



**UNIVERSIDADE FEDERAL DO VALE DO SÃO FRANCISCO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO**

SÉRGIO MURILO GONZAGA SILVA

**DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO DE
MATERIAIS PARTICULADOS PARA A REGIÃO DO POLO GESSEIRO DO
ARARIPE UTILIZANDO LORAWAN**

JUAZEIRO - BA

2023

**UNIVERSIDADE FEDERAL DO VALE DO SÃO FRANCISCO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO**

SÉRGIO MURILO GONZAGA SILVA

**DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO DE
MATERIAIS PARTICULADOS PARA A REGIÃO DO POLO GESSEIRO DO
ARARIPE UTILIZANDO LORAWAN**

Trabalho de Conclusão de Curso apresentado
como requisito parcial para obtenção do título
de Bacharel em Engenharia de Computação,
pela Universidade Federal do Vale do São
Francisco.

Orientador: Prof. Dr. Max Santana Rolemberg
Farias

Coorientador: Prof. Esp. Eziom Alves de Oliveira

JUAZEIRO - BA

2023

S586d	<p>Silva, Sérgio Murilo Gonzaga Desenvolvimento de um sistema de monitoramento de materiais particulados para a região do polo gesseiro do Araripe utilizando LoRaWAN / Sérgio Murilo Gonzaga Silva - Juazeiro - BA, 2023. xv; 127 f.: il ; 29 cm.</p> <p>Trabalho de Conclusão de Curso (Graduação em Engenharia da Computação) - Universidade Federal do Vale do São Francisco, Campus Juazeiro, 2023.</p> <p>Orientador: Prof. Dr. Max Santana Rolemberg Farias.</p> <p>1. Sistema de computação . 2. Protocolo LoRaWAN. I. Título. II. Farias, Rolemberg Max Santana. III. Universidade Federal do Vale do São Francisco.</p> <p>CDD 005.42</p>
-------	---

**UNIVERSIDADE FEDERAL DO VALE DO SÃO FRANCISCO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO**

FOLHA DE APROVAÇÃO

SÉRGIO MURILO GONZAGA SILVA

**DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO DE
MATERIAIS PARTICULADOS PARA A REGIÃO DO POLO GESSEIRO DO
ARARIPE UTILIZANDO LORAWAN**

Trabalho de Conclusão de Curso apresentado
como requisito parcial para obtenção do título
de Bacharel em Engenharia de Computação,
pela Universidade Federal do Vale do São
Francisco.

Orientador: Prof. Dr. Max Santana Rolemberg
Farias

Coorientador: Prof. Esp. Eziom Alves de Oliveira

Aprovado em: 6 de março de 2023

Banca Examinadora

**Prof. Max Santana Rolemberg Farias, Doutor,
Universidade Federal do Vale do São
Francisco**

**Prof. Fábio Nelson de Souza Pereira, Mestre,
Universidade Federal do Vale do São
Francisco**

**Prof. Washington Pereira Lacerda, Graduado,
IF Sertão Pernambuco**

AGRADECIMENTOS

Dedico esta conquista aos meus pais.

“Not all those who wander are lost”

J.R.R. Tolkien

RESUMO

Os processos envolvidos na produção de gesso na região do Araripe causam emissão aérea de materiais particulados que, quando inalados, podem levar a uma série de danos à saúde das pessoas que vivem e/ou trabalham na região, além de provocar impactos ambientais relacionados à deterioração da qualidade do solo e comprometimento de reservas hídricas. Em contextos como este, o monitoramento da qualidade do ar se faz necessário para o acompanhamento de tendências de médio ou longo prazo, para verificar a eficácia de programas de controle de qualidade do ar e avaliar necessidade de aprimoramentos dos mesmos. Este trabalho apresenta uma proposta de arquitetura e desenvolvimento de um sistema IoT para monitoramento da qualidade do ar usando as tecnologias LoRa e LoRaWAN, com foco na detecção da presença de materiais particulados MP₁₀ e MP_{2,5} em suspensão. Em ambiente controlado, o projeto testa e confirma a viabilidade do uso das referidas tecnologias como plataforma para uma rede de monitoramento de materiais particulados potencialmente aplicável ao estudo desses poluentes no cenário do Polo Gesseiro do Araripe.

Palavras-chave: qualidade do ar; materiais particulados; internet das coisas; LoRa; LoRaWAN;

ABSTRACT

The processes involved in the production of gypsum in the Araripe region cause air emission of particulate matter that, when inhaled, can lead to a series of damages to the health of people who live and/or work in the region. In addition, it has caused environmental impacts related to deterioration of soil quality and commitment of hydric resources. In contexts like this, air quality monitoring is necessary to monitor medium or long-term trends, to verify the effectiveness of air quality control programs and to validate the need for their improvements. This work presents a proposal for the architecture and development of an IoT system for monitoring air quality using LoRa and LoRaWAN technologies, focusing on detecting the presence of MP₁₀ and MP_{2,5} particulate matter in suspension. In controlled environment, this project tests and corroborate the feasibility of using these technologies as a platform for a monitoring network of particulate matter potentially applied in the study of those pollutants in Araripe plaster production hub.

Keywords: air quality; particulate matter; internet of things; LoRa; LoRaWAN;

LISTA DE ILUSTRAÇÕES

Figura 1 – Região do Polo Gesseiro do Araripe	18
Figura 2 – Proporção de diâmetros de materiais particulados	22
Figura 3 – Categorização de equipamentos LPWAN com relação a largura de banda e alcance de transmissão	29
Figura 4 – Divisão regional do globo terrestre pela ITU	30
Figura 5 – Variação da frequência no tempo na modulação LoRa/CSS	32
Figura 6 – Transmissão de símbolos na modulação LoRa/CSS	32
Figura 7 – Efeitos da variação do fator de espalhamento (SF)	34
Figura 8 – Organização da pilha LoRa/LoRaWAN	36
Figura 9 – Arquitetura típica na utilização protocolo LoRaWAN	38
Figura 10 – Interação entre as camadas em uma arquitetura LoRAWAN	38
Figura 11 – Obtenção das chaves de sessão no modo OTAA	40
Figura 12 – Obtenção das chaves de sessão no modo ABP	41
Figura 13 – Classes de dispositivos LoRaWAN	42
Figura 14 – Dispositivos LoRaWAN classe A	43
Figura 15 – Dispositivos LoRaWAN classe B	43
Figura 16 – Dispositivos LoRaWAN classe C	43
Figura 17 – Placa Heltec WiFi LoRa 32	44
Figura 18 – Sensor DHT22 (ou AM2302)	46
Figura 19 – Sensor de materiais particulados PMS5003	48
Figura 20 – Gateway LoRaWAN Dragino LPS8	51
Figura 21 – Camadas de abstração propostas pelas definições LMIC da MCCI	56
Figura 22 – Disposição geográfica de <i>gateways</i> públicos da TTN	59
Figura 23 – Console do TTS <i>Community Edition</i>	60
Figura 24 – Arquitetura do sistema de monitoramento proposta	63
Figura 25 – Diagrama da montagem do ensaio de integração do DHT22	65
Figura 26 – Montagem real do ensaio de integração do DHT22	65
Figura 27 – Arquivo <i>platformio.ini</i> para ensaio com o DHT22	66
Figura 28 – Monitoramento serial das leituras realizada no ensaio com o DHT22	66
Figura 29 – Esquema do divisor de tensão a ser utilizado	68
Figura 30 – Ensaio de integração do PMS5003	69
Figura 31 – Montagem real do ensaio de integração do PMS5003	69
Figura 32 – Arquivo <i>platformio.ini</i> para ensaio com o PMS5003	70
Figura 33 – Monitoramento serial das leituras realizada no ensaio com o PMS5003	71
Figura 34 – Montagem do protótipo de hardware	72

Figura 35 – Montagem real do ensaio de integração simultânea do DHT22 e PMS5003	73
Figura 36 – Arquivo <i>platformio.ini</i> para ensaio unificado	73
Figura 37 – Monitoramento serial das leituras realizada no ensaio unificado	74
Figura 38 – Grupo de configurações [<i>platformio</i>] na customização do <i>end device</i>	76
Figura 39 – Alocação reservada para identificador e chaves de sessão do modo ABP do <i>end device</i> no arquivo <i>keyfiles/lorawan-keys.h</i>	77
Figura 40 – Grupo de configurações [<i>common</i>] e [<i>esp32</i>] na customização do <i>end device</i>	77
Figura 41 – Grupo de configurações [<i>mcci_lmic</i>] na customização do <i>end device</i>	79
Figura 42 – Grupo de configurações [<i>env:heltec_wifi_lora_32_v2</i>] na customização do <i>end device</i>	80
Figura 43 – Indicadores do arquivo <i>src/LMIC-node.cpp</i> para inclusão de código do usuário	81
Figura 44 – Menu inicial do Dragino LPS8, acessível através de um navegador	83
Figura 45 – Opções do Dragino LPS8 relacionadas ao WiFi	84
Figura 46 – Opções do Dragino LPS8 relacionadas ao LoRa	84
Figura 47 – Opções do Dragino LPS8 relacionadas ao LoRaWAN	85
Figura 48 – Telas de escolha de cluster e cadastro no TTN	86
Figura 49 – Registro de <i>gateway</i> no console do TTN	87
Figura 50 – Criação da abstração de uma aplicação no TTN	88
Figura 51 – Registro do <i>end device</i> no console do TTN	89
Figura 52 – Provisão de identificação e chaves de sessão do modo ABP no console do TTN	90
Figura 53 – <i>Payload Formatter</i> para transformação dos pacotes enviados pelo <i>end device</i>	91
Figura 54 – Indicação de conexão do <i>gateway</i> registrado no console do TTN	92
Figura 55 – Atividades de <i>uplink</i> pelo <i>sensorar-application-1</i> no console do TTN	93
Figura 56 – Caminho para a ativação do recurso de <i>Storage Integration</i> no console do TTN	95
Figura 57 – Caminho para a criação de uma chave de API no console do TTN	95
Figura 58 – Aplicação <i>web</i> criada para apresentação dos dados do SensorAr	97
Figura 59 – Exemplo fictício da representação de barras com os valores do IQAr e representação de cores associadas	98

LISTA DE TABELAS

Tabela 1 – Classificações da qualidade do ar com base no IQAr	27
Tabela 2 – Efeitos estimados para cada classificação do IQAr	28
Tabela 3 – Principais características técnicas da Heltec ESP32 LoRa	45
Tabela 4 – Descrição da função dos pinos do DHT22	46
Tabela 5 – Principais características técnicas do DHT22	47
Tabela 6 – Descrição da função dos pinos do PMS5003	48
Tabela 7 – Classificações de diâmetro de MP feitas pelo PMS5003	49
Tabela 8 – Principais características técnicas do PMS5003	50
Tabela 9 – Principais características técnicas do Dragino LPS8	52

LISTA DE ABREVIATURAS E SIGLAS

ABP	Activation By Personalisation
ANATEL	Agência Nacional de Telecomunicações
AP	Access Point
API	Application Programming Interface
AQI	Air Quality Index
BW	Bandwidth
CO	Monóxido de Carbono
CONAMA	Conselho Nacional do Meio Ambiente
CR	Coding Rate
CSS	Chirp Spread Spectrum
DNPM	Departamento Nacional de Produção Mineral
FER	Forward Error Correction
gRPC	Google Remote Procedure Call
IoT	Internet of Things
IQAr	Índice de Qualidade do Ar
ISM	Industrial, Scientific and Medical
ITU	International Telecommunications Union
LMIC	LoRaWAN MAC In C
LoRa	Long Range
LoRaWAN	LoRa Wide-Area Network
LPWAN	Low-Power Wide-Area Network
MAC	Medium Access Control
MMA	Ministério do Meio Ambiente
MP _{1,0}	materiais particulados com diâmetro de 2,5 micrômetros

MP ₁₀	materiais particulados com diâmetro de 10 micrômetros
MP _{2,5}	materiais particulados com diâmetro de 2,5 micrômetros
MQTT	Message Queuing Telemetry Transport
NO ₂	Dióxido de Nitrogênio
O ₃	Ozônio Troposférico
OIT	Organização Internacional do Trabalho
OMS	Organização Mundial da Saúde
ORM	Object Relational Mapper
OTAA	Over The Air Activation
PNMA	Política Nacional do Meio Ambiente
PRONAR	Programa Nacional de Controle de Qualidade do Ar
SF	Spreading Factor
SO ₂	Dióxido de Enxofre
TP	Transmission Power
TTN	The Thing Network
TTS	The Thing Stack
USEPA	United States Environmental Protection Agency
VOC	Volatile Organic Compounds

