porque não utiliza ferramentas adicionais, mantendo o núcleo simples, porém extensível. Não possui camada de abstração de banco de dados, validação de formulário ou quaisquer outros componentes onde bibliotecas de terceiros pré-existentes fornecem funções comuns. No entanto, o Flask oferece suporte a extensões que podem adicionar recursos do aplicativo como se fossem implementados no próprio Flask.

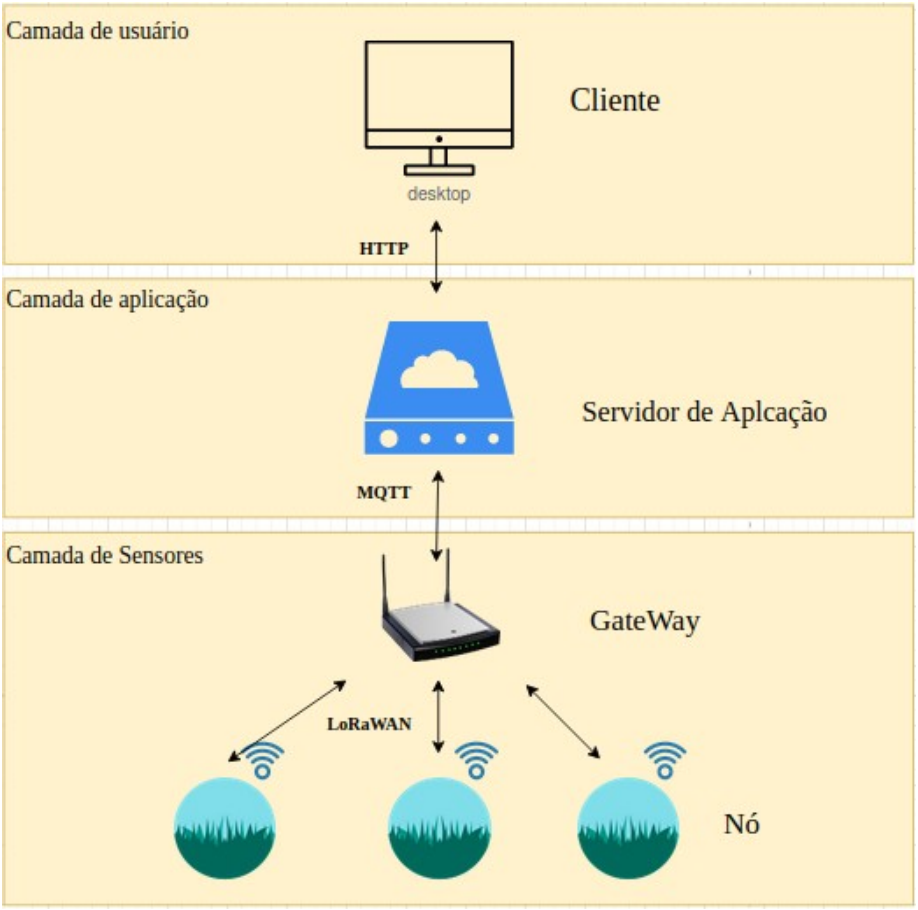
Para o armazenamento dos dados coletados e enviados pelos nós foi configurado o banco de dados não relacional, InfluxDB. As mensagens são enviadas e filtradas em duas camadas, onde o *gateway* realiza o filtro de estrutura do protocolo desenvolvido e o MQTT WebHook realiza o filtro pelo padrão de tópicos configurado neste trabalho. As mensagens válidadas pelas duas camadas são consideradas legítimas e então armazenadas no banco de dados.

A padronização dos serviços foi resolvida com a utilização da ferramenta de virtualização de contêineres, onde cada serviços compunha de seu próprio Dockerfile. Os contêineres Linux são um conjunto de processos organizados que ficam isolados do sistema. Todos os arquivos necessários para executá-los são fornecidos por uma imagem distinta. Na prática, os contêineres Linux são portáteis e consistentes durante toda a migração entre os ambientes de desenvolvimento, teste e produção, sendo transparentes ao ambiente que estiver rodando.

Para implementação da infraestrutura foram utilizadas as ferramentas Docker Swarm e Shell Script.

A arquitetura geral é apresentada na Figura 15, nela é possível compreender como ocorre a comunicação entre todas as camadas do sistema, a comunicação e distribuição dos nós e *gateways*, passando pelos servidores de aplicação até a camada final de usuário.

Figura 15 - Arquitetura geral



Fonte: Elaborado pelo autor.

No tópico seguinte serão apresentados o local e ferramentas utilizadas para implementação do sistema, junto os códigos desenvolvidos para análise dos resultados.

3.1 Ferramentas e ambiente de testes

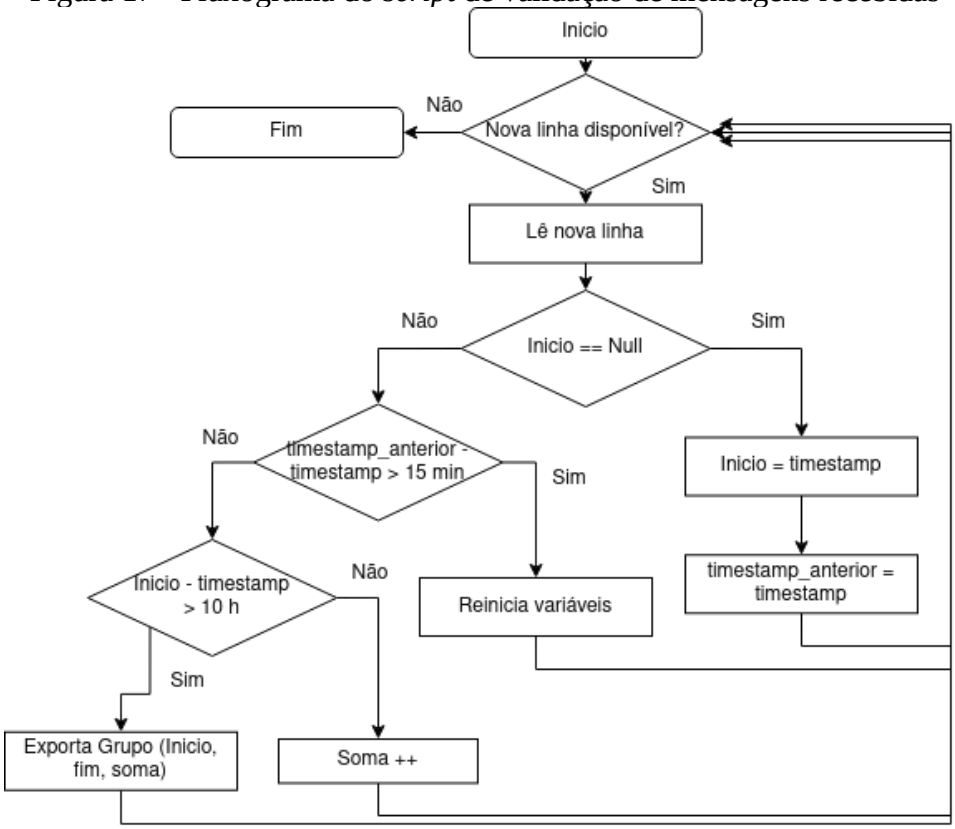
O trabalho foi instalado para o funcionamento de longo prazo em uma fazenda no interior de Caçapava, no Rio Grande do Sul – Brasil. Os nós foram alocados dentro de aviários de 130 x 15 m, gradeados nas laterais. A distância entre os nós e o *gateway* foram de 180 e 200 m, aproximadamente, sem visada direta. O *gateway* foi instalado no interior de uma casa de alvenaria ao lado do roteador WiFi, e a comunicação via internet foi realizada utilizando a tecnologia de comunicação via satélite, disponibilizada pela empresa Hughesnet, com velocidade aproximada de 3 Mbps.

Figura 16 - Local da instalação

Fonte: Elaborado pelo autor.

Para realizar os testes de comparação entre a quantidade de sinais enviados válidos real e ideal foi configurada a estrutura descrita neste trabalho com taxas de atualizações de 1 e 2 minutos para dois diferentes nós durante o período de seis meses. Os dados foram exportados da base de dados para um arquivo CSV (*Comma-separated values* – do inglês Valores Separados por Vírgulas) e analisados por um *script* desenvolvido utilizando a linguagem de *script* Python apresentado na Figura 17. Dentro deste período foram selecionadas faixas de 10 horas contínuas que continham o intervalo máximo entre mensagens igual a 15 minutos, desta forma isolando possíveis problemas de comunicação entre o *gateway* e o banco de dados, como problemas nos servidores ou no provedor de internet. Os valores -127 e 85 são conhecidos como falhas de leitura do transdutor DS18B20 considerados como comunicações válidas quando analisadas junto a base de dados. Intervalos de comunicação maiores que um dia são considerados que o sistema foi desconectado para manutenção do ambiente instalado. O segundo teste refere-se a alteração na potência do sinal durante o período. Os dados finais são importados para o *software* Excel onde são analisados os grupos com menores e maiores perdas assim como os valores totais, para que possa ser definido a taxa mínima de mensagens enviadas confiável.

Figura 17 - Fluxograma do *script* de validação de mensagens recebidas



Fonte: Elaborado pelo autor.

Para validação da usabilidade do sistema foram selecionados seis usuários sem experiência na área para a instalação e configuração do sistema, desconsiderando a instalação da infraestrutura nos servidores de aplicação, dado que ela é configurada a partir de um comando em *bash script*. Ao final os usuários foram questionados com perguntas onde as respostas variam de 1 a 5, sendo 1 muito difícil e 5 muito fácil, sendo elas:

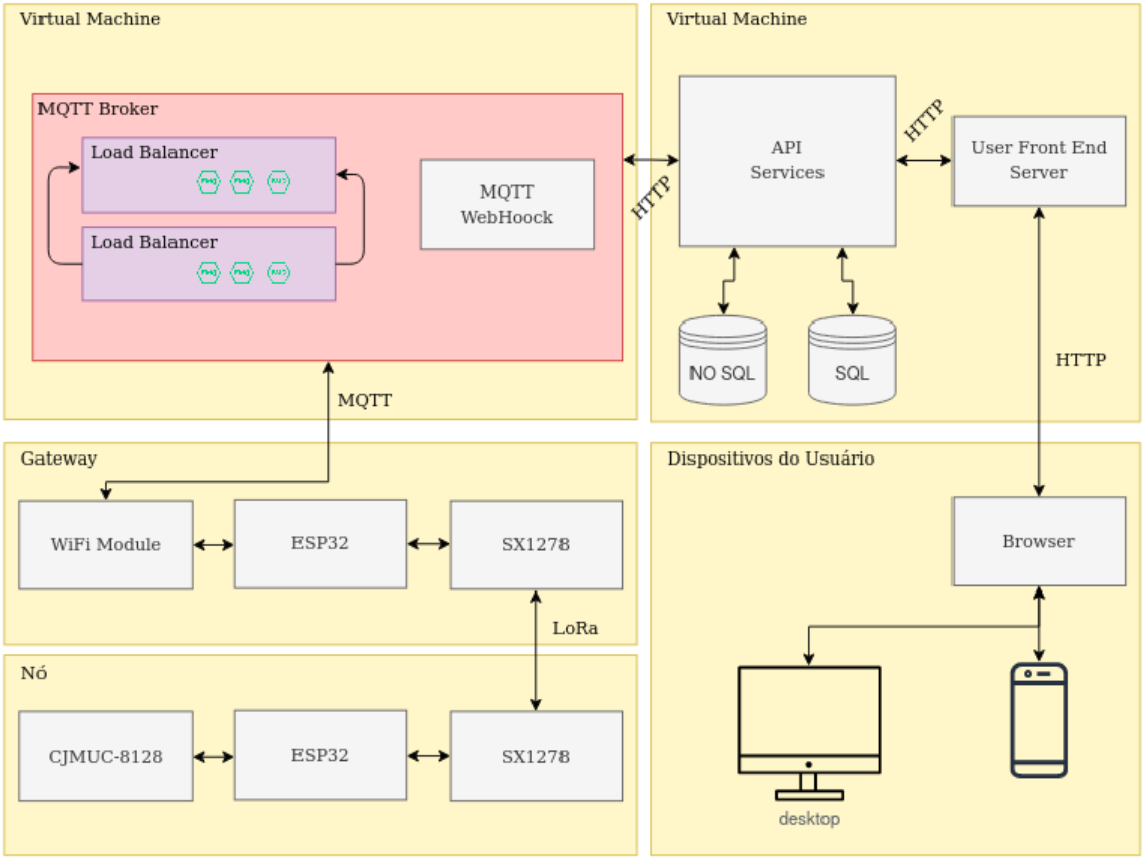
- Em sua percepção, qual o nível de facilidade de configuração do *gateway*?
- Em sua percepção, qual o nível de facilidade de configuração do nó?
- Em sua percepção, qual o nível de facilidade de interação com o nó?
- Em sua percepção, qual o nível de facilidade de cadastro de dispositivos na interface de usuário?
- Em sua percepção, qual o nível de facilidade de interpretação dos resultados pela interface de usuário?

Após o questionário foram compilados os todos os resultados e assim verificada a viabilidade da utilização do sistema em situações reais.

4 SISTEMA PROPOSTO

Neste trabalho é proposto o desenvolvimento de um sistema capaz de integrar as leituras realizadas pelos nós no interior de um aviário à tela de um dispositivo móvel utilizado pelo usuário final. A Figura 18 apresenta a arquitetura dethalhada representando as etapas e protocolos do fluxo de comunicação. É proposto também a configuração de um *cluster* MQTT, para melhor escalabilidade e infraestrutura da aplicação, suportando uma quantidade de nós consideravelmente mais alta. Na Figura também pode-se visualizar de forma geral a arquitetura dos serviços web e base de dados.

Figura 18 - Arquitetura detalhada

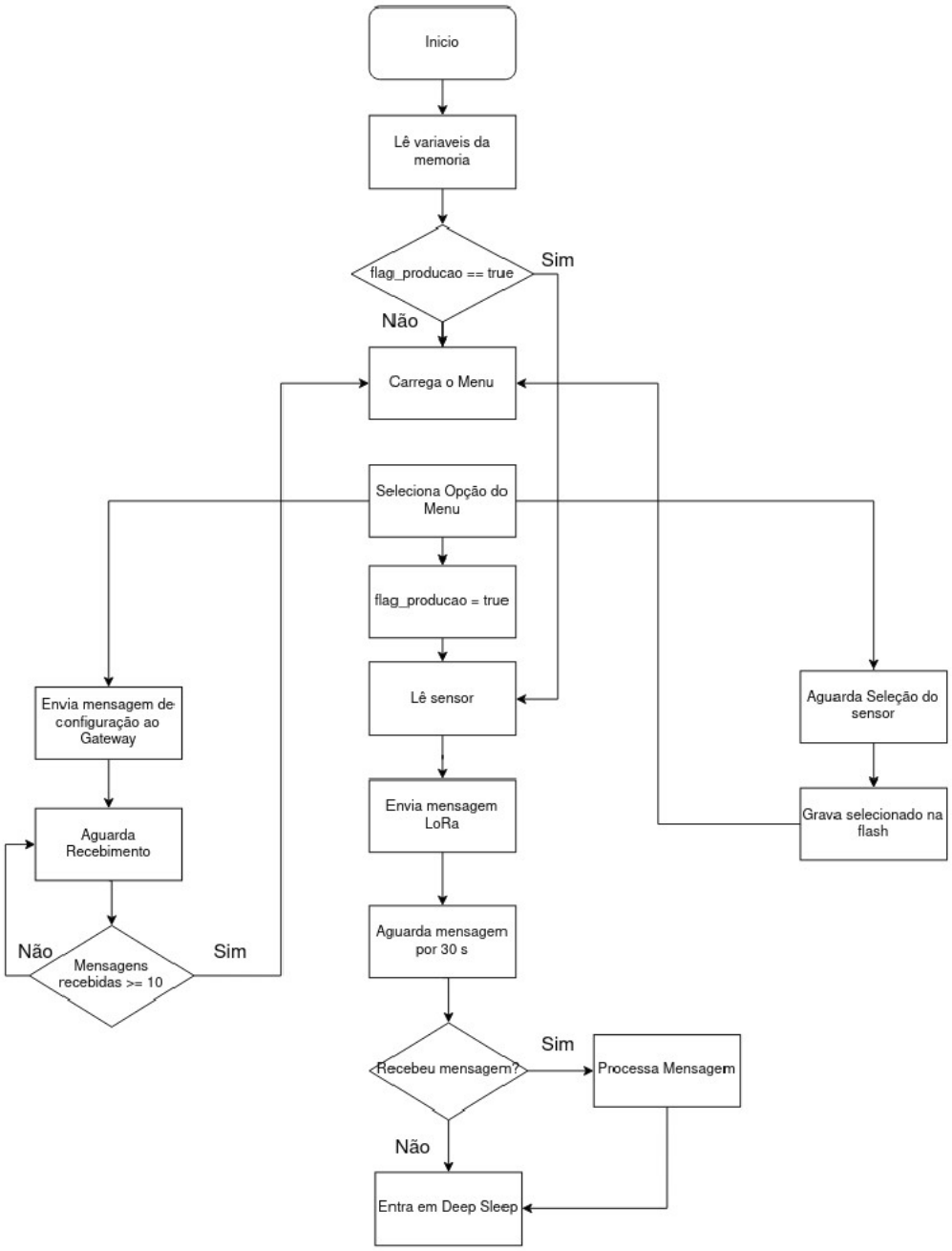


Fonte: Elaborado pelo autor.

A programação do nó foi desenvolvida essencialmente em linguagem de programação C++. O dispositivo visa a economia de bateria se adequando na classe A do protocolo LoRaWAN. Ele inicia em uma tela de menu aguardando a interação com o usuário para sua configuração inicial. O menu apresenta 3 opções, sendo elas: configuração do sensor; configuração da posição (leitura de sinal); e modo

produção. A aplicação principal funciona em modo de máquina de estados, aguardando a seleção de alguma das opções para dar prosseguimento a execução. No estado de configuração do sensor o dispositivo apresentará todas as possibilidades de sensores a serem utilizados, aguardando a seleção ou o cancelamento do usuário. Caso seja selecionado outro sensor, diferente do atual, o dispositivo então alterará na memória o transdutor requerido e voltará para a tela de menu. No estado de configuração de posição o nó enviará uma mensagem de pedido de configuração, assim o *gateway* interpretará a mensagem e enviará uma sequência de sinais com uma contagem no campo de mensagem, assim o nó receberá o sinal e apresentará no *display* o valor da mensagem e a potência do sinal recebido pelo *gateway*. Após as 20 mensagens ou o tempo determinado de *timeout* o *display* volta a tela de menu. A opção de modo de produção inicializa as medições de forma infinita, este é o estado principal do dispositivo. As variáveis de memória armazenam os seguintes dados: sensor a ser lido e tempo de intervalo entre leituras. Assim o dispositivo realiza a leitura do transdutor e envia os valores para o *gateway*, aguarda 30 segundos caso o *gateway* contenha alguma mensagem para o nó, se houver, recebe a mensagem e a interpreta, caso contrário volta ao modo *deep sleep* e aguarda a próxima leitura, conforme apresentado na Figura 19. Os protocolos para interpretação das mensagens são apresentados nos Quadros 3 e 4.

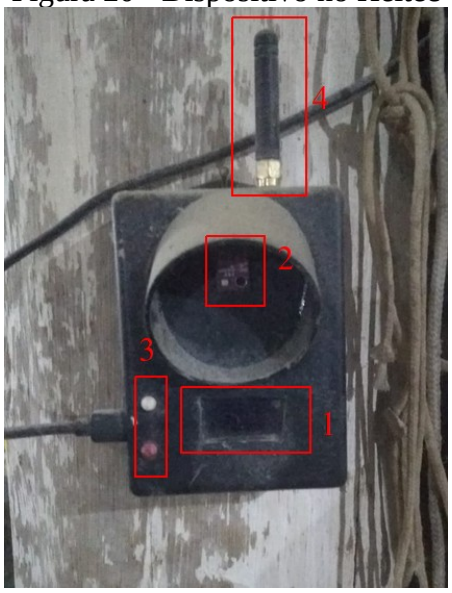
Figura 19 - Fluxograma nó



Fonte: Elaborado pelo autor

As Figura 20, 21 e 22 apresentam o resultado final da construção do nó Heltec, nó Ttgo e *gateway*, respectivamente, instalados no ambiente de testes.