SUMÁRIO

1	INTRODUÇÃO	16
1.1	OBJETIVO GERAL	17
1.2	OBJETIVOS ESPECÍFICOS	17
2	FUNDAMENTAÇÃO TEÓRICA	18
2.1	POLUIÇÃO DO AR NO POLO GESSEIRO DO ARARIPE	18
2.1.1	Cadeia Produtiva do Gesso e Qualidade do Ar	19
2.1.2	Consequências à Saúde Humana e ao Meio Ambiente	20
2.2	PRINCÍPIOS SOBRE QUALIDADE DO AR	22
2.2.1	Poluentes Clássicos	22
2.2.1.1	Materiais Particulados (MP ₁₀ e MP _{2,5})	22
2.2.1.2	Poluentes Gasosos (O ₃ , CO, NO ₂ e SO ₂)	23
2.2.2	Qualidade do Ar no Brasil	25
2.2.2.1	Índice de Qualidade do Ar (IQAr)	26
2.3	PADRÃO LORA	29
2.3.1	Ocupação do Espectro Eletromagnético	29
2.3.2	Técnica de Modulação Digital	31
2.3.3	Principais Parâmetros Configuráveis	33
2.3.3.1	Frequência da Portadora (<i>Carrier Frequency - CF</i>)	33
2.3.3.2	Largura de Banda (<i>Bandwidth - BW</i>)	33
2.3.3.3	Fator de Espalhamento (<i>Spreading Factor - SF</i>)	34
2.3.3.4	Taxa de Codificação (<i>Coding Rate - CR</i>)	35
2.3.3.5	Potência de Transmissão (<i>Transmission Power - TP</i>)	35
2.4	PROTOCOLO LORAWAN	36
2.4.1	Elementos de Rede e Topologia Típica	36
2.4.2	Modos de Ativação	38
2.4.2.1	OTAA (<i>Over The Air Activation</i>)	39
2.4.2.2	ABP (<i>Activation By Personalisation</i>)	40
2.4.3	Classes de Dispositivos	41
2.4.3.1	Classe A (<i>All</i>)	42
2.4.3.2	Classe B (<i>Beacon</i>)	43
2.4.3.3	Classe C (<i>Continuous</i>)	43
3	MATERIAIS E MÉTODOS	44
3.1	HARDWARE	44
3.1.1	Placa de desenvolvimento Heltec WiFi LoRa 32 v2	44
3.1.2	Sensor de Temperatura e Umidade DHT22	46
3.1.3	Sensor de Materiais Particulados PMS5003	48

3.1.4	<i>Gateway LoRaWAN Dragino LPS8</i>	51
3.2	FIRMWARE	53
3.2.1	Extensão PlatformIO para Microsoft VSCode	53
3.2.2	Integração dos Sensores DHT22 e PMS5003	53
3.2.3	Implementação de End-node LoRaWAN com o LMIC	54
3.3	SOFTWARE	57
3.3.1	<i>The Things Stack (TTS) Community Edition</i>	57
3.3.2	Ecossistema da Linguagem Python	61
3.3.2.1	<i>SQLite</i> e <i>SQLAlchemy</i>	61
3.3.2.2	<i>Pandas</i> e <i>Streamlit</i>	62
3.4	ARQUITETURA PROPOSTA	63
4	RESULTADOS E DISCUSSÃO	64
4.1	MONTAGENS DE HARDWARE PARA O END-DEVICE	64
4.1.1	Ensaio de Integração do DHT22	64
4.1.2	Ensaio de Integração do PMS5003	67
4.1.3	Ensaio Unificado dos Sensores	72
4.1.4	Prototipação do <i>End-Device LoRaWAN com o Inlp/LMIC-node</i>	75
4.1.4.1	Definição de Dispositivo e Modo de Operação	75
4.1.4.2	Definições Relacionadas ao Suporte LMIC	78
4.1.4.3	Utilização do Dispositivo e Instalação das Dependências	79
4.1.4.4	Intervenção de Código para Leitura dos Sensores	80
4.2	CONFIGURAÇÃO DA REDE NEUTRA	83
4.2.1	Configuração do <i>Gateway Dragino LPS8</i>	83
4.2.2	Configuração do Servidor de Rede TTN	86
4.2.2.1	Criação de Conta e Localização de Serviço do TTN	86
4.2.2.2	Registro do <i>Gateway</i> no TTN	87
4.2.2.3	Registro de uma Aplicação e <i>End Devices</i> Associados	88
4.2.2.4	Configuração <i>Payload Formatters</i>	90
4.2.2.5	Teste de Tráfego de Mensagens de <i>Uplink</i>	92
4.3	APLICAÇÃO DE MONITORAMENTO	94
4.3.1	Ingestão de Dados e Persistência	94
4.3.2	Apresentação dos Dados em <i>Dashboard</i>	96
5	CONSIDERAÇÕES FINAIS	99
5.1	CONCLUSÃO	99
5.2	TRABALHOS FUTUROS	100
REFERÊNCIAS		101

Apêndices	107
APÊNDICE A Arquivo <i>platformio.ini</i> para o ensaio com o DHT22	108
APÊNDICE B Código fonte para ensaio com o DHT22	109
APÊNDICE C Arquivo <i>platformio.ini</i> para o ensaio com o PMS5003	111
APÊNDICE D Código fonte para ensaio com o PMS5003	112
APÊNDICE E Arquivo <i>platformio.ini</i> para o ensaio unificado	114
APÊNDICE F Código fonte para ensaio unificado	115
APÊNDICE G Arquivo <i>platformio.ini</i> na customização do <i>end device</i>	118
APÊNDICE H Customização do <i>end device</i> LoRaWAN - declarações globais	120
APÊNDICE I Customização do <i>end device</i> LoRaWAN - trecho de inicialização	121
APÊNDICE J Customização do <i>end device</i> LoRaWAN - trecho em <i>loop</i>	122
APÊNDICE K <i>Payload Formatter</i> inserido no TTN	125
APÊNDICE L Ingestão da Dados e Persistência	126
APÊNDICE M Apresentação dos Dados em <i>Dashboard</i>	127

1 INTRODUÇÃO

Localizada na divisa entre os estados de Pernambuco, Ceará e Piauí, a região do Araripe é conhecida por ser o principal centro produtor gesseiro do Brasil. Estima-se que esta área seja responsável pela produção de cerca de 97% do gesso consumido no país, sendo as atividades relacionadas ao processo produtivo desta matéria-prima essenciais para o desenvolvimento econômico da região (SINDUSGESSO, 2014). Em contrapartida, os impactos decorrentes dessas atividades tem causado degradação ambiental da localidade em diferentes esferas, sendo a deterioração da qualidade do ar uma delas (GRANJA *et al.*, 2017).

A produção gesseira na região do Araripe causa a emissão aérea de poeira de gesso na forma gases poluentes e materiais particulados (SANTOS; EL-DEIR, 2019). Esses subprodutos, quando suspensos no ar, podem levar a uma série de danos à saúde das pessoas que vivem ou/e trabalham na região. No caso dos materiais particulados, a inalação suficientemente prolongada pode levar a irritação de olhos e mucosas, problemas respiratórios, efeitos crônicos permanentes ou até mesmo distúrbios neuropsíquicos (MEDEIROS; SILVA; HURTADO-GUERRERO, 2010). Além disso, sedimentação desses resíduos também causa impactos ambientais relacionados à deterioração da qualidade do solo e comprometimento de reservas hídricas (SANTOS; EL-DEIR, 2019).

Partículas sólidas em suspensão com idealizações de diâmetros de 10 e 2,5 micrômetros (MP_{10} e $MP_{2,5}$, respectivamente) são destacadas por entidades nacionais e internacionais como indicadores importantes sobre a presença de materiais particulados no ar atmosférico (MMA, 2019; WHO, 2021). Essa presença afeta a qualidade do ar, sendo que este impacto é expresso através da consideração dessas variáveis em planos interinos de redução dos níveis de poluentes recomendados pela OMS (*Organização Mundial da saúde*) e no índices de qualidade do ar utilizados como referência (MMA, 2019).

No Brasil, o CONAMA (Conselho Nacional do Meio Ambiente), além de projetar um índice de qualidade do ar em escopo nacional e projetar tais planos interinos no país, enfatiza a importância da utilização de redes de monitoramento de qualidade do ar para a verificação o grau de exposição da população aos poluentes atmosféricos, considerando critérios de saúde pública (MMA, 2019). Além disso, o órgão consultivo destaca que, nesse contexto, sistemas de monitoramento são importantes para o acompanhamento de tendências de médio e longo prazo para verificar a eficácia dos programas de controle de qualidade do ar e avaliar necessidade de aprimoramentos (MMA, 2019).

Nesse contexto, tecnologias que vem sendo empregadas para o desenvolvimento de sistemas de Internet das Coisas (*Internet of Things - IoT*) poderiam compor uma proposta de solução conveniente para a realização de tarefas de monitoramento escalável e flexível sobre a presença de materiais particulados em zonas de interesse. Para o referido cenário, equipamentos de rádio classificados como LPWAN (*Low-Power Wide-Area Network*) se mostram

uma alternativa adequada quando é necessário lidar com uma taxa de transmissão de dados reduzida, longos distanciamentos entre dispositivos sensores e estação base e requisitos limitantes em relação ao consumo energético (MUTEBA; DJOUANI; OLWAL, 2019).

Neste trabalho de conclusão de curso é apresentada uma proposta de arquitetura e desenvolvimento de um sistema IoT para monitoramento da qualidade do ar usando as tecnologias LoRa e LoRaWAN, com foco na presença de materiais particulados MP₁₀ e MP_{2,5}. O projeto visa verificar, em ambiente controlado, a viabilidade do uso de um sistema IoT como plataforma para rede de monitoramento de qualidade do ar em zonas emissoras dessas partículas, como o Polo Gesseiro do Araripe.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho é o desenvolvimento de um sistema IoT para monitoramento da presença de materiais particulados MP₁₀ e MP_{2,5} para áreas impactadas pelo pó de gesso em suspensão, como o Polo Gesseiro do Araripe. Além do monitoramento dessas variáveis o sistema deve ser capaz de armazenar esses dados e disponibilizá-lo para que possam ser consultados e analisados posteriormente.

1.2 OBJETIVOS ESPECÍFICOS

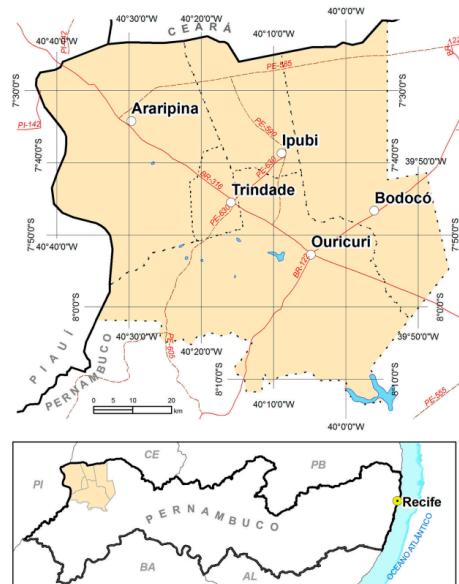
- Desenvolver e testar protótipo de circuito para a captura de dados sobre agentes poluentes MP₁₀ e MP_{2,5}, bem como de condições pontuais de umidade e temperatura;
- Desenvolver e testar o protótipo de um dispositivo periférico e sua integração a uma arquitetura que consiga trafegar dados utilizando o protocolo LoRaWAN;
- Desenvolver aplicação para visualização e análise dos dados coletados pelo sistema de monitoramento de materiais particulados em suspensão;

2 FUNDAMENTAÇÃO TEÓRICA

2.1 POLUIÇÃO DO AR NO POLO GESSEIRO DO ARARIPE

A região da Chapada do Araripe é conhecida por abrigar a maior reserva de minério de gipsita em exploração do Brasil (MEDEIROS; SILVA; HURTADO-GUERRERO, 2010). Trata-se da matéria-prima do gesso, que é encontrado na região em alto grau de pureza, com qualidade que permite a produção de materiais com aplicabilidades diversas que vão desde área da construção civil até a odontologia (OIT, 2021). É nessa região, na divisa entre os estados de Pernambuco, Ceará e Piauí, onde fica localizado o Polo Gesseiro do Araripe, principal centro produtor de gesso em território brasileiro, responsável pela produção de 97% do gesso consumido no país (SINDUSGESSO, 2014).