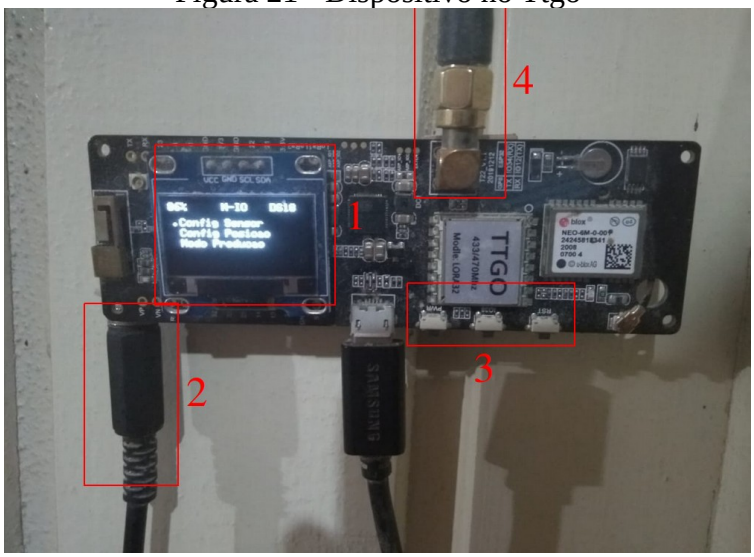
Figura 20 - Dispositivo nó Heltec



Fonte: Elaborado pelo autor

Os itens 1 e 3 da Figura 20 apresentam as interfaces com os usuários, onde o item 1 é o monitor para configuração e navegação no menu desenvolvido e o item 3 são os botões de seleção e reinício do aparelho. O item 2 é o módulo CJMCU-8128 exposto ao lado de fora da caixa de proteção porém protegido de poeiras e possíveis partículas que se depositam nas superfícies, como pode ser visto na imagem. O item 4 é a antena utilizada para comunicação LoRa.

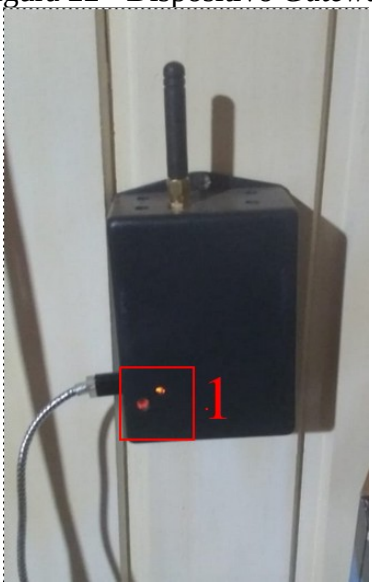
Figura 21 - Dispositivo nó Ttgo



Fonte: Elaborado pelo autor

Os itens 1, 3 e 4 da Figura 21 são relacionados a Figura 20. O item 2 é o conector P2 utilizado para comunicação com o transdutor DS18B20. O módulo TTGO conta com o conector para bateria acoplado no verso da placa, enquanto o módulo HELTEC utiliza o conector Jst Sh1 de 2 Vias.

Figura 22 - Dispositivo *Gateway*



Fonte: Elaborado pelo autor

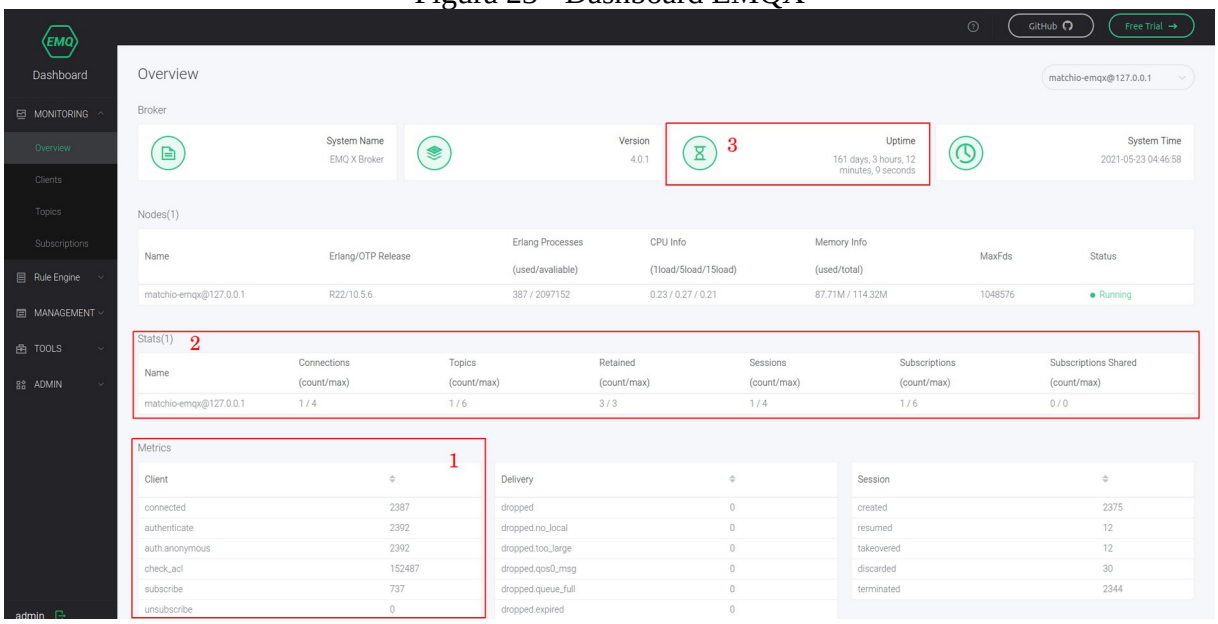
O item 1 da Figura 22 apresenta a interface de usuário luminosa que representa o estado do dispositivo, com ligado e duas piscadas a cada mensagem recebida válida ao protocolo.

Seguindo o mesmo padrão de programação utilizado no nó. O *gateway* conta com a vantagem de não precisar lidar com leitura de transdutores, sendo o código voltado ao processamento e envio de mensagens entre os protocolos. Como apresentado no Apêndice 1, o *firmware* é orientado a interrupções. Mensagens recebidas pelo protocolo LoRa passam pelo filtro do protocolo para identificação da chamada. Com valor 300 o *gateway* entra em modo de configuração e envia 20 mensagens sequenciais com o valor da mensagem sendo um contador de 1 a 20 com intervalo de 3 segundos. Essa configuração é essencial para o auxílio do posicionamento do nó. Caso o código da mensagem seja 301 o *gateway* envia a mensagem ao *broker* MQTT, e logo após busca na fila de mensagens pendentes pela chave de identificação do nó que enviou a mensagem, caso encontre alguma mensagem na fila, o *gateway* enviará a mensagem para o nó pela rede LoRa.

Interrupções geradas pelas chamadas MQTT são validadas pelo protocolo e são armazenadas em uma fila no formato chave-valor, onde o ID do dispositivo destino e o valor da mensagem são a chave e valor, respectivamente.

Inicialmente a plataforma ideal para o desenvolvimento do *broker* MQTT seria o *software* Mosqitto, porém apresentou-se problemático para lidar em formato de *cluster* e configurar exportações dos valores recebidos pelo protocolo MQTT, com isso optou-se pela utilização da plataforma EMQX, inicialmente paga, porém com uma versão *open-source* e gratuita. A plataforma conta com ótima distribuição para gerenciamento em forma de *cluster*, fácil configuração, *dashboard* para análise das métricas do *broker* e uma ferramenta de WebHook, para exportação dos dados recebidos no *broker*. Todas as configurações iniciais podem ser realizadas utilizando a API (*Application Programming Interface* – do inglês Interface de Programação de Aplicações) disponibilizada pela plataforma, convergindo para a utilização e configuração da infraestrutura em forma de código. Configurou-se o *WebHook* da plataforma para que todas as mensagens recebidas pelo *broker* sejam encaminhadas a API REST desenvolvida enviando as informações: hora de recebimento; origem; mensagem; entre outras informações. Como apresentado na Figura 23, os itens 1 e 2 disponibilizam métricas sobre as conexões realizadas ao *cluster*. O item 3 apresenta o *uptime* do servidor.

Figura 23 - Dashboard EMQX

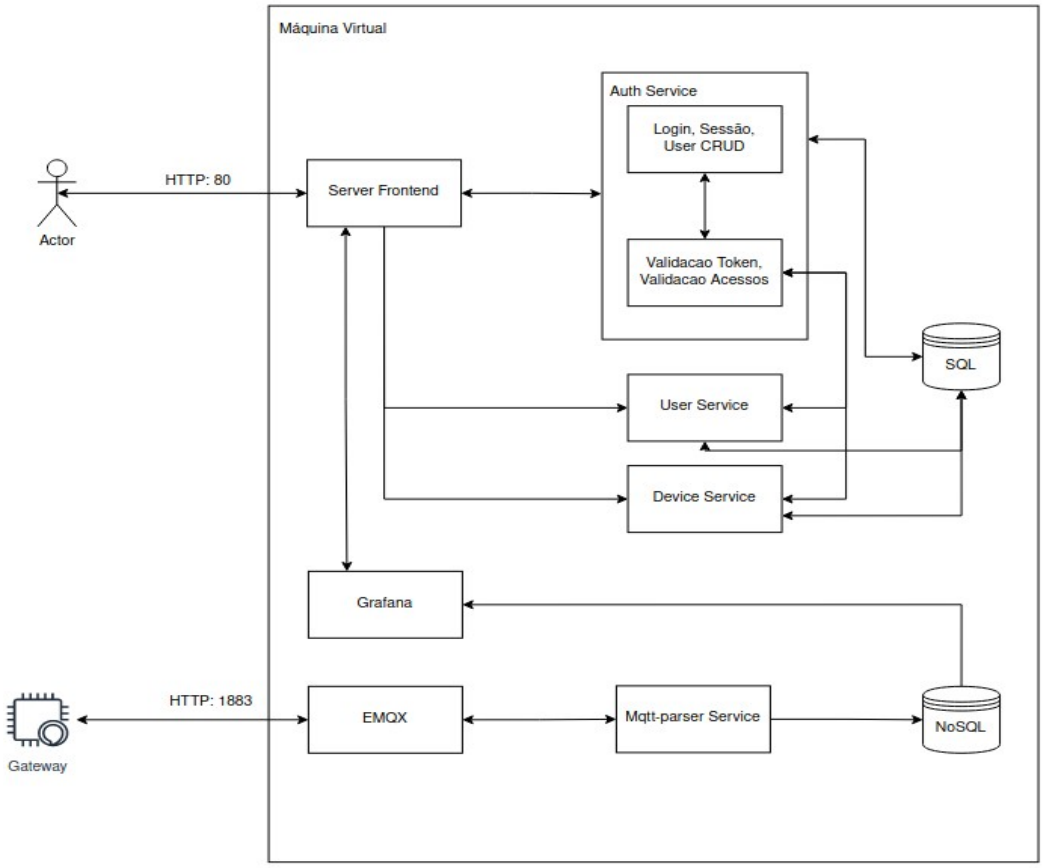


Fonte: Elaborado pelo autor

A arquitetura de serviços desenvolvida busca o isolamento entre funcionalidades, possibilitando a escalabilidade de diferentes partes do projeto, sem carregar o ônus de escalar a parte não utilizada. O microsserviço com maior necessidade de escalabilidade é chamado pelo *WebHook* configurado no *broker* MQTT, dessa forma possibilita a utilização de diversos nós conectados a rede, sem a necessidade de escalar o serviço de interface com o usuário, por exemplo.

O serviço “mqtt-parser” tem por finalidade expor uma API REST que é chamada toda vez que uma mensagem chega ao *broker* MQTT. O serviço interpreta a chamada, filtra os valores recebidos em *JSON* (*JavaScript Object Notation* – do inglês Notação de objeto Javascript) e adiciona os valores no banco de dados não relacional. Para controle e consulta dos dispositivos em relação a taxa de atualização, acesso, identificador e proprietário, desenvolveu-se o serviço “device-service”. Os serviços de usuários e login responsabilizam-se pelo controle de acesso e o relacionamento entre dispositivos e usuários. O padrão de segurança implementado utiliza a autenticação por JWT (Java Web Token – do inglês Símbolo Web Java). Todos os usuários após logados recebem o *token* de autenticação. Este *token* consiste de uma sequência alfanumérica única com prazo de validade. A cada nova requisição executada pelo usuário o *token* é atualizado. Por utilizar uma arquitetura de microsserviços, a vulnerabilidade dos *end-points* é maior, tanto em questão de exposição para qualquer usuário web acessar, tanto quanto para usuários sem permissão. Para contornar esta exposição foi desenvolvido um *end-point* responsável pela validação de acesso dos usuários em relação ao cargo obtido através do *token* de login. Este *end-point* retorna verdadeiro ou falso, assim o serviço requisitante pode dar ou não prosseguimento a execução da requisição.

Figura 24 - Arquitetura Serviços



Fonte: Elaborado pelo autor

Para o armazenamento considerou-se a utilização de bancos de dados gratuito e que contassem com API’s para configuração. Com estes requisitos foi escolhido o banco de dados Influx DB para o armazenamento das informações em formato de séries temporais. Para otimização da base de dados optou-se pela utilização da política de retenção de dados de 6 meses, assim o próprio banco realiza a rotação dos dados para não sobrecarregar o sistema. Para o banco de dados relacional considerou-se a familiaridade com as configurações de segurança, instalação e facilidade de manipulação em relação as API’s disponibilizadas, assim escolheu-se o MySql. Para otimização do potencial do banco relacional desenvolveu-se o relacionamento apresentado na Figura 24.

O *frontend* da aplicação utilizou do mesmo padrão de desenvolvimento dos serviços. O serviço retorna páginas web para cada *end-point* acessado. A interface *web* foi desenvolvida integrando ferramentas existentes de monitoria e códigos

HTML (*Hyper Text Markup Language* – do inglês Linguagem de Marcação de Hipertexto), CSS (*Cascading Style Sheets* – do inglês Folhas de Estilo Cascata*) e* JavaScript para realizar o relacionamento delas. O Grafana foi a principal ferramenta de monitoramento utilizada, ele permite a consulta e visualização de métricas e *logs* a partir da conexão com o banco de dados. A integração foi realizada a partir do acoplamento embutido utilizando HTML e a configuração dos *dashboards* pela API Rest disponibilizada pela ferrementa (GRAFANA, 2021).

As Figuras 25, 26 e 27 apresentam as telas de interface web do usuário.

Figura 25 - Página de login

Fonte: Elaborado pelo autor

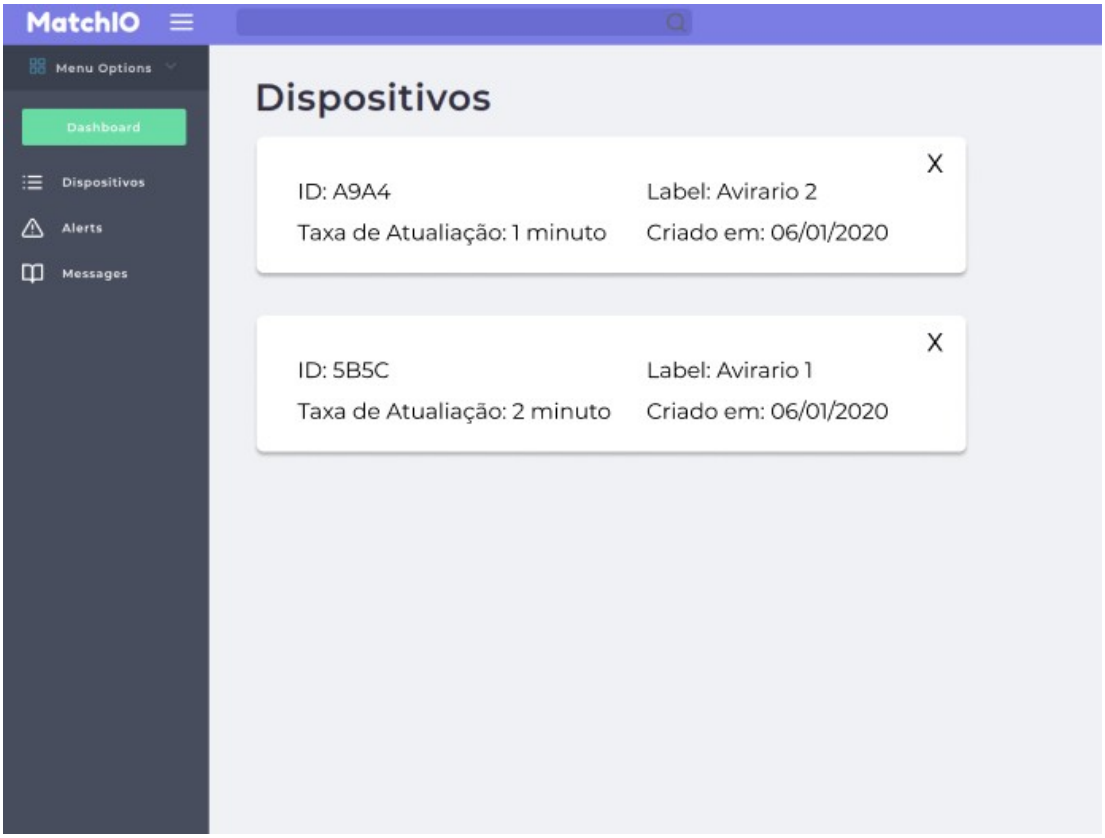
Os itens destacados na Figura 26 são funcionalidades e informações úteis para o usuário. O item 1 apresenta uma barra de consulta para acesso rápido aos dispositivos. O item 2 destaca as camadas de agrupamento dos dispositivos com a opção de remover para diminuir a quantidade de informação no item 3, o *dashboard.* O item 4 mostra os títulos dos gráficos. O item 5 apresenta o botão para adição de um novo dispositivo.

Figura 26 - Página principal

Fonte: Elaborado pelo autor

A Figura 27 apresenta a tela de controle e configuração dos dispositivos, listando todos os dispositivos cadastrados para o usuário, destacando os pontos principais: ID; Taxa de atualização da leitura do nó e data de criação.

Figura 27 - Página de login



Fonte: Elaborado pelo autor

A infraestrutura necessária para rodar as aplicações deveria suportar as ferramentas escolhidas de forma nativa. Com isso as opções de mercado se limitaram a uma: sistemas operacionais baseados em Linux. Windows Server e macOS Server utilizam virtualizações de Kernel Linux para rodar a ferramenta Docker. Assim a otimização do processo é utilizar diretamente alguma distribuição Linux. Dentre as opções presentes escolheu-se o Ubuntu, pela maior familiaridade do projetista. Tratando de máquinas virtuais próprias o provedor é independente, podendo utilizar serviços *on premise* ou provedores *cloud*. Por motivos de praticidade optou-se pelo fornecimento de máquinas virtuais no provedor Google Cloud Platform. A máquina escolhida utiliza a imagem ubuntu-1604-xenial-v20201210, nomenclatura e2-micro e hardware 2 vCPUs com 1 GB de memória. Contém armazenamento de 20 Gb e está localizada no parque de Nevada nos Estados Unidos.

A utilização de uma imagem limpa de sistema operacional garante grande liberdade de uso, com isso desenvolveu-se um *script* em Shell Script que realiza os passos de liberação do *firewall*, verificação de conectividade com a internet, instalação dos pacotes necessários, download e *build* das imagens dos serviços, inicialização dos serviços e banco de dados e por fim a configuração das ferramentas EMQX, Grafana, MySql e Influx DB para o correto funcionamento via API’s REST e chamadas curl, conforme apresentado na Figura 28. Para padronização das redes, imagens e configurações dos serviços utilizou-se o arquivo “docker-compose” com escrita em yaml para definição de todas as variáveis, como apresentado na Figura 28.

Figura 28 - Arquivo docker-compose.yml

docker-compose.yml /media/felipe/DATA12/Faculdade/11_Semestre/TCC_I/Repositorio-GitHub/TCC/Project/web/docker-compose.yml

```yaml
version: '3.7'
services:
  database:
    image: mysql:8
    container_name: database
    ports: ...
    volumes: ...
    environment: ...
    deploy: ...

  influxdb:
    image: influxdb:1.8.2
    container_name: influxdb
    ports: ...
    volumes: ...

  grafana: ...

  emqx: ...

  device_service:
    image: device_service
    container_name: device_service
    build:
      context: ./services/device-service
      dockerfile: Dockerfile
    ports:
      - "7080:7080"

  user_service: ...

  login_service: ...

  mqtt-parser: ...

  mqtt-parser: ...

  frontend-matchio: ...
```