Figura 1 – Região do Polo Gesseiro do Araripe



Fonte: CPRM (2019)

As atividades decorrentes da exploração da gipsita e da produção de gesso é responsável por grande parte do desenvolvimento econômico da região. Estima-se que Polo Gesseiro do Araripe seja responsável pela produção de 1,6 milhão de toneladas de gipsita bruta por ano. Somente no ano 2017, a quantidade do minério originado na região vendida consumida ou transferida para a industrialização foi de 699 mil toneladas, totalizando 112,6 milhões de reais (OIT, 2021). De acordo com Ministério Público do Trabalho de Pernambuco, funcionam na localidade, 49 mineradoras e, em média, 140 indústrias de calcinação e 600 fabricantes de pré-moldados. A mesma fonte enfatiza a associação da produção de gesso com grande parte da geração de empregos na região, sendo essas atividades responsáveis por 13,8 mil postos de trabalho diretos e cerca de 69 mil indiretos (MPT-PE, 2019).

A despeito da importância econômica da produção do gesso na região do Araripe, os problemas socio-ambientais relacionados com a exploração da gipsita e produção do gesso são muitos, sendo que, em geral, os ganhos econômicos não se refletem em melhorias das condições de vida da população local (OIT, 2021). Alguns desses problemas estão relacionados diretamente com impactos ao meio ambiente, com consequências que atingem não só as condições de moradia e trabalho das populações locais, mas também põem em evidência riscos à própria sustentabilidade ambiental da região nas quais essas atividades estão inseridas.

Nesse contexto, uma das causas desse problema sofre contribuições diretas de emissões de agentes detratores da qualidade do ar, na forma de emissões de gases poluentes e a suspensão de materiais particulados. O objetivo das seções subsequentes é contextualizar este trabalho a respeito dos principais impactos da deterioração da qualidade do ar causada pelos poluentes atmosféricos originados de atividades produtoras de gesso na região.

2.1.1 Cadeia Produtiva do Gesso e Qualidade do Ar

Como supracitado, a cadeia produtiva do gesso na região do Araripe trás consigo consequências que estão relacionadas com a poluição do ar na forma de poluentes gasosos e materiais particulados. Esses efeitos são gerados pela forma como são conduzidas as atividades associadas à extração da gipsita e beneficiamento do gesso na região, sendo os subprodutos de processos nessas atividades responsáveis por parte da deterioração da qualidade do ar no polo gesseiro.

A extração do minério de gipsita na região é feita através de atividades mineradoras que precedem o beneficiamento do gesso como produto. A obtenção do minério na forma *in natura* possui fins diversos, como a produção de insumos para agricultura, indústria cimenteira e construção civil (OIT, 2021). De acordo com Departamento Nacional de Produção Mineral (DNPM), para que a extração da gipsita bruta ocorra, utiliza-se de uma série de procedimentos que normalmente incluem a decapagens mecânicas e manuais (retirada da terra até que se observe a camada mineralizada), perfurações, detonações periódicas por meio da utilização de explosivos e marroamento (quebra dos blocos maiores com o auxílio de marrões) DNPM (2001). Martins (2011) complementa, relatando que a exposição constante a máquinas e equipamentos causadores de ruídos e poluentes, sem as devidas proteções, torna o ambiente das minas desfavorável para a saúde dos trabalhadores, especialmente no tocante a problemas auditivos, cardíacos e respiratórios.

Após a gipsita ser extraída pela mineração, é comum que a mesma seja utilizada como matéria-prima em etapas de beneficiamento do minério, necessárias para a produção de gesso que é utilizado na fabricação de pré-moldados e posterior comercialização e consumo final. Esse beneficiamento ocorre por meio de instalações calcinadoras. Martins (2011) detalha que, nesses estabelecimentos, a gipsita bruta passa por processos de britagem, moagem e peneiramento, onde já acontece a emissão de poeira e materiais particulados. No entanto, é na

etapa de calcinação do material que se apresentam maiores impactos ambientais decorrente da cadeia produtiva do gesso, sejam eles diretos ou indiretos. Nessa etapa, a gipsita moída e peneirada é submetida a altas temperaturas para que desidrate e tenha como produto final o gesso. Como a maioria dos equipamentos de calcinação utilizados na região do Araripe não possuem sistemas eficientes na contenção de gases e partículas derivadas do processo, as reações químicas decorrentes desse processo terminam por emitir esses poluentes no ar atmosférico, formando plumas que podem chegar a quilômetros de distância do ponto onde foram emitidas (MARTINS, 2011).

De forma indireta, o processo de calcinação pode acrescentar dano ainda maior à qualidade do ar e ao meio ambiente no geral. Na região do Araripe, é comum que instalações calcinadoras utilizem como fonte energética a queima de biomassa proveniente da vegetação nativa, o que em si já é um grande problema no que diz respeito dos efeitos do desmatamento intensivo na devastação da caatinga e de outras formações vegetais existentes no alto e entorno da Chapada do Araripe (SILVA *et al.*, 2013). A queima do carvão vegetal oriundo dessa biomassa é também uma fonte de poluentes do ar que contribui para elevar o nível de deterioração da qualidade do ar onde é empregada (ARBEX *et al.*, 2004).

2.1.2 Consequências à Saúde Humana e ao Meio Ambiente

As referidas emissões de poluentes acabam trazendo consequências diretas às populações locais, seja através de efeitos diretos à saúde humana causada por exposição prolongada, ou mesmo pela deterioração do meio ambiente na região. Esses efeitos tem sido objeto de estudo no intuito de embasar a promoção de políticas que melhor lidem com a regulamentação e fiscalização da qualidade do ar e preservação ambiental no polo gesseiro.

No que diz respeito aos efeitos direto à saúde humana, de acordo com Medeiros, Silva e Hurtado-Guerrero (2010), considerando ocorrências registradas em termos de internações e óbitos causadas por doenças no aparelho respiratório, os municípios do Polo Gesseiro do Araripe apresentam incidência significativamente maior do que a média do estado de Pernambuco, sendo que os padrões de internações e óbitos na região gesseira possuem relação direta com atividades relacionadas ao trabalho com gesso. A mesma fonte enfatiza que a poeira decorrente do processo de calcinação leva a uma série de problemas à saúde dos trabalhadores, como irritação nos olhos, nas mucosas e no aparelho respiratório, acarretando, em alguns casos, até mesmo efeitos crônicos permanentes e distúrbios neuropsíquicos (MEDEIROS; SILVA; HURTADO-GUERRERO, 2010).

Além disso, essas emissões acabam por gerar consequências relacionadas com o comprometimento do equilíbrio ambiental na região, de forma geral. A sedimentação dessa poeira proveniente dos processos relacionados à exploração da gipsita e beneficiamento do gesso sobre o solo e vegetação causam problemas ambientais relacionados com o abastecimento hídrico, por exemplo. De acordo com Santos e El-Deir (2019), quando esses sedimentos são solubilizados pelas chuvas, podem ser levados a corpos d'água, alterando as características

organolépticas de reservatórios em áreas adjacentes de onde ocorrem processos da cadeia de produção do gesso. Isso resulta não somente em modificações do sabor da água destinada ao abastecimento público, tornando-o amargo ou adstringente, como adiciona efeitos laxativos. Uma vez que a região já sofre com escassez de água e utiliza esses reservatórios como uma das suas principais reservas hídricas para períodos de estiagem, este pode ser identificado como um problema adicional ocasionado pela emissão dos referidos materiais particulados (SANTOS; EL-DEIR, 2019).

2.2 PRINCÍPIOS SOBRE QUALIDADE DO AR

2.2.1 Poluentes Clássicos

A Organização Mundial da Saúde (OMS) estabelece diretrizes globais para classificações a respeito da qualidade do ar, considerando os riscos à saúde humana causados pela exposição a determinados poluentes. A instituição define alguns poluentes do ar como *poluentes clássicos*, e aconselha a respeito de quantificações recomendadas desses poluentes às quais seres humanos podem ser expostos, seja em logo ou curto prazo (WHO, 2021).

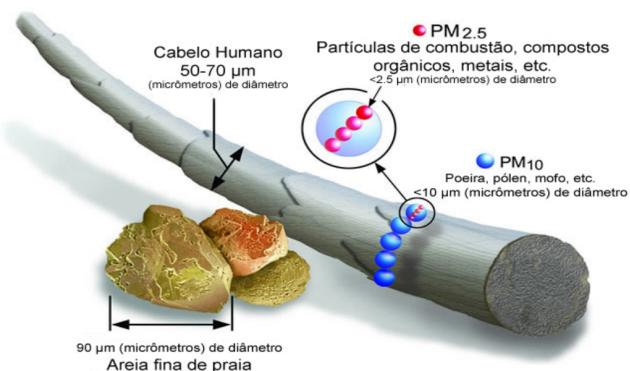
Embora este trabalho tenha como foco o monitoramento de materiais particulados de forma mais específica, a visão geral sobre todos os poluentes clássicos é importante para a contextualização da abordagem proposta. Sendo assim, as características gerais desses poluentes serão detalhadas nas seções subsequentes.

2.2.1.1 Materiais Particulados (MP₁₀ e MP_{2,5})

Materiais particulados são encontrados em ambientes urbanos e não-urbanos, sendo compostos por uma complexa mistura de componentes com características físicas e químicas diversas. Por conta da sua heterogeneidade de características físicas e químicas, os efeitos de sua composição são de difícil estudo. Sendo assim, são genericamente classificadas pelas suas propriedades aerodinâmicas, uma vez que esta determina a forma que acontecem processos de transporte e decomposição dessas partículas para efeitos de depuração pelo aparelho respiratório (EUROPE, 2006).

Nesse contexto, o diâmetro aerodinâmico corresponde a idealização dessas partículas em esferas, sendo essa idealização a base para estudos comparativos. Nas últimas décadas, estudos sobre os efeitos da presença de materiais particulados tem se focado em suas suas propriedade aerodinâmicas, considerando principalmente os diâmetros de 10 µm (MP₁₀) ou 2,5 µm (MP_{2,5}) (EUROPE, 2006).

Figura 2 – Proporção de diâmetros de materiais particulados



Fonte: Adaptado de EPA (2022)

Os principais efeitos à saúde humana da exposição suficientemente prolongada a materiais particulados com esses diâmetros são resumidos a seguir:

- **Diâmetros de 10 μm (MP₁₀):** De acordo com NSW (2020), essas partículas são pequenas o suficiente para passar pela garganta e nariz e entrar nos pulmões. Uma vez inaladas, essas partículas podem afetar o coração e os pulmões e causar sérios efeitos à saúde. A exposição a altas concentrações de MP₁₀ pode resultar em vários impactos na saúde, desde tosse e chiado até ataques de asma e bronquite, pressão alta, ataques cardíacos, derrames e morte prematura (NSW, 2020).
- **Diâmetros de 2,5 μm (MP_{2,5}):** De acordo com NSW (2020), essas partículas pequenas o suficiente para penetrar profundamente nos pulmões e na corrente sanguínea. Health (2018) pontua que a exposição essas partículas pode causar tanto efeitos de saúde a curto prazo, como irritação nos olhos, nariz, garganta e pulmões, tosse, espirros, coriza e falta de ar, quanto afetar a função pulmonar e piorar condições médicas, como asma e doenças cardíacas. A mesma fonte afirma que científicos tem associado o aumento da exposição diária ao MP_{2,5} com o aumento das admissões hospitalares respiratórias e cardiovasculares, visitas ao departamento de emergência e mortes, além de sugerir que a exposição a longo prazo a essas partículas pode estar associada ao aumento das taxas de bronquite crônica, redução da função pulmonar e aumento da mortalidade por câncer de pulmão e doenças cardíacas. Pessoas com problemas respiratórios e cardíacos, crianças e idosos podem ser particularmente sensíveis ao MP_{2,5} (HEALTH, 2018).