Fonte: Elaborado pelo autor

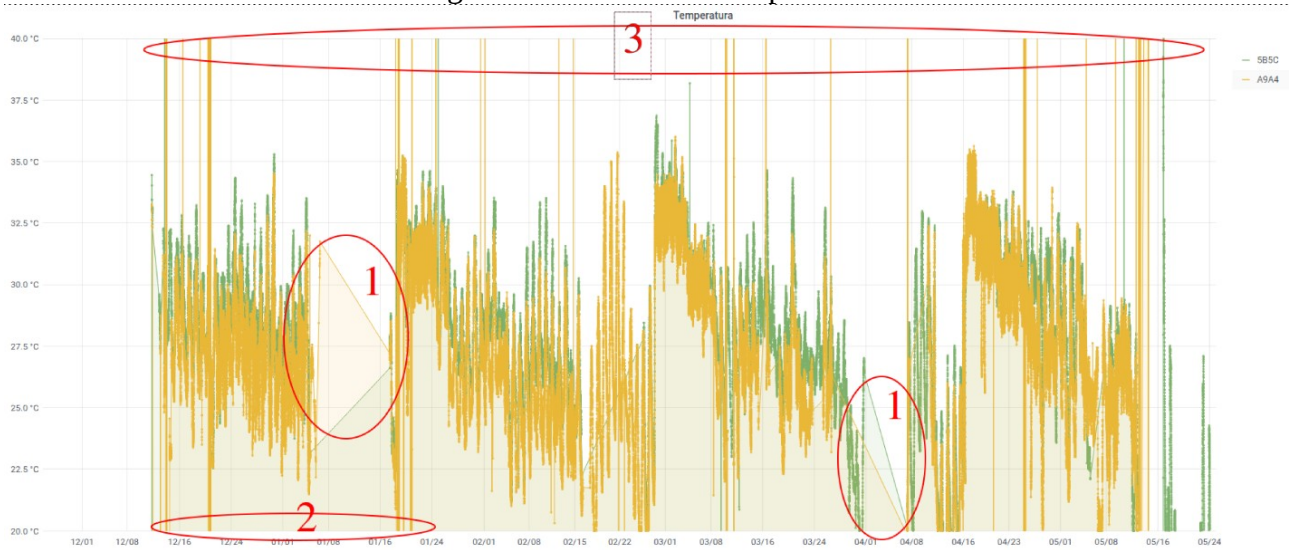
Com isto, finda-se o sistema proposto, onde foi apresentado o sistema de forma detalhada junto com sua implementação. Desta forma, no capítulo seguinte, serão apresentados os resultados obtidos durante e pós o período de implementação com a análise da base de dados.

5 ANÁLISE DE RESULTADOS

Com a estrutura para o estudo de caso apresentada na Figura 18 instalada, o protótipo foi analisado por um período de 6 meses, com intervalos de funcionamento por questões de regras de negócio do usuário. Com a base de dados populada foram analisados os seguintes itens: quantidade de perdas de envios em amostras de tempo; leitura de sensores inválidas; alterações na potência do sinal; alteração de pacotes em mensagens recebidas com sucesso.

Os resultados finais podem ser analisados diretamente na tela de controle do usuário. Como mostra na Figura 29 de forma ampliada.

Figura 29 - Leitura de Temperatura



Fonte: Elaborado pelo autor

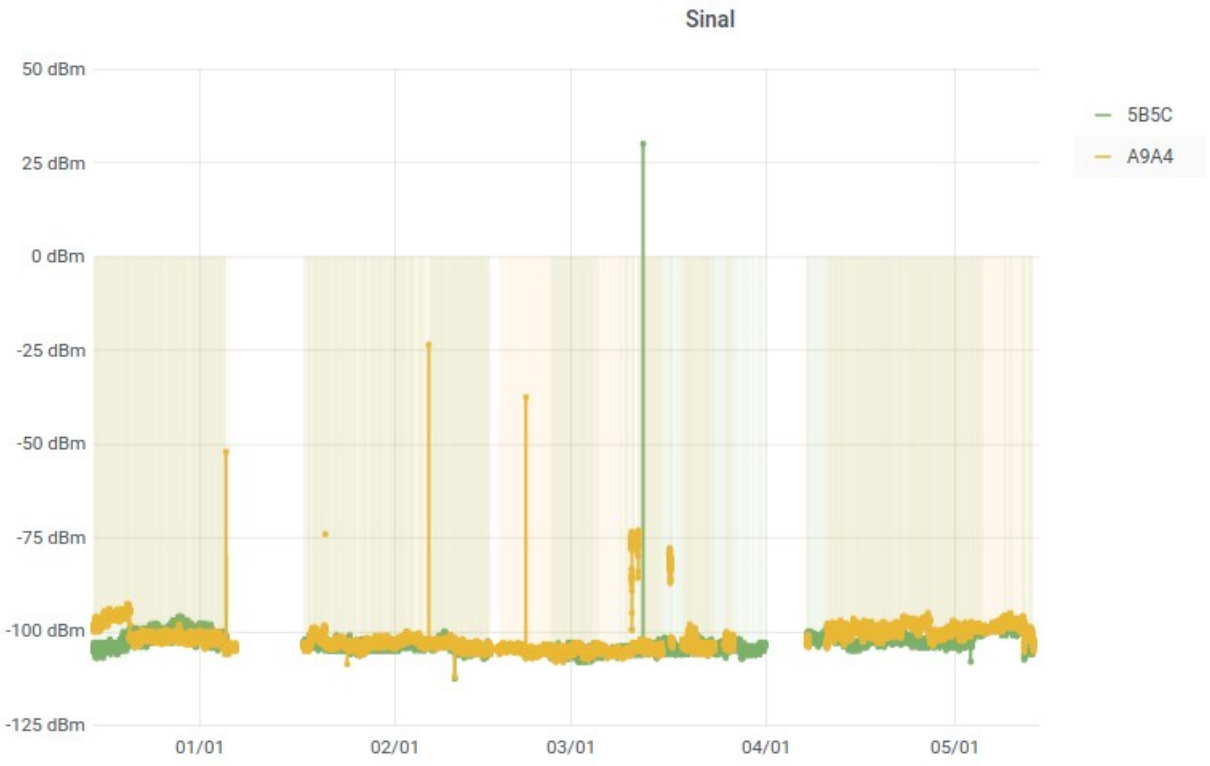
A partir da análise geral dos valores de temperatura armazenados é possível visualizar na Figura 29 os itens 1, 2 e 3. O item 1 representa as interrupções de funcionamento por necessidades do usuário, com intervalos não padronizados. Os itens 2 e 3 apresentam a falha na leitura do transdutor DS18B20 comentada no capítulo 3. A Figura 30 apresenta uma pequena amostra da variedade de identificadores falhos recebidos pelo *gateway* resultantes de interferências no sinal. No total houve 192 identificadores inválidos aprovados pela comunicação LoRa, onde apenas dois eram identificadores válidos.

Figura 30 - Variações de identificadores recebidos

Fonte: Elaborado pelo autor

Na segunda análise, considerando a potência de sinal ao longo dos dias apresentou estabilidade, como mostra a Figura 31, resultando em oscilações médias de 3 dB, desconsiderando as falhas de comunicação apresentadas na Figura 29, itens 2 e 3, estando acima dos -120 dBm recomendados no capítulo 3.

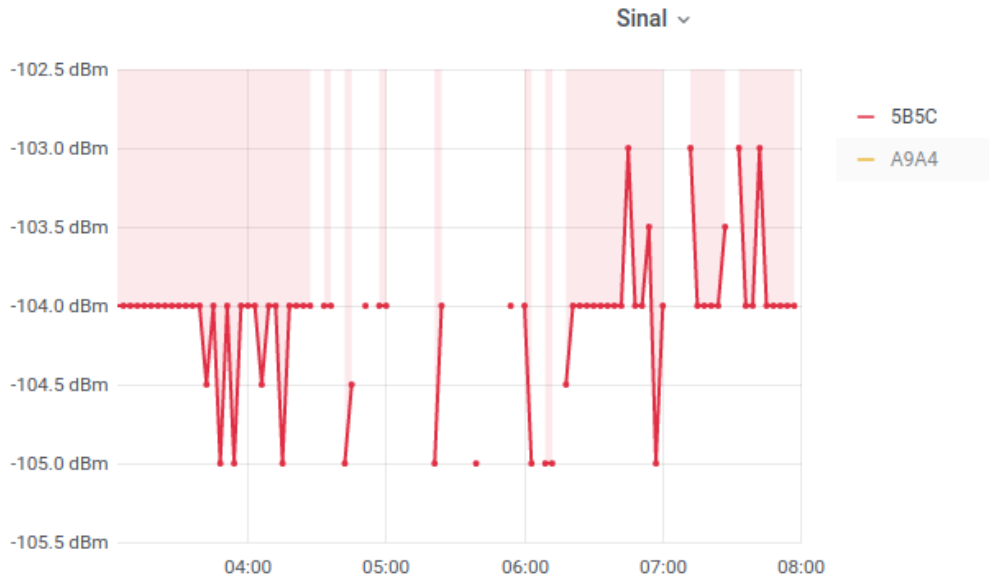
Figura 31 - Gráfico potência do sinal versus tempo



Fonte: Elaborado pelo autor

Analisando os gráficos em uma escala reduzida, considerando as taxas de atualização de aproximadamente um minuto, é possível visualizar as interrupções geradas pelas perdas de comunicação, como apresentado na Figura 32.

Figura 32 - Gráfico sinal em escala reduzida



Fonte: Elaborado pelo autor

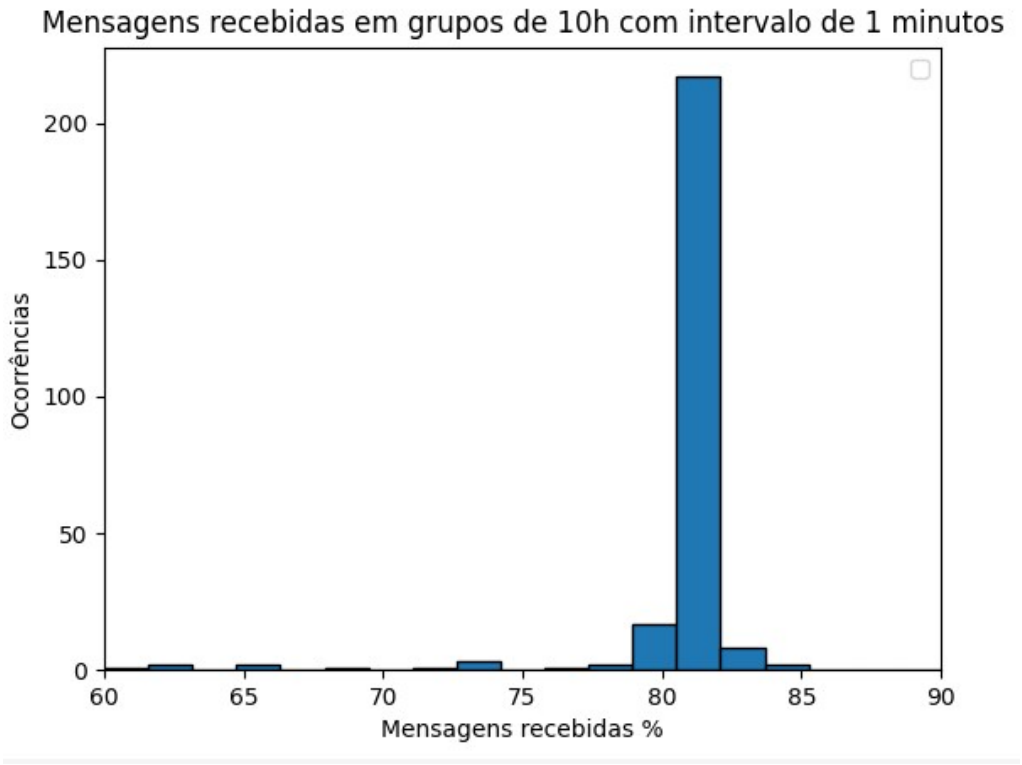
Considerando intervalos de 10 horas ao longo dos seis meses de dados, conforme descritos no Capítulo 3, com comunicação de aproximadamente um e dois minutos, dependendo do nó, é obtido o total de 197493 mensagens comunicadas com sucesso, do nó ao banco de dados, de 234900 mensagens em um ambiente ideal. O número obtido resulta na proporção de 84,07% de mensagens entregues, conforme apresenta o Quadro 6. O Quadro também apresenta um resultado superior do nó que enviou mensagens a cada dois minutos resultando na proporção de 89,37%, em relação a 81,3% quando enviadas a cada um minuto. A Figura 33 apresenta um gráfico para melhor visualização das proporções atingidas.

Figura 33 - Gráfico mensagens recebidas x não recebidas

Fonte: Elaborado pelo autor

As Figuras 34, 35 e 36 apresentam os histogramas das mensagens recebidas ao longo do período, distribuídas em %. É possível notar uma diferença significativa quando comparados os nós com atualização de um e dois minutos, conforme apresentado na Figura 36.

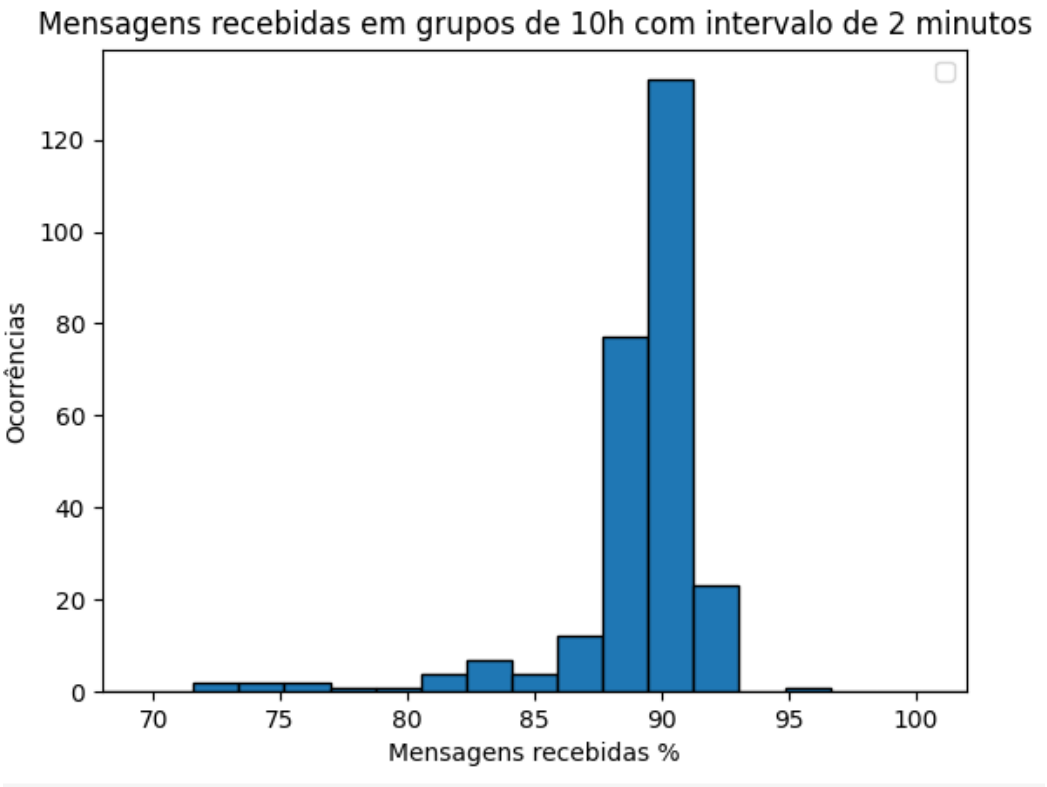
Figura 34 - Histograma de mensagens recebidas por grupo com intervalo de 1 minuto



Fonte: Elaborado pelo autor

A Figura 35 apresenta a distrubuição em formato de histograma do nó com taxa de atualização de 2 minutos, onde é possível notar a concentração dos grupos na faixa de 90%.

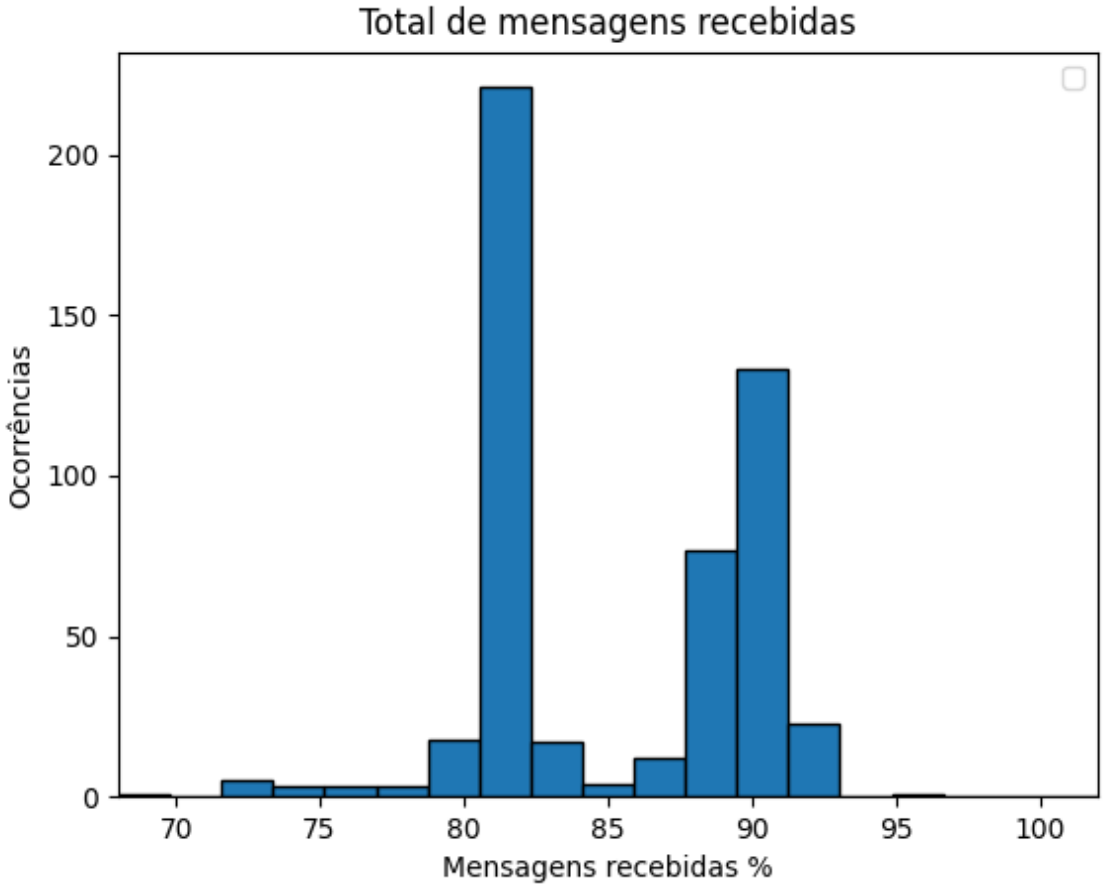
Figura 35 - Histograma de mensagens recebidas por grupo com intervalo de 2 minutos



Fonte: Elaborado pelo autor

É possível visualizar na Figura 36 a distribuição total das mensagens recebidas em grupos de 10 horas, com a distinção dos picos dos envios pelos dispositivos com taxa de atualização de um minuto e dois minutos.

Figura 36 - Histograma total de mensagens recebidas



Fonte: Elaborado pelo autor

Diversos fatores influenciam na divergência das mensagens entregues com sucesso. Fatores como QoS 0, internet via satélite de baixa velocidade e estabilidade, faltas de energia na região afetando o *gateway* e o roteador WiFi. O resultado obtido se mostrou satisfatório para o uso proposto neste trabalho.

Quadro 6 - Tabela de amostra dos resultados

Nó ID	Intervalo	Esperado	Recebido	%
5B5C	2021-03-01 16:00:00 / 2021-03-02 02:00:00	300	266	88,67%
5B5C	2020-12-26 00:00:00 / 2020-12-26 10:00:00	300	268	89,33%
5B5C	2020-12-29 12:00:00 / 2020-12-29 22:00:00	300	266	88,67%
5B5C	2020-12-29 12:00:00 / 2020-12-30 22:00:00	300	258	86,00%
5B5C	2021-02-15 12:00:00 / 2021-02-15 22:00:00	300	270	90,00%
A9A4	2021-02-15 12:00:00 / 2021-02-15 22:00:00	600	493	82,17%
A9A4	2020-12-29 12:00:00 / 2020-12-29 22:00:00	600	492	82,00%
A9A4	2021-04-17 14:00:00 / 2021-04-18 00:00:00	600	495	82,50%
A9A4	2021-04-17 14:00:00 / 2021-04-18 00:00:00	600	490	81,67%

Fonte: Elaborado pelo Autor

Após os testes de usabilidade realizado com os usuários as médias das respostas são apresentadas no Quadro 7.

Quadro 7 - Média de notas do questionário

Pergunta	**Nota**
Em sua percepção, qual o nível de facilidade de configuração do *gateway*?	3,5
Em sua percepção, qual o nível de facilidade de configuração do nó?	4
Em sua percepção, qual o nível de facilidade de interação com o nó?	3,5
Em sua percepção, qual o nível de facilidade de cadastro de dispositivos na interface de usuário?	5
Em sua percepção, qual o nível de facilidade de interpretação dos resultados pela interface de usuário?	4

Fonte: Elaborado pelo Autor

Após o questionário destaca-se a facilidade de uso com a interface, porém a mesma não se apresentou na configuração do *gateway* e do nó, ainda estando acima do ideal. Com a média de 4 de 5 na usabilidade do sistema proposto é considerado um resultado satisfatório dado que todos os usuários conseguiram realizar a implementação do sistema com sucesso.