2.2.1.2 Poluentes Gasosos (O₃, CO, NO₂ e SO₂)

Embora não seja o foco deste trabalho, por participarem do grupo de poluentes classificados como clássicos pela OMS, as principais características do poluentes gasosos são resumidas a seguir:

- **Ozônio Troposférico (O₃):** Trata-se de um composto formado através de uma série de reações que acontecem por efeitos fotoquímicos. O ozônio na troposfera não é emitido diretamente por fontes primárias, mas pode ser formado através de reações de compostos de nitrogênio com compostos orgânicos voláteis diferentes do metano (*Volatile Organic Compounds - VOC*) ou com o metano (CH₄). Apesar de ser menos reativos do que quando acontecem com os VOC, essas reações encontram muito mais metano disponível na atmosfera. Sendo assim, a fotoquímica envolvendo o metano é responsável por grande parte do aumento do ozônio sobre os oceanos e áreas terrestres remotas de cerca de 30 μg/m³ até cerca de 75 μg/m³ (EUROPE, 2006).

- **Monóxido de Carbono (CO):** O monóxido de carbono é um gás tóxico incolor, não irritante, inodoro e insípido. É produzido pela combustão incompleta de combustíveis carbonáceos, como madeira, gasolina, carvão, gás natural e querosene. O peso molecular do carbono é semelhante ao do ar. Mistura-se livremente com o ar em qualquer proporção e move-se com o ar através do transporte a granel. É combustível, pode servir como fonte de combustível e pode formar misturas explosivas com o ar. Reage vigorosamente com oxigênio, acetileno, cloro, flúor e óxido nitroso. O monóxido de carbono não é detectável pelos humanos através da visão, paladar ou olfato. É apenas ligeiramente solúvel em água, soro sanguíneo e plasma. No corpo humano, reage com a hemoglobina para formar carboxiemoglobina (EUROPE, 2010)
- **Dióxido de Nitrogênio (NO₂):** Sob o ponto de vista da saúde humana, o óxido de nitrogênio que tem maior destaque é o dióxido de nitrogênio. Trata-se de um gás de cor marrom avermelhado de odor característico. O óxido nítrico produz espontaneamente o dióxido de nitrogênio quando exposto ao ar, sendo este um oxidante forte. O NO₂ reage com a água produzindo ácido nítrico e óxido nítrico. Este poluente possui papel direto em mudanças climáticas globais e desempenha um papel crítico na determinação das concentrações de ozônio na troposfera uma vez que a fotólise do dióxido de nitrogênio é chave da formação fotoquímica de ozônio nas camadas mais baixas da atmosfera (EUROPE, 2010).
- **Dióxido de Enxofre (SO₂):** Juntamente com os materiais particulados, o dióxido de enxofre é historicamente associado com a queima de combustíveis fósseis, compondo os principais componentes da poluição do ar em muitas partes do mundo. O dióxido de enxofre na atmosfera é derivado da queima de combustíveis fósseis contendo enxofre, sendo um importante gás poluente em muitas partes do mundo. A oxidação do dióxido de enxofre, especialmente na superfície das partículas na presença de catalisadores metálicos, leva à formação de ácidos sulfurosos e sulfúricos. A neutralização, pela amônia, leva à produção de bisulfatos e sulfatos. O dióxido de enxofre é um gás incolor que é facilmente solúvel em água. Com base em estudos controlados da exposição humana ao dióxido de enxofre na última década foi possível o levantamento de evidências sugestivas sobre os efeitos do dióxido do referido poluente à saúde humana. Esses estudos são relevantes para exposições a picos de concentração que podem surgir de fontes que queimam carvão ou óleo pesado, acompanhadas ou não de concentrações substanciais de materiais particulados (EUROPE, 2010).

2.2.2 Qualidade do Ar no Brasil

No Brasil, as principais definições e resoluções legais que abordam questões relacionadas à qualidade do ar partem de instituições criadas para dar efetividade à Política Nacional do Meio Ambiente (PNMA). A PNMA foi criada em 1981 e recepcionada pela Constituição de 1988, embasando o artigo constitucional 225, que enfatiza o compromisso do estado com a preservação ambiental (BRASIL, 1988). De acordo com esse artigo:

"Art. 225: Todos têm direito ao meio ambiente ecologicamente equilibrado, bem de uso comum do povo e essencial à sadia qualidade de vida, impõe-se ao Poder Público e à coletividade o dever de defendê-lo e preservá-lo para as presentes e futuras gerações."

Neste contexto, a PNMA tem objetivos voltados para a regulamentação de atividades que envolvam possíveis riscos ao meio ambiente, tendo em vista a preservação, melhoria e recuperação da integridade ambiental. Desta forma, a política aborda o favorecimento da vida e da manutenção de condições propícias ao desenvolvimento social e econômico da população, além de garantir o uso não indiscriminado do meio ambiente quando sua utilização colocar em risco o equilíbrio ambiental (BRASIL, 1981).

O Sistema Nacional do Meio Ambiente (SISNAMA) é posto como estrutura institucional adotada para a gestão ambiental, com o objetivo de implementar a PNMA no país. De forma geral, o SISNAMA possui estrutura constituída por órgãos e entidades na União, dos Estados, do Distrito Federal, dos Municípios e elas fundações do Poder Público. Dentro dessa estrutura, o Conselho Nacional do Meio Ambiente (CONAMA) se apresenta como órgão deliberativo e consultivo, sendo este responsável por convenções e normatizações a respeito da relação da população com meio ambiente em âmbito nacional (BRASIL, 1981).

Na sua função, o CONAMA caracteriza as principais referências legais sobre poluição atmosférica no Brasil, tomando como base a quantidade de substâncias poluentes presentes no ar. De acordo com a resolução nº 491 de novembro de 2018, o conselho considera poluição a seguinte definição (CONAMA, 2018):

"(...) qualquer forma de matéria em quantidade, concentração, tempo ou outras características, que tornem ou possam tornar o ar impróprio ou nocivo à saúde, inconveniente ao bem-estar público, danoso aos materiais, à fauna e flora ou prejudicial à segurança, ao uso e gozo da propriedade ou às atividades normais da comunidade."

No que diz respeito aos efeitos da deterioração da qualidade do ar, sob a resolução nº 5, de 15 de junho de 1989, o CONAMA dispõe também sobre o Programa Nacional de Controle de Qualidade do Ar (PRONAR), que aborda a política empregada em ações que promovam o monitoramento da qualidade do ar, determinando como estratégica a criação de uma Rede Nacional de Monitoramento da Qualidade do Ar, dada a necessidade de conhecer e acompanhar os níveis da qualidade do ar no país (CONAMA, 1989).

O conselho é mais preciso com relação às prioridades de detecção de poluentes quando publica versões do *Guia de Monitoramento de Qualidade do Ar*, documento que estabelece diretrizes sobre a disposição de equipamentos de monitoramento da qualidade do ar no Brasil. O guia faz referência direta ao mesmos poluentes clássicos postos como de essencial prioridade pela OMS e projeta a utilização de um índice de qualidade do ar com um sistema de cores que torna facilmente reconhecíveis níveis diferentes de relação com cada poluente clássico (MMA, 2019).

2.2.2.1 Índice de Qualidade do Ar (IQAri)

O referido *Guia de Monitoramento de Qualidade do Ar* do CONAMA toma como referência o índice de qualidade do ar utilizado pela CETESB (Companhia Ambiental do Estado de São Paulo), que por sua vez se inspira na USEPA (*United States Environment Protection Agency*) na construção de seu indicador desde 1981. A agência americana formulou uma metodologia oficial para o seu índice, o AQI (*Air Quality Index*), baseando seus cálculos no conhecimento obtido durante muitos anos de estudos sobre os poluentes e deterioração da qualidade do ar atmosférico nos contextos dos Estados Unidos e no Canadá (LISBOA; KAWANO, 2007).

A ideia original do AQI é o estabelecimento de um indicador classificável de fácil entendimento pela população, com o intuito de possibilitar melhor compreensão geral sobre níveis de exposição aos agentes poluidores relativos aos poluentes clássicos. Dessa forma, haveria maior facilidade de assimilação quando comparado ao contato da população geral com estudos complexos sobre poluição atmosférica (CETESB, 2017). Classificações simplificadas, associadas a cores intuitivas tornam mais simples a identificação dos níveis de qualidade do ar, sem a necessidade de entendimento aprofundado sobre características específicas de concentrações de cada um dos poluentes abordados pelo índice.

O referido indicador é um valor adimensional calculado por uma fórmula que utiliza como parâmetros valores de limites tabelados relativos a escalas de concentrações de poluentes, como será detalhado mais adiante em um caso mais específico. Após calculado, este valor pode ser classificado em cinco diferentes categorias que vão de "boa" (quando a exposição não possui efeitos diretos à saúde humana) a "péssima" (quando a exposição apresenta os riscos mais sérios apresentados pela escala). A escala prioriza os efeitos da poluição atmosférica quando existe exposição humana a aos poluentes clássicos com base em períodos curtos. Assim, o posicionamento dessas classificações é feito em termos de concentrações desses poluentes em termos de horas ou dias. Dessa forma, os efeitos da exposição a cada poluente por determinado período de tempo, levando em consideração os efeitos da magnitude dessas exposições àqueles que respiram aquele ar (USEPA, 2014).

Atualmente, o AQI é utilizado como referência para a implementação de indicadores de qualidade do ar em diversos países ao redor do mundo. Ainda assim, é comum que existam variações entre o índice e indicadores implementados em outras localidades. Essas

variações ocorrem de país para país e decorrem pelo fato de haverem diferenças na forma que cada legislação trata e estabelece limites de tolerância sobre a exposição da população aos poluentes atmosféricos da escala (WAI *et al.*, 2012).

No Brasil a CETESB implementa metodologia baseada no AQI no estado de São Paulo, sendo que este indicador serve de referência para o CONAMA, que o chama de IQAr (Índice de Qualidade do Ar). Assim como no AQI, a formulação do IQAr como indicador considera a medida do índice para cada poluente, sendo estes calculados em resultados adimensionais e proporcionais ao nível de exposição a cada um deles. A esses resultados são atribuídas classificações que fazem referência à qualidade do ar com relação a cada poluente.

Os cálculos para se ter o IQAr são realizados para cada um dos poluentes clássicos, sendo que cada um deles será enquadrado em sua classificação de qualidade do ar. Neste sentido, a CETESB enfatiza que, para efeito de simplificação do resultado para divulgação, uma vez que se possa ter valores do índice para cada um dos poluentes, a convenção é a utilização do índice mais elevado entre eles. Dessa forma, quando é necessário agregação dos valores de cada poluente em um único índice geral, considera-se sempre o maior índice e também em sua classificação, sendo este o pior caso (CETESB, 2017).

O cálculo para o IQAr relativo a qualquer um dos poluentes é feito utilizando a relação estabelecida pela Equação (2.1), cujos parâmetros incluem as faixas de valores apresentados na Tabela 1.

$$IQAr = I_{ini} + \frac{(I_{fim} - I_{ini})}{(C_{fim} - C_{ini})}(C - I_{ini}) \quad (2.1)$$

onde:

I_{ini} : valor do índice que corresponde à concentração inicial da faixa;

I_{fim} : valor do índice que corresponde à concentração final da faixa;

C_{ini} : concentração inicial da faixa em que se localiza a concentração medida;

C_{fim} : concentração final da faixa em que se localiza a concentração medida;

C : concentração medida do poluente.

Tabela 1 – Classificações da qualidade do ar com base no IQAr

Qualidade do Ar	Índice	MP ₁₀ (μg/m ³) 24h	MP _{2,5} (μg/m ³) 24h	O ₃ (μg/m ³) 8h	CO (ppm) 8h	NO ₂ (μg/m ³) 1h	SO ₂ (μg/m ³) 24h
N1 - Boa	0 - 40	0 - 50	0 - 25	0 - 100	0 - 9	0 - 200	0 - 20
N2 - Moderada	41 - 80	>50 - 100	>25 - 50	>100 - 130	>9 - 11	>200 - 240	>20 - 40
N3 - Ruim	81 - 120	>100 - 150	>50 - 75	>130 - 160	>11 - 13	>240 - 320	>40 - 365
N4 - Muito Ruim	121 - 200	>150 - 250	>75 - 125	>160 - 200	>13 - 15	>320 - 1130	>365 - 800
N5 - Péssima	201 - 400	>250	>125	>200	>15	>1130	>800

Fonte: Adaptado de CETESB (2017)

Esses cálculos consideram enquadramentos relativos às concentrações iniciais e finais da coluna índice, bem como da coluna correspondente ao poluente sobre o qual se deseja obter o IQAr, para criar uma relação classificável entre medidas individuais e a escala utilizada. A classificação do IQAr para cada poluente possui significados com relação aos efeitos gerais que cada nível alcançado pode provocar na saúde humana quando há a exposição prolongada.