6 CONCLUSÃO

Com a disseminação do paradigma IoT e o contato cada vez mais próximo no cotidiano das pessoas com dispositivos inteligentes como, smartfones, smart TVs, smart Speaker, sensores, atuadores, entre outros, em ambientes e momentos como, lazer a domicílio, trabalho em indústrias e comércios, viagens, entre outros, acredita-se que a IoT se tornará tão presente no dia a dia das pessoas quanto a internet. Dessa forma, o desenvolvimento de aplicações que tornem as interações mais humanas são cada vez mais fundamentais, seja para acesso, gerenciamento ou instalação de sistemas inteligentes através da rede, ou ferramentas completas, que além da administração da rede os usuários possam realizar todas as configurações necessárias para a adaptação do sistema de forma a resolver suas peculiaridades e problemas específicos, através de controles de voz, controles manuais ou até mesmo movimentos, que após sua identificação é executada alguma ação. Assim como é atualmente a Internet, repleta de soluções criadas para permitir que usuários não especialistas possam criar suas próprias soluções, como por exemplo, Blogs, WordPress, Wix, entre outros.

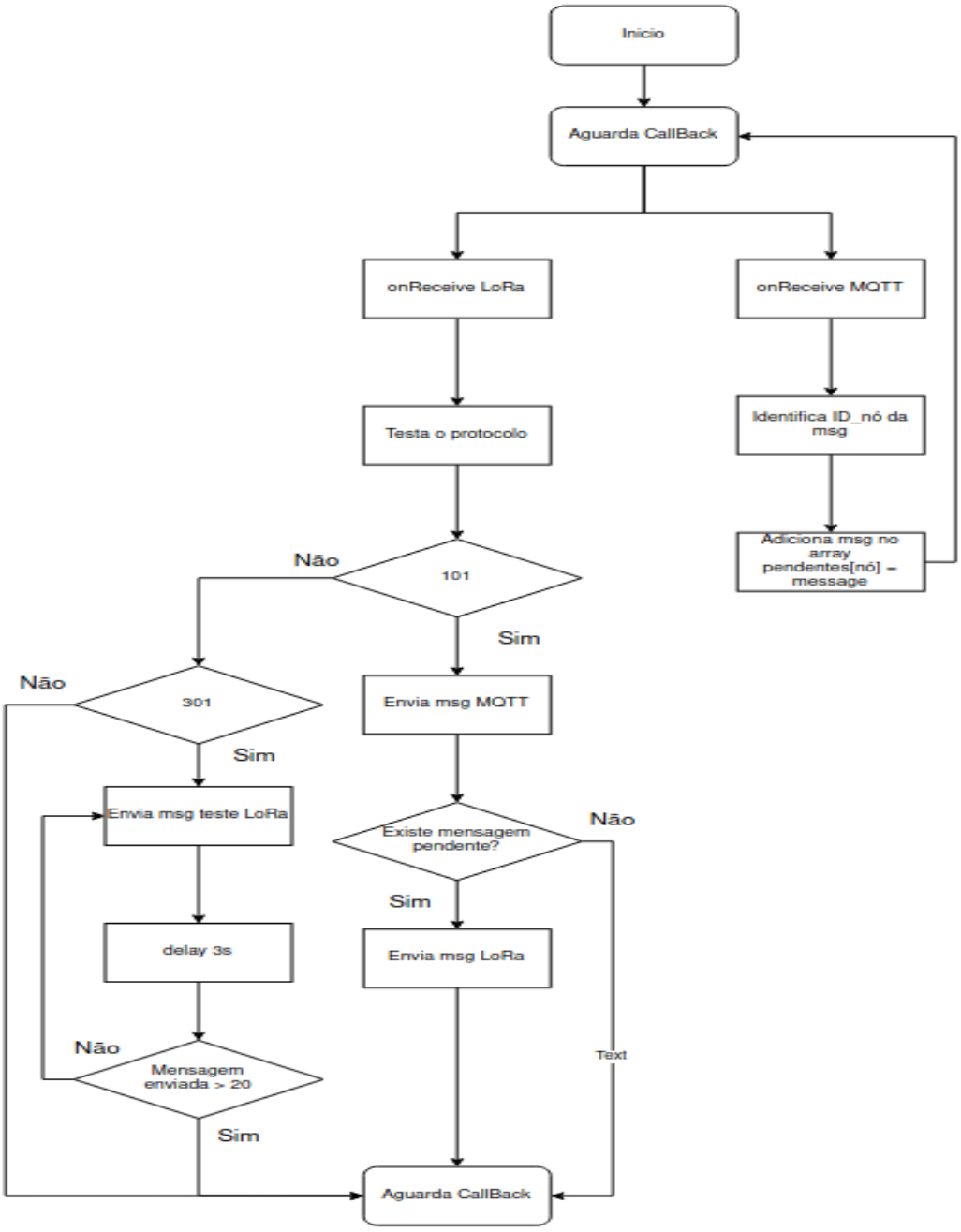
Nesse sentido, este trabalho apresentou uma solução completa, explorando e desenvolvendo soluções para todas as áreas utilizadas por um sistema de IoT. Cada etapa descrita neste trabalho é importante para o correto funcionamento e a organização da solução proposta, como: A escolha do *hardware* para redução de custos materiais; o desenvolvimento do *firmware* com interface visual disponível para melhor compreensão do funcionamento do sensor para o usuário, junto com o protocolo de comunicação para configuração da rede e seus dispositivos; a configuração dos serviços de comunicação e integração entre os dispositivos de campo e a infraestrutura em nuvem, junto com a arquitetura dos serviços, banco de dados e camada de segurança; o desenvolvimento de uma interface amigável e de fácil utilização, e, por fim, a configuração de toda infraestrutura para rodar como código, facilitando a configuração e o início do uso das novas instalações. A análise de resultados comprovou o funcionamento satisfatório em relação as mensagens entregues com total de 84,01% no período de 6 meses e a usabilidade do sistema com nota 4 em 5, sendo considero de fácil utilização. Com isso, pode-se concluir que o trabalho é uma solução funcional para o monitoramento de ambientes

industriais e rurais que reduz os custos de instalação e operação, facilitando a entrada de novos usuários ao mercado de IoT.

Trabalhos futuros a partir desse podem ser desenvolvidos com o aprimoramento das camadas de segurança, implementações de ferramentas de qualidade de serviço, algoritmos de verificação de integridade dos dados transmitidos e o desenvolvimento e configuração para o uso e controle de acionadores nos dispositivos de ponta. Este trabalho possibilita o desenvolvimento de produtos baseados nesta solução, integrando todas as etapas necessárias para o funcionamento do sistema, hospedando os softwares de alto nível em servidores internos as indústrias ou servidores em nuvens públicas, de acordo com as restrições de compartilhamento de dados do usuário. Ainda, futuros estudos que visam a popularização de tecnologias IoT com a usabilidade e acessibilidade ao usuário, a integração de sistema de ponta a ponta analisando, desde o hardware até a interface de usuário, onde todos os protocolos e comunicações estão relacionados para um único fim, e, por fim, pesquisas que buscam a análise de taxa de rececbimento das comunicações podem utilizar este trabalho como base.