A relação entre as classificações do indicador com seus efeitos à saúde humana é demonstrada na Tabela 2.

Tabela 2 – Efeitos estimados para cada classificação do IQAr

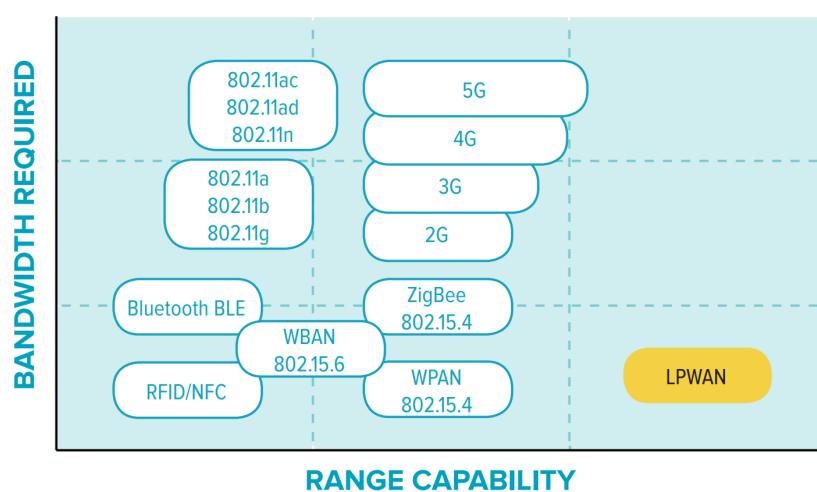
Qualidade do Ar	Índice	Efeitos
N1 - Boa	0 - 40	-
N2 - Moderada	41 - 80	Pessoas de grupos sensíveis (crianças, idosos e pessoas com doenças respiratórias e cardíacas) podem apresentar sintomas como tosse seca e cansaço. A população, em geral, não é afetada.
N3 - Ruim	81 - 120	Toda a população pode apresentar sintomas como tosse seca, cansaço, ardor nos olhos, nariz e garganta. Pessoas de grupos sensíveis (crianças, idosos e pessoas com doenças respiratórias e cardíacas) podem apresentar efeitos mais sérios na saúde.
N4 - Muito Ruim	121 - 200	Toda a população pode apresentar agravamento dos sintomas como tosse seca, cansaço, ardor nos olhos, nariz e garganta e ainda falta de ar e respiração ofegante. Efeitos ainda mais graves à saúde de grupos sensíveis (crianças, idosos e pessoas com doenças respiratórias e cardíacas).
N5 - Péssima	201 - 400	Toda a população pode apresentar sérios riscos de manifestações de doenças respiratórias e cardiovasculares. Aumento de mortes prematuras em pessoas de grupos sensíveis.

Fonte: Adaptado de CETESB (2017)

2.3 PADRÃO LORA

A escolha por uma tecnologia de comunicação sem fio pode variar de acordo com os requisitos de uma aplicação. Existem várias possibilidades, que incluem Bluetooth/BLE, ZigBee, WiFi e 3G/4G/5G. Entretanto, no contexto de desenvolvimento de sistemas IoT, tecnologias de rede classificadas *Low-Power Wide-Area Network* (LPWAN) tem se tornado promissoras para abordar aplicações típicas, utilizando troca de dados com baixa taxa de transferência (podendo variar de 10 bps até alguns kbps), mas que possam ocorrer através de transmissões de longas distâncias (5 a 40 km em campo aberto), com baixo consumo de energia. Juntamente com Sigfox e NB-IoT, o padrão de rádios LoRa se enquadra como tecnologia LPWAN e tem se destacado no cumprimento de requisitos atendidos por suas características (LINKLABS, 2016).

Figura 3 – Categorização de equipamentos LPWAN com relação a largura de banda e alcance de transmissão



Fonte: LINKLABS (2016)

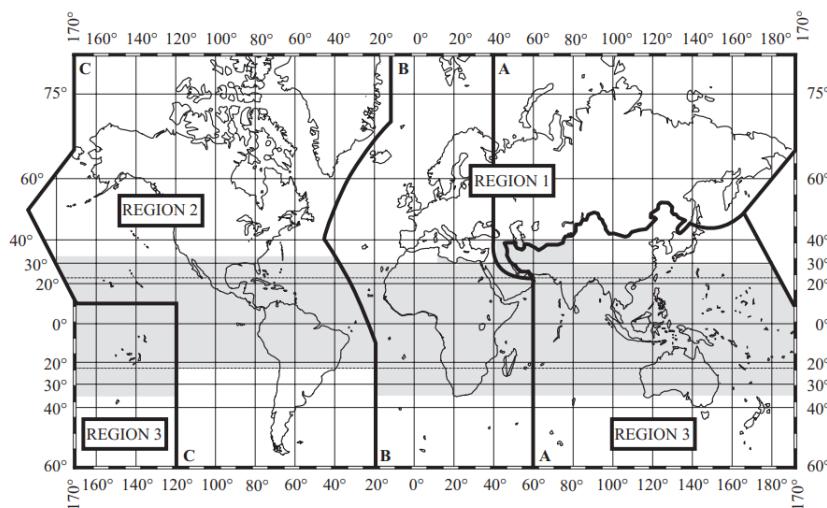
De forma geral, LoRa é um padrão de definições para o funcionamento de rádios que fundamenta a comunicação entre dispositivos LoRa. O objetivo do padrão é definir, a nível de camada física, a forma como dispositivos LoRa conseguiriam interagir e trocar dados em escopo fundamental, ou seja, a nível de trocas de *bits* (SEMTECH, 2019a). De forma geral, isso envolve especificações sobre bandas de operação, largura de banda utilizada, modulação e até mesmo parâmetros a serem ajustados para adequar a camada física implementada a requisitos e restrições de projetos, tendo em vista a melhoria de desempenho ou o cumprimento de normas e conformidades legais na utilização regional do espectro de eletromagnético. Nas subseções a seguir essas especificações serão melhor detalhadas.

2.3.1 Ocupação do Espectro Eletromagnético

Quando se trata de tecnologias que apresentam conectividade entre dispositivos remotos e estações base (como é o caso de equipamentos Bluetooth/BLU, WiFi, ZigBee e

o próprio LoRa) é comum a operação em bandas ISM (Industrial, Scientific and Medical), conhecidas internacionalmente por serem alocações no espectro abertas para fins industriais, científicos e médicos (ITU, 1998). Esta convergência de adequações parte de delineamentos realizados pela ITU (*International Telecommunication Union*), agência especializada da Organização das Nações Unidas responsável por propor padronizações na administração do espectro de radiofrequência a nível global. Vale ressaltar que fica a cargo da legislação vigente em cada país as considerações sobre a conformidade ou não de suas normas com tais propostas de padronização. A ITU divide o globo terrestre três regiões, relativas ao uso padronizado da banda ISM na União Européia (região 1), Estados Unidos (região 2) e Austrália (região 3) (ITU, 1998).

Figura 4 – Divisão regional do globo terrestre pela ITU



Fonte: ITU (1998)

O padrão LoRa define rádios com capacidade de operar em faixas de frequência que oportunamente sobrepoem bandas ISM nas diferentes regiões definidas pela ITU. O LoRa opta por operar em bandas ISM consideradas sub-GHz (menores que um Gigahertz), uma vez que a utilização de frequências menores é conveniente aos propósitos dos equipamentos LPWAN por trazer consigo vantagens quando se compara ao uso de frequências maiores (como 2,4 e 5 GHz, utilizadas por tecnologias associadas ao WiFi). Sendo assim, no que diz respeito ao alcance de operação no espectro eletromagnético, rádios LoRa estariam aptos a serem adaptado às necessidades de regulamentação de países, desde que haja uma conformidade com as bandas ISM sub-GHz (SEMTECH, 2015).

A exemplo disso, tranceptores como o SX1272/73 (Semtech), que são utilizados para a construção de módulos de rádio que suportam o padrão LoRa, possibilitam uma cobertura do espectro de 860 até 1020 MHz (SEMTECH, 2019b). Esta faixa de operação permite que os módulos construídos utilizando esses tranceptores consigam alcançar bandas ISM sub-GHz acima de 860 MHz. Já tranceptores como o SX1276/78 (Semtech) possuem um espectro de atuação possível maior, conseguindo alcançar frequências entre 137 e 1020 MHz (SEMTECH,

2020b). Sendo assim, a configuração dos parâmetros de operação desses módulos os adaptaria à cumprir uma ampla diversidade de normas e conformidades regionais para o uso de dispositivos LPWAN.

No Brasil, a Agência Nacional de Telecomunicações (ANATEL) é a agência reguladora responsável por lidar com conformidades relacionadas com a regulamentação do uso do espectro eletromagnético. A agência define as faixas de frequência relativas à ISM dentro do país abertas e gratuitas para uso, embora ainda haja a necessidade de homologação dos dispositivos que fazem uso dessa banda junto ao referido órgão regulador (ANATEL, 2021), de acordo com a Lei Geral de Telecomunicações (resolução nº 680 de 2017). No entanto, a ANATEL possui classificações próprias a respeito da operação de equipamentos de rádio no território nacional, sendo que as definições de LPWAN e características de operação relativas ao padrão LoRa se enquadrariam na definição de "equipamentos de radiação restrita", de acordo com ANATEL (2020) e ANATEL (2021).

Apesar de a ANATEL adotar designações da ITU relativas à região 2, que coincide com a forma que os Estados Unidos faz uso do seu espectro, existem diferenciações a serem consideradas. Enquanto que no território estadunidense, rádios LoRa operam na faixa entre 902-928 MHz (LORA ALLIANCE, 2020), no Brasil a portaria nº 50637, de 18 de dezembro de 2015, no escopo da resolução nº 454/2006, transfere a faixa entre 907,5 a 915 MHz para serviços móveis pessoais e serviços multimídia. Sendo assim, aos equipamentos de radiação restrita restam as faixas entre 902-907,5 MHz e 915-928 MHz (ANATEL, 2006). Sendo assim, para que sejam homologados pela ANATEL, equipamentos que utilizam rádios LoRa precisam ser adaptados para operar dentro dessas duas faixas não contíguas.

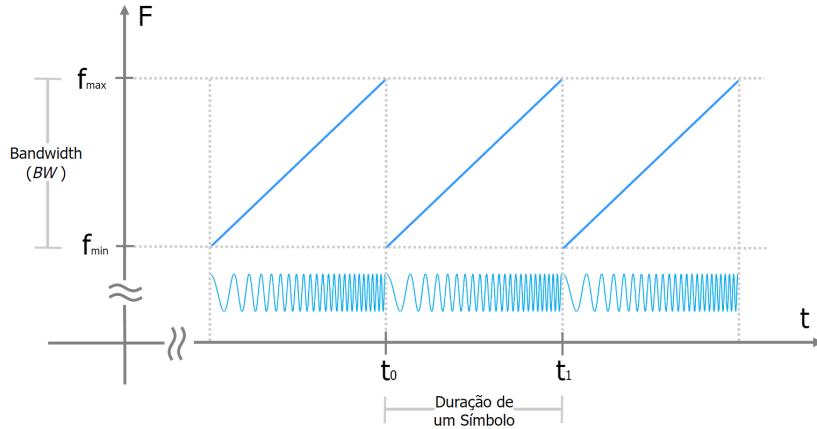
2.3.2 Técnica de Modulação Digital

O padrão LoRa utiliza espectro de operação do rádio para alocação de canais de comunicação dentro dos quais sinais portadores realizam a modulação digital de dados a serem transmitidos. Desta forma, essas transmissões conseguem ser captadas por rádios LoRa que fazem a demodulação do sinal para reconstrução do dado trafegado. Para que isso ocorra, o LoRa especifica um padrão próprio de modulação, buscando enfatizar e melhorar as características de sua proposta: minimização do consumo energético e longo alcance de transmissão (SEMTECH, 2019a).

De acordo com a SEMTECH (2019a), a modulação proposta pelo LoRa se baseia em uma derivação da técnica denominada CSS (*Chirp Spread Spectrum*), que utiliza a extensão da faixa espectral do canal utilizado para gerar ondas portadoras que variam de frequência, aumentando ou diminuindo em função do tempo. No LoRa, essas variações acontecem dentro dos limites das frequências máxima e mínima do referido canal. Assim, apesar do sinal portador variar em frequência ao longo do tempo, essas variações estão sempre acontecendo dentro de limites da largura de banda (*bandwidth - BW*) atribuída ao canal, reiniciando as variações de frequências em períodos de tempo determinados para um mesmo estado de

funcionamento (MAUDET *et al.*, 2021).

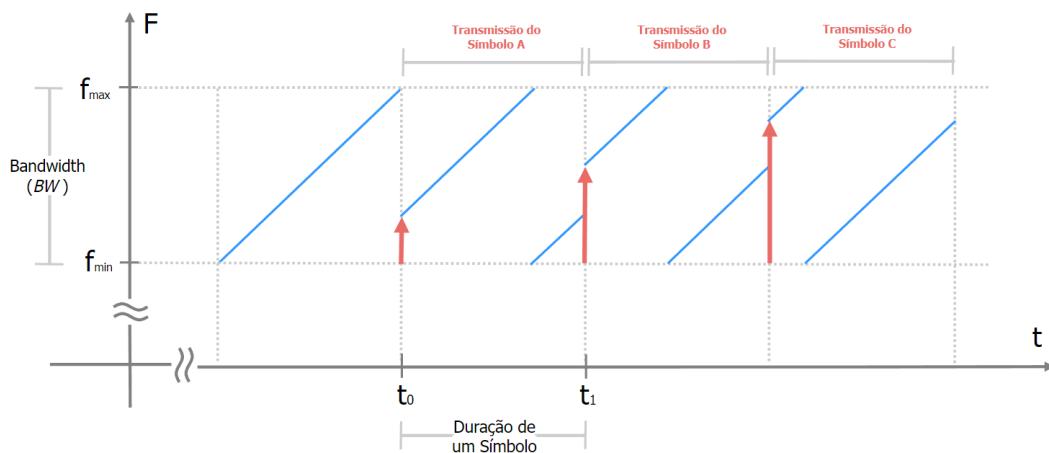
Figura 5 – Variação da frequência no tempo na modulação LoRa/CSS



Fonte: Autoria própria (2023)

Na modulação LoRa, a duração desse período acaba também sendo a referência de tempo para o envio de cada símbolo. A diversificação na forma com a qual acontecem as variações de frequência do sinal portador dentro de um determinado período caracteriza a pluralidade de símbolos potencialmente enviados (MAUDET *et al.*, 2021). São necessários múltiplos *chirps* em composição para formar cada período, sendo cada símbolo transmitido utilizando um período completo (SEMTECH, 2015).

Figura 6 – Transmissão de símbolos na modulação LoRa/CSS



Fonte: Autoria própria (2023)

Durante a variação da frequência dentro da duração de um símbolo, esteja ela acontecendo de forma incremental (*chirp-up*) ou decremental (*chirp-down*), acrescimos e decrescimentos são realizados de forma discreta (SEMTECH, 2015). Sendo assim, existe um número definido de variações de frequência que comporta a largura de banda de um canal. Este

número possível de variações de frequências é determinado pelo fator de espalhamento (*spreading factor* - SF), um parâmetro cuja função será detalhada na próxima seção.

2.3.3 Principais Parâmetros Configuráveis

Rádios LoRa estabelecem sua comunicação de forma parametrizada por valores que podem ser ajustados de acordo com necessidades relacionadas a desempenho dado um determinado caso de uso, adequações a padrões ou mesmo o cumprimento de normas regionais. Alguns desses parâmetros já foram mencionados neste trabalho, uma vez que os mesmos se relacionam com a forma que o padrão LoRa trabalha de forma geral. O reconhecimento desses parâmetros é importante, uma vez que isso embasa o entendimento de como a camada física de dispositivos baseados no LoRa pode variar em termos possibilidades de configuração e funcionamento.

Desta forma, para que seja possível dar prosseguimento à manipulação desses equipamentos em fases práticas deste trabalho, é necessário o entendimento de alguns desses parâmetros, bem como de outros que se apresentarão de forma acumulativa em outras camadas, como é o caso da camada MAC, implementada pelo protocolo LoRaWAN. Mesmo ao fazer uso de camadas de abstração superiores os principais parâmetros que configuram os trancceptores LoRa se mostraram úteis para lidar com a tecnologia. Os principais parâmetros e seus efeitos quando variados serão detalhados nas seções subsequentes.

2.3.3.1 Frequência da Portadora (*Carrier Frequency* - CF)

A frequência do sinal portador em rádios LoRa é a frequência central de operação que abrange uma determinada faixa de acordo com a capacidade do trancceptor utilizado. Como mencionado, a banda de operação do LoRa se apresenta nas faixas ISM sub-GHz (SEMTECH, 2019a). Dentro dos limites possíveis de alcance espectral do rádio, é comum a possibilidade de passos de 61 Hz para o estabelecimento de canais dentro da faixa operação do dispositivo. Dependendo do chips LoRa utilizado, essa faixa pode cobrir limites diferentes, sendo comum os intervalos de operação entre 137 e 1020 MHz e 860 to 1020 MHz (BOR *et al.*, 2016).

2.3.3.2 Largura de Banda (*Bandwidth* - BW)

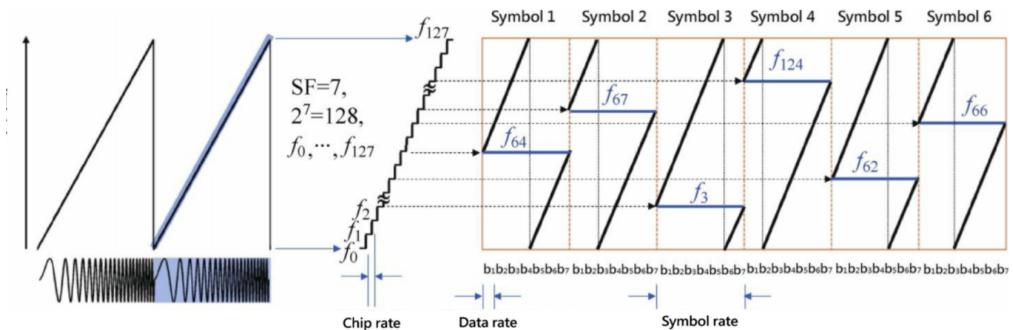
Dentro da banda de operação de um rádio LoRa, é possível a definição da largura de banda dos canais, o que determina a quantidade de canais possíveis dentro de uma mesma faixa de frequências de operação (SEMTECH, 2019a). Como mencionado anteriormente, a largura de banda também determina a faixa de frequências dentro da qual o sinal portador varia. Sendo assim, mantidos os outros parâmetros, a variação da largura se relaciona com o fator de espalhamento na determina (SEMTECH, 2019a). Por outro lado, quanto maior for *BW*, maior tende a ser a taxa de transmissão de dados. No entanto, maiores larguras de banda causa menor sensibilidade da antena de recepção do sinal devido à maior integração com

ruídos adicionais (BOR *et al.*, 2016). Embora seja possível a disponibilidade de seleção de larguras de banda em um intervalo entre 7,8 e 500 kHz, as opções de *BW* de um rádio LoRa típico geralmente se dá nas opções de 125, 250 ou 500 kHz (BOR *et al.*, 2016).

2.3.3.3 Fator de Espalhamento (*Spreading Factor - SF*)

Através do fator de espalhamento *SF*, é possível o ajuste da quantidade de *chirps* necessário para o envio de um símbolo e por consequência é possível ajustar a relação entre sensibilidade e taxa de transmissão de dados. Este parâmetro pode assumir valores que vão de SF7 a SF12. Para uma mesma largura de banda, quanto maior *SF* for, maior a quantidade de *chirps* necessários para se transmitir um símbolo. Sendo assim, a duração do símbolo é definida não somente por *BW*, mas também por *SF* (SEMTECH, 2019a).

Figura 7 – Efeitos da variação do fator de espalhamento (SF)



Fonte: Adaptado de CSDN (2021)

Para efeitos de dimensionamento e entendimento das relações entre esses parâmetros A partir do número de *SF*, pode-se calcular o número de pequenas quantizações que vão se espalhar para completar uma largura de banda do canal utilizado. Este número define o número de símbolos e é referenciado como *N* na Equação (2.2). Pode-se também usar *N* para dividir a largura de banda *BW*, o que resulta na taxa de símbolos transmitidos a cada segundo *R_{simb}*, como mostrado na Equação (2.3).

$$N = 2^{SF} (\text{símbolos}) \quad (2.2)$$

$$R_{\text{simb}} = \frac{BW}{N} = \frac{BW}{2^{SF}} (\text{símbolos/segundo}) \quad (2.3)$$

Muitas vezes pode ser necessário realizar esses dimensionamentos para obtenção de um valor em *bitrate*, ou seja, em bits por segundo. Isso é possível invertendo a Equação (2.3) que resulta no período de cada símbolo em segundos da Equação (2.4), sendo esta utilizada para a relação com *SF* para a obtenção da relação em bits por segundo, na Equação (2.5).

$$T_{símb} = \frac{2^{SF}}{BW} (\text{segundos}) \quad (2.4)$$

$$R_{bit} = \frac{SF}{T_{símb}} = \frac{SF}{\frac{2^{SF}}{BW}} (\text{bits/segundo}) \quad (2.5)$$

Para uma mesma largura de banda do canal, quanto menor for a quantidade de *chirps* necessária para chegar completar um período, mais rapidamente o rádio LoRa é capaz de transmitir símbolos (o que acaba refletindo em um *bitrate* maior) (SEMTECH, 2019a). No entanto, isso também implica na diminuição da sensibilidade do dispositivo receptor, uma vez que, por haver menor resolução espectral na composição, menor será a capacidade de se diferenciar esses símbolos (o que implica em uma resiliência menor a interferências em transmissões de longa distância). O oposto também é válido: caso se aumente a quantidade de *chirps* para compor um símbolo, menor será o *bitrate*, mas a sensibilidade do receptor aumentará, acarretando em uma melhor diferenciação dos símbolos e maior imunidade a interferência eletromagnética (SEMTECH, 2019a).

2.3.3.4 Taxa de Codificação (*Coding Rate - CR*)

Transmissões de rádio podem sofrer interferência eletromagnéticas dados os ambientes aos quais são submetidas. Essas interferências podem causar o corrompimento dos dados transmitidos. O LoRa implementa recursos para a correção de erros causados por essas interferências na forma de *Forward Error Correction* (FER) no padrão (SEMTECH, 2019a). Esta técnica especifica o grau de redundância nos bits paridade através dos quais erros de transmissões podem ser identificados. Assim, é possível a reconstituição do pacotes transmitidos, quando necessário. O valor de *CR* nos rádios LoRa podem assumir os valores 4:5, 4:6, 4:7 e 4:8. Quanto maior o *CR* selecionado, maior é a proteção contra essas interferências. No entanto, o *time on air* acaba sendo menor, o que resulta em menor *bitrate* nas transmissões. Rádios LoRa com diferentes *CR* podem estabelecer comunicação uns com os outros, desde que eles utilizem modo explícito no cabeçalho dos seus pacotes (BOR *et al.*, 2016).