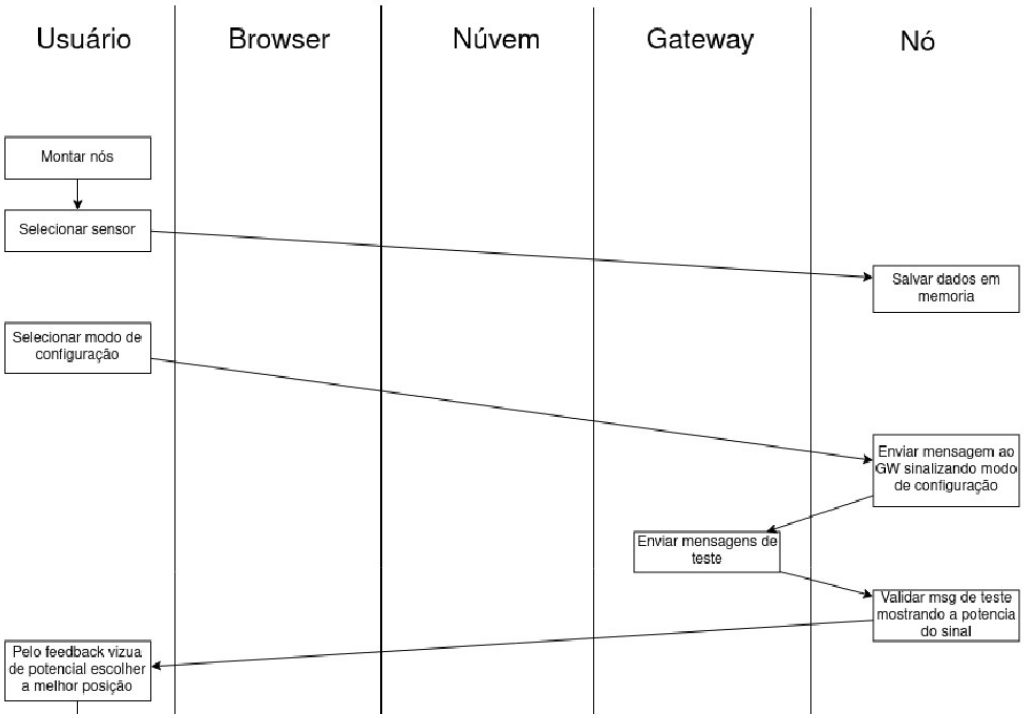
7 APÊNDICE

Apêndice 1 - Fluxograma do nó



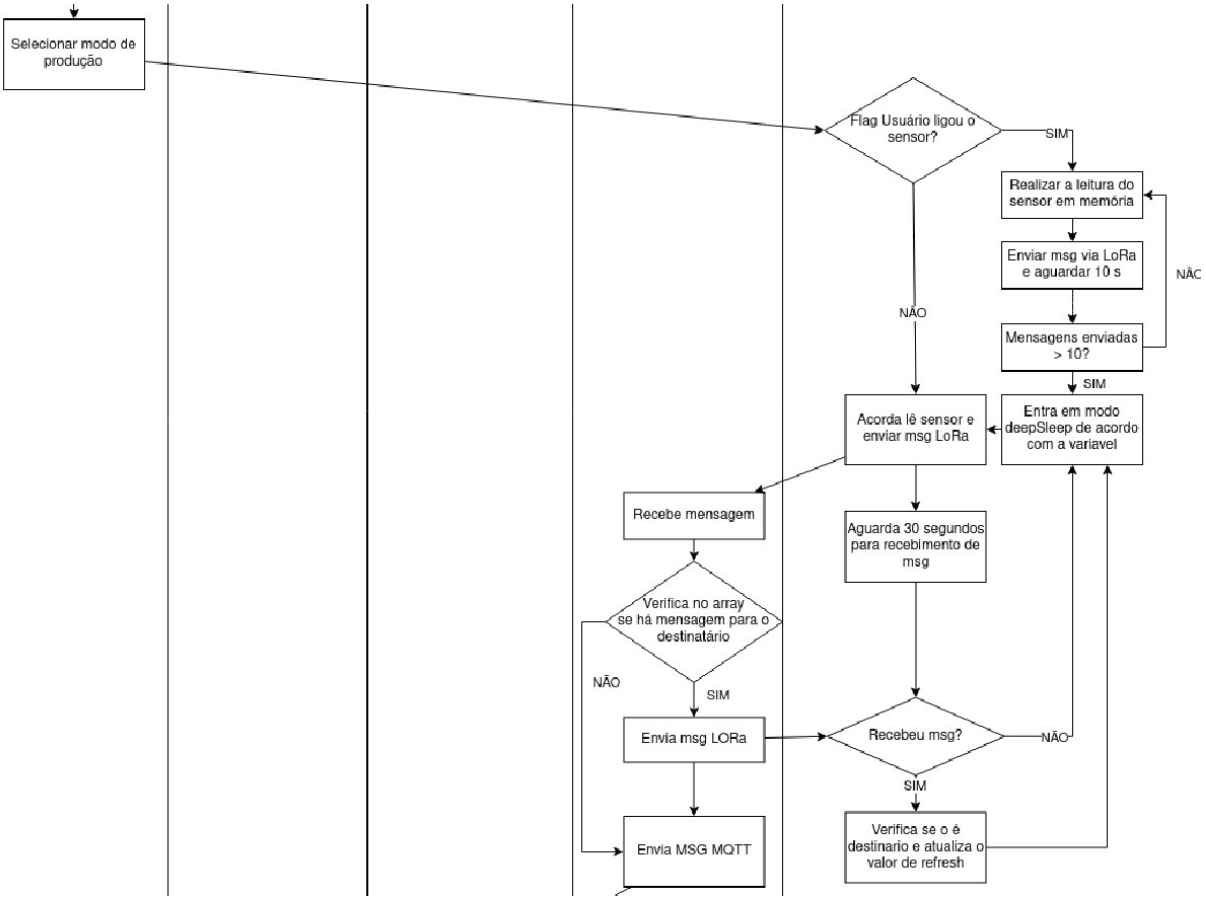
Fonte: Elaborado pelo autor

Apêndice 2 - Fluxograma do projeto



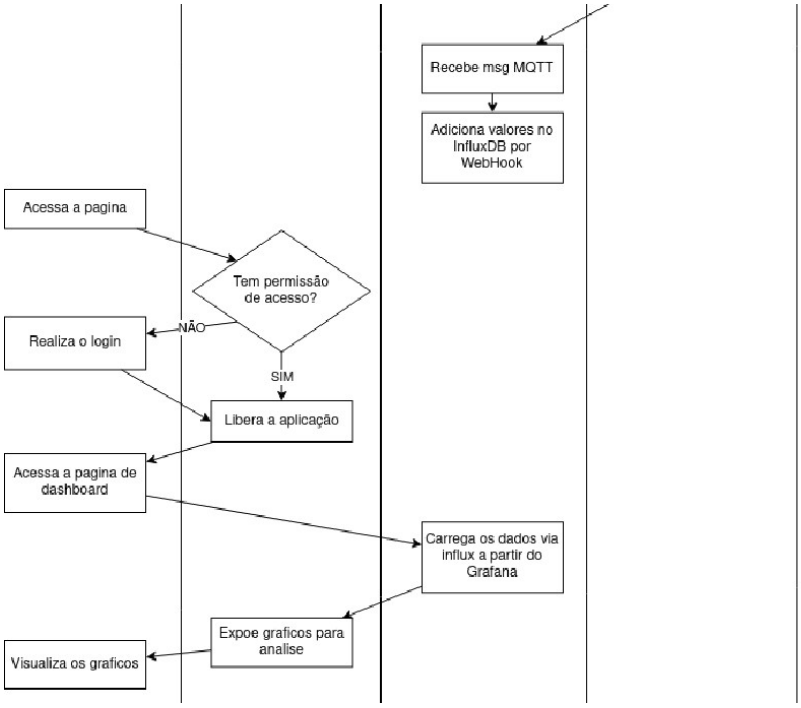
Fonte: Elaborado pelo autor

Apêndice 3 - Continuação fluxograma do projeto



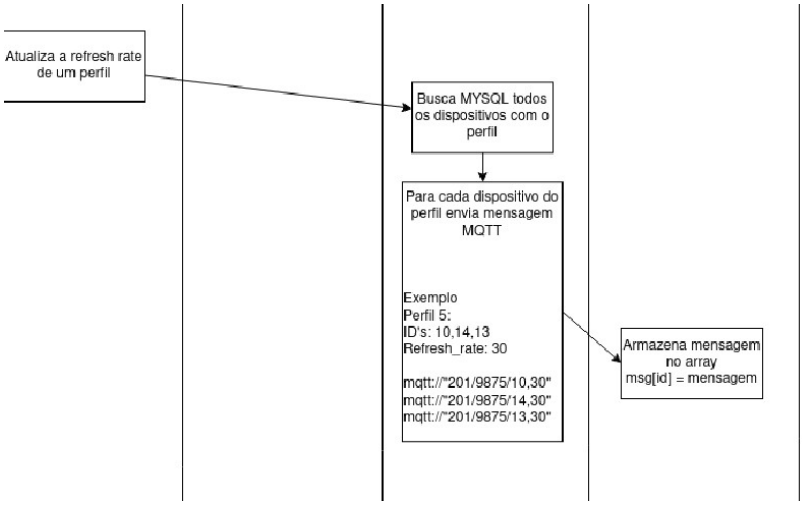
Fonte: Elaborado pelo autor

Apêndice 4 - Continuação fluxograma do projeto



Fonte: Elaborado pelo autor

Apêndice 5 - Continuação fluxograma do projeto



Fonte: Elaborado pelo autor

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023**: informação e documentação – referências – elaboração. Rio de Janeiro, 2002.

BIBLIOTECA DA UNISINOS. **Guia para Elaboração de Trabalhos Acadêmicos (Artigo e Periódico, Dissertação, Projeto, Trabalho de Conclusão de Curso e Tese)**. SãoLeopoldo, 2011. Disponível em:<http://www.unisinos.br/biblioteca/images/stories/downloads/guia_biblioteca_abnt_2011.pdf>.Acesso em: 1 jun. 2020

GITHUB HELTEC; **HelTecAutomation/ESP32_LoRaWAN**. HELTEC, [2020?]. Disponível em: https://github.com/HelTecAutomation/ESP32_LoRaWAN. Acesso em: 5 maio 2020. SHA, 501f6d82ee9fc8f6b431fbdb2b3913643b50b495.

HELTEC. **WiFi LoRa 32 (V2)**. HELTEC, [2020?]. Disponível em: https://heltec.org/project/wifi-lora-32/. Acesso em: 5 maio 2020.

LORA ALLIANCE. What is the LoRaWAN® Specification?. LORA ALLIANCE, [2020?]. Disponível em: https://lora-alliance.org/about-lorawan. Acesso em: 19 maio 2020.

MCKINSEY, **Internet of Things. We help clients unlock value by digitizing the physical world**. MCKINSEY [2019]. Disponível em: https://www.mckinsey.com/featured-insights/internet-of-things/how-we-help-clients. Acesso em 20 maio 2021

SEMTECH. **What is LoRa®?**. SEMTECH, [2019?]. Disponível em: https://www.semtech.com/lora/what-is-lora. Acesso em 5 maio 2019.

XIAO, Perry. **Designing Embedded Systems and the Internet of Things (IoT) with the ARM® MbedTM**. 1. Ed. Hoboken: Wiley, 2018. E-book.

GUILLERMO, J. C.; GARCÍA-CEDEÑO, A.; RIVAS-LALALEO, D.; HUERTA, M.; CLOTET, R. IoT Architecture Based on Wireless Sensor Network Applied to Agricultural Monitoring: a case of study of cacao crops in ecuador. **Advances in Intelligent Systems and Computing**, [S.l.], v. 893, p. 42–57, 2019.

HUANG, A.; HUANG, M.; SHAO, Z.; ZHANG, X.; WU, D.; CAO, C. A practical marine wireless sensor network monitoring system based on LoRa and MQTT. **2019 2nd International Conference on Electronics Technology, ICET 2019**, [S.l.], p. 330–334, 2019.

KE, K. H.; LIANG, Q. W.; ZENG, G. J.; LIN, J. H.; LEE, H. C. Demo abstract: a lora wireless mesh networking module for campus-scale monitoring. **Proceedings - 2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN 2017**, [S.l.], p. 259–260, 2017.

NOBREGA, L.; TAVARES, A.; CARDOSO, A.; GONCALVES, P. Animal monitoring based on IoT technologies. **2018 IoT Vertical and Topical Summit on Agriculture - Tuscany, IOT Tuscany 2018**, [S.l.], n. February 2019, p. 1–5, 2018.

RIZZI, M.; FERRARI, P.; FLAMMINI, A.; SISINNI, E.; GIDLUND, M. Using LoRa for industrial wireless networks. **IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS**, [S.l.], p. 1–4, 2017.

USMONOV, M.; GREGORETTI, F. Design and implementation of a LoRa based wireless control for drip irrigation systems. **2017 2nd International Conference on Robotics and Automation Engineering, ICRAE 2017**, [S.l.], v. 2017-December, p. 248–253, 2018.

NOR, R. F. A. M.; ZAMAN, F. H.; MUBDI, S. Smart traffic light for congestion monitoring using LoRaWAN. **2017 IEEE 8th Control and System Graduate Research Colloquium, ICSGRC 2017 - Proceedings**, [S.l.], n. August, p. 132–137, 2017.

GRIÓN, F. J.; PETRACCA, G. O.; LIPUMA, D. F.; AMIGÓ, E. R. LoRa network coverage evaluation in urban and densely urban enviroment simulation and validation tests in Autonomous City of Buenos Aires. **2017 17th Workshop on Information Processing and Control, RPIC 2017**, [S.l.], v. 2017-January, p. 1–5, 2017.