2.3.3.5 Potência de Transmissão (*Transmission Power - TP*)

Assim como em outras formas de transmissão de rádio, o padrão LoRa também prevê os ajustes da potência de transmissão. No padrão, é possível a definição de potência de transmissão que podem ser ajustadas entre -4 dBm até 20 dBm, utilizando 1 dB de passo para variações dentro desses limites. No entanto, devido a limitações de implementação de *hardware*, é comum que esses limites sejam de 2 dBm até 20 dBm (BOR *et al.*, 2016). Maiores *TP* implicam diretamente em uma maior quantidade de energia necessária para envios de dados, bem como em maiores alcances de transmissão. Menores *TP* utilizam menos energia,

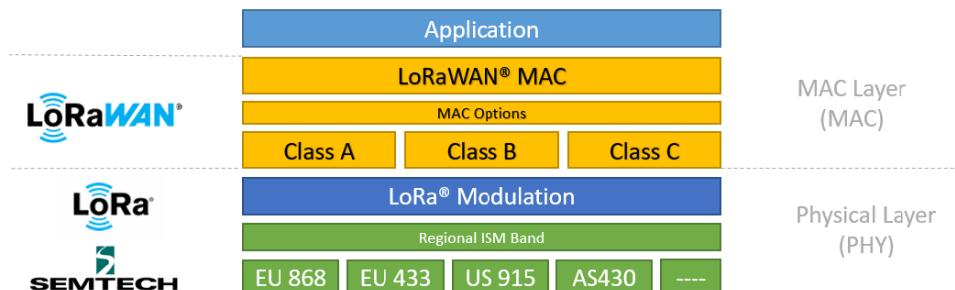
mas possibilitam menores alcances. Assim, dependendo da aplicação e caso de uso, pode-se optar pela priorização de maiores alcances em detrimento da economia energética ou o contrário.

2.4 PROTOCOLO LORAWAN

Enquanto o LoRa é definido como uma forma de padronização da camada física para rádios que utilizam o padrão, o LoRaWAN se apresenta como solução para a implementação de um protocolo de comunicações, que usa rádios LoRa para o estabelecimento de redes de comunicações LPWAN. O protocolo LoRaWAN é aberto e mantido por uma organização denominada LoRa Alliance. A organização é composta por membros da indústria, que possuem em comum o objetivo de participar de definições e aprimoramentos do protocolo (SEMTECH, 2019a).

O LoRaWAN se faz presente no gerenciamento, segurança, confirmação de mensagens, ativação de dispositivos, entre outros aspectos de uma rede administrada pelo protocolo, constituindo assim a camada de acesso ao meio (*MAC - Medium Access Control*) que atua em cima da camada física estabelecida pelo padrão LoRa (SEMTECH, 2019a), como mostrado na pilha de abstrações na Figura 8.

Figura 8 – Organização da pilha LoRa/LoRaWAN



Fonte: SEMTECH (2019a)

Os elementos de uma topologia típica, bem como modos de ativação e classe de dispositivos em uma rede LoRaWAN são melhor detalhados nas seções subsequentes.

2.4.1 Elementos de Rede e Topologia Típica

A utilização do protocolo LoRaWAN implica na interação entre elementos de rede específicos que fazem parte de uma topologia que o protocolo suporta. Existem, neste sentido, papéis definidos para cada um desses elementos, com atribuições essenciais para o funcionamento do protocolo enquanto orquestrador da comunicação entre os dispositivos da rede. Essas atribuições, relativas ao esquema da topologia padrão proposto pelo LoRaWAN, são apresentadas a seguir:

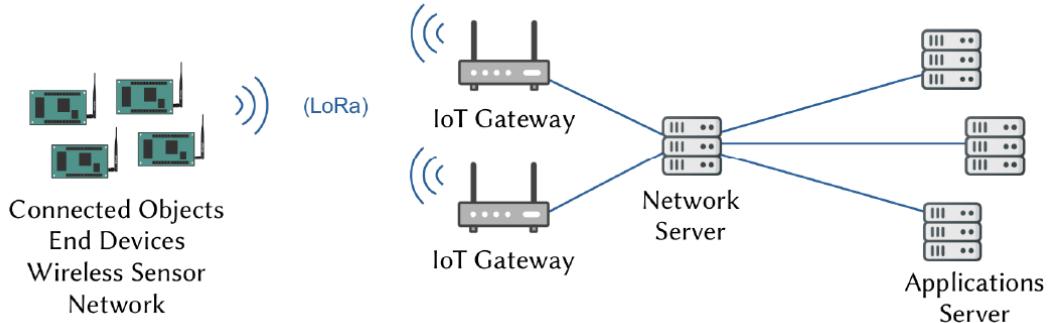
- **Dispositivos Periféricos (*end devices*):** são dispositivos que possuem como atribuição funções relacionadas com a presença digital conectada em ambientes onde a mesma é necessária. Sendo assim, é comum a associação de *end devices* com a integração de sensores para a coleta de dados sobre alguma variável que se deseja manter registro, por exemplo. Em uma rede LoRaWAN os dispositivos periféricos estão sempre associados com uma aplicação específica, utilizando a rede como um elemento neutro intermediário na troca de dados com essa aplicação (SEMTECH, 2019a).
- **Dispositivos Concentradores (*gateway devices*):** são dispositivos cuja principal atribuição é receber os dados enviados pelos dispositivos periféricos em nome de um servidor de rede. Vale ressaltar que no LoRaWAN não existe vinculação determinística entre *end devices* a determinado *gateway device*, uma vez que o sistema de endereçamento lógico das versões do LoRaWAN são feitos através designações nos escopos de identificações redes e aplicações (SEMTECH, 2019a).
- **Servidores de Rede (*network servers*):** são servidores responsáveis pela recepção das mensagens que *gateway devices* associados recebem e pelo encaminhamento das mesmas até a aplicação final. Podem também desempenhar papel inverso, de encaminhar mensagens originadas em servidores de aplicação até os nós associados com as mesmas. Assim, o trabalho de comutação da troca de mensagens entre *end devices* e suas aplicações é feita através dos servidores de rede (SEMTECH, 2019a).
- **Servidores de Aplicação (*application servers*):** Qualquer dado capturado por sensores ou enviado para mudança de estado ou atuação de *end devices* possui um contexto prático. Na arquitetura proposta pelo LoRaWAN, esse contexto serve como escopo de finalidade de comunicação com *end devices* associados (SEMTECH, 2019a).

Fazendo uso dos elementos de rede supracitados, o protocolo LoRaWAN segue um método de acesso ao meio denominado ALOHA. Neste método, os dispositivos periféricos transmitem dados sempre que os mesmos se tornam disponíveis para o envio, fazendo uso alternado de forma pseudo-aleatória entre os canais de frequência disponíveis (LORA ALLIANCE, 2015).

No que diz respeito forma que acontece as comunicações via rádio LoRa em uma rede LoRaWAN, as interações acontecem utilizando uma topologia que forma uma estrela de estrelas, de modo que cada dispositivo periférico pode se conectar a qualquer dispositivo concentrador que consiga alcançar, desde que estejam configurados para atuarem em uma mesma rede lógica (LORA ALLIANCE, 2018).

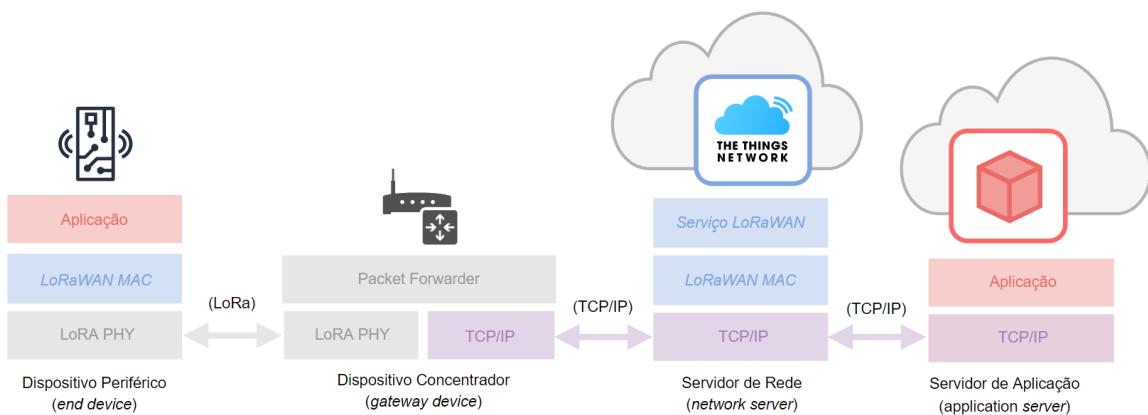
Uma topologia típica com os elementos de rede mencionados é demonstrado na Figura 9, sendo um diagrama de pilhas de abstrações equivalente apresentado na Figura 10.

Figura 9 – Arquitetura típica na utilização protocolo LoRaWAN



Fonte: Maudet *et al.* (2021)

Figura 10 – Interação entre as camadas em uma arquitetura LoRAWAN



Fonte: Autoria própria (2023)

2.4.2 Modos de Ativação

No protocolo LoRaWAN, o servidor de rede se encarrega de encaminhar os dados transmitidos para serem processados em servidores de aplicação aos quais foram designados. Para que isso ocorra, é necessário que haja uma sessão reconhecendo o dispositivo periférico como autorizado a trocar dados, tanto no escopo do servidor de rede quanto do servidor de aplicação. O protocolo faz isso através de chaves de sessão chamadas de *NwkSKey* (*Network Session Key*) e *AppSKey* (*Application Session Key*). Sendo assim, dispositivos periféricos só conseguem enviar dados para servidores de aplicação se possuírem autorização através da posse validada dessas chaves (LORA ALLIANCE, 2021).

A forma como essas chaves são obtidas pelos dispositivos periféricos varia conforme o modo de ativação atribuído aos mesmos na ocorrência ativação (LORA ALLIANCE, 2021). Assim, cada dispositivo periférico possui sua forma de operar dependendo do modo de ativação que foi atribuído ao mesmo. Vale ressaltar que a resposta do servidor de rede a esses dois modos sofrem algumas alterações conforme versão do protocolo LoRaWAN utilizada. No entanto, de forma geral, apresentam comportamentos similares.

Nas seções subsequentes, serão detalhados os dois modos de ativação possíveis para dispositivos periféricos no contexto do protocolo LoRaWAN, denominados de ABP (*Activation By Personalization*) e OTAA (*Over The Air Activation*). Os conceitos apresentados aqui se baseiam nas interações com o LoRaWAN 1.0.x.

2.4.2.1 OTAA (*Over The Air Activation*)

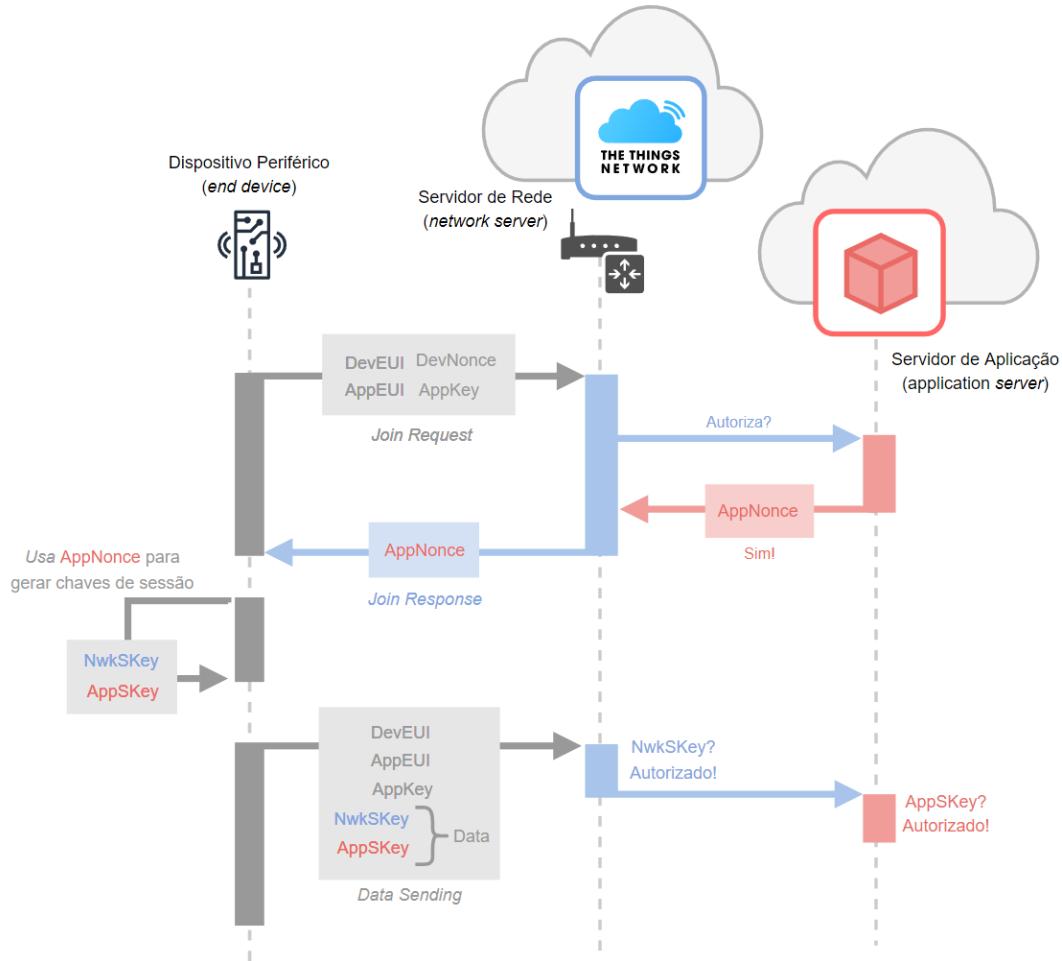
Quando dispositivos periféricos estão configurados para operar em modo de ativação OTAA, é dado aos mesmos uma relação direta com um servidor de aplicação específico, sendo o servidor de rede de importância neutra e de alto nível de abstração no tráfego de dados até a aplicação. A ideia é que, desde que o dispositivo possua credenciais válidas para aplicação a qual está associado, ele poderá obter sua autenticação por meio de um processo de requisição de abertura de sessão, referenciado como *Join Request* (LORA ALLIANCE, 2021). Nesse procedimento, o dispositivo precisa se identificar e apresentar as credenciais para a autenticação na aplicação, usando como intermediário um servidor de rede com alto grau de desacoplamento nesse processo (LORA ALLIANCE, 2021).

A identificação do dispositivo é feita através de um código denominado *DevEUI*, um valor único atribuído ao dispositivo pelo fabricante ou desenvolvedor. Já para realizar a autenticação no servidor de aplicação, o dispositivo periférico precisa apresentar a identificação da aplicação e a chave de permissão de acesso, denominados de *AppEUI* e *AppKey*, respectivamente (LORA ALLIANCE, 2021). O protocolo LoRaWAN demanda também o envio de um código gerado de forma randômica pelo dispositivo, chamado de *DevNonce*, que terá sua utilidade logo após o servidor de aplicação validar as credenciais apresentadas e será explicado mais a frente no texto.

Esses valores são enviados pelo *end device*, no modo OTAA, inicialmente para o *gateway* mais próximo. Após o *Join Request*, o servidor de rede ao qual o gateway está associado encaminha a requisição para o servidor de aplicação associado ao *AppEUI* recebido, onde é verificada também validade de *AppKey*. Caso a validação seja confirmada, o servidor de aplicação utiliza o *DevNonce* recebido pra gerar um código chamado *AppNonce*, a partir do qual o end device requisitor poderia gerar chaves da sessão. Esse código é então enviado de volta, passando pelo servidor de rede até o dispositivo na forma da resposta à sua requisição inicial (*Join Response*). Nessa resposta, o dispositivo recebe *AppNonce*, usando-o para gerar e armazena internamente as chaves *NwkSKey* e *AppSKey* (LORA ALLIANCE, 2021).

Através dessas chaves, agora o dispositivo periférico consegue ter acesso direto a uma sessão de troca de dados com ao servidor de aplicação a qual está associado. O processo necessário para a obtenção dessas chaves no modo OTAA é ilustrado no diagrama de sequência apresentado na Figura 11.

Figura 11 – Obtenção das chaves de sessão no modo OTAA



Fonte: Autoria própria (2023)

Além da obtenção criptografada da chaves de sessão, o dispositivo periférico também recebe outras referências de operação importantes, como *RxDelay*, *DLSettings* e *CList*. Essas referências são vistas sob a óptica do funcionamento geral das interações entre *end devices* e aplicações, independendo do modo de ativação utilizado.

2.4.2.2 ABP (*Activation By Personalisation*)

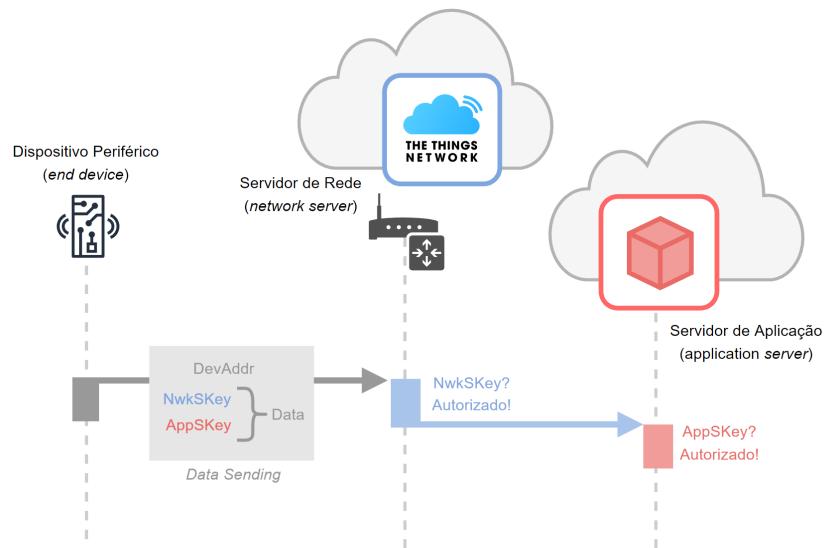
Diferentemente do modo OTAA, onde as chaves de sessão precisam ser obtidas sempre que são expiradas ou perdidas, o modo ABP oferece uma opção de utilização onde o dispositivo periférico é pré-configurado como portador de *NwkSKey* e *AppSKey*. Por possuir de forma prévia essas chaves, existe no modo ABP um acoplamento considerável entre o *end device* não só com a aplicação mas também com a própria rede LoRaWAN utilizada. Sendo assim, não existe a necessidade de uso de *DevEUI*, *DevNonce* *AppEUI* ou *AppKey*, uma vez que os mesmos são utilizados no modo OTAA justamente para a obtenção das chaves de sessão (SEMTECH, 2019a).

No modo ABP, a comunicação não necessita de um processo de *Join Request*, já que o dispositivo periférico já porta, de antemão, as chaves de sessão de forma fixa. A forma de reconhecimento de um *end device* como um agente autenticado é também feita mediante à pré-configuração do dispositivo na rede, sendo isso feito através de um código único para cada dispositivo chamado de *DevAddr*. Então, para se comunicar com a aplicação, basta o *end device* enviar pacotes que contenham a identificação do mesmo (*DevAddr*), assim como as chaves de sessão *NwkSKey* e *AppSKey* (SEMTECH, 2019a).

Apartir do envio, passando pelo *gateway* de acesso, a mensagem chegaria ao servidor de rede e, caso *NwkSKey* for reconhecida como uma chave de sessão de rede válida, o mesmo encaminharia a encaminharia para o sevidor de aplicação. A aplicação então consideraria *DevAddr* e *AppSKey* para servidor de aplicação, onde também seria feita a verificação antes de levar em consideração o conteúdo do pacote LoRaWAN para o processamento.

O processo necessário para a obtenção dessas chaves no modo ABP é ilustrado no diagrama de sequência apresentado na Figura 12.

Figura 12 – Obtenção das chaves de sessão no modo ABP



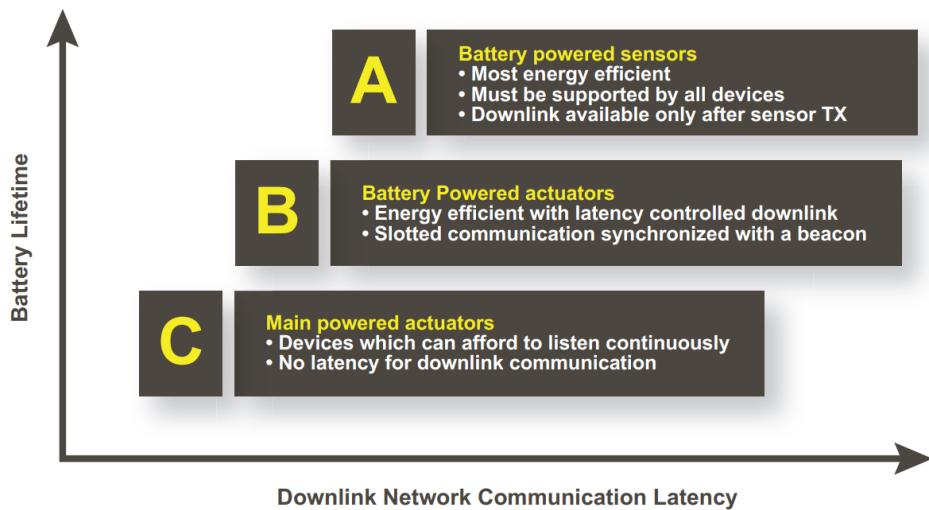
Fonte: Autoria própria (2023)

2.4.3 Classes de Dispositivos

O LoRaWAN trabalha considerando atribuições de classes que fazem referência ao funcionamento básico dos dispositivos periféricos que fazem uso do protocolo. Essas classes são nomeadas de *A*, *B* e *C*, sendo que cada uma delas possui um comportamento específico no que diz respeito à forma que as interações de trocas de dados ocorrem entre *end devices* e *gateway devices* (LORA ALLIANCE, 2015). Essas variações podem ser utilizadas para atender a

casos de uso distintos, considerando principalmente poderações relacionadas ao consumo energético e latência dos canais de resposta (*downlink*) após as transmissões.

Figura 13 – Classes de dispositivos LoRaWAN



Fonte: LORA ALLIANCE (2015)

Em geral, dispositivo que utilizam o protocolo LoRaWAN possuem implementações que obrigatoriamente englobam o comportamento geral dos dispositivos da classe A, sendo as formas de operar das classes B e C derivações da primeira, dadas como de opcional disponibilidade (SEMTECH, 2019a). As especificidades de cada uma das classes são melhor detalhadas nas seções subsequentes.

2.4.3.1 Classe A (*All*)

Dispositivos desta classe possuem a maior eficiência energética entre três classes de dispositivos LoRaWAN. Isso ocorre porque seu funcionamento se baseia em eventos transmissão de dados pelos dispositivos remotos cuja resposta do *gateway device* é esperada somente por dois breves períodos de tempo determinados (SEMTECH, 2019a). Caso não haja resposta em nenhuma das duas janelas de espera, o dispositivo transmissor fecha o canal para de esperar. A limitação de tempos em que dispositivo remoto está disponível para a obtenção da resposta é um fator importante na redução do consumo energético para esta classe. No entanto, isso trás consigo a consequência de limitações na disponibilidade nos canais de *downlink*, uma vez que os mesmos só estão sendo utilizados durante as janelas de espera dos dispositivos transmissores (LORA ALLIANCE, 2015). A Figura 14 ilustra como ocorrem as trocas de dados entre dispositivos LoRaWAN classe A.

Figura 14 – Dispositivos LoRaWAN classe A

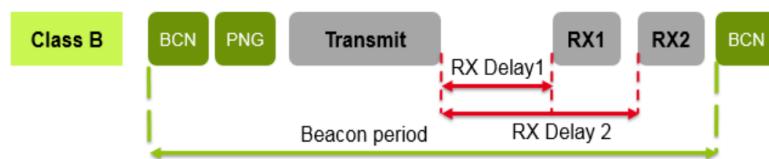


Fonte: Bouguera *et al.* (2018)

2.4.3.2 Classe B (*Beacon*)

Dispositivos desta classe se comportam de maneira similar aos dispositivos configurados na classe A, no entanto, os mesmos recebem sinais de sincronização dos *gateway devices* de tempos em tempos. O objetivo dessa sinalização é indicar para o dispositivo periférico qual o melhor momento para se abrir a janela de recepção da mensagem. (LORA ALLIANCE, 2015). A Figura 15 ilustra como ocorrem as trocas de dados entre dispositivos LoRaWAN classe B.

Figura 15 – Dispositivos LoRaWAN classe B



Fonte: Bouguera *et al.* (2018)

2.4.3.3 Classe C (*Continuous*)

Dispositivos desta classe mantém seus canais de downlink permanentemente abertos, esperando por respostas a eventuais transmissões. Por este motivo, esta classe é a que consome mais energia entre todas as classes. (LORA ALLIANCE, 2015). A Figura 16 ilustra como ocorrem as trocas de dados entre dispositivos LoRaWAN classe C.

Figura 16 – Dispositivos LoRaWAN classe C



Fonte: Bouguera *et al.* (2018)

3 MATERIAIS E MÉTODOS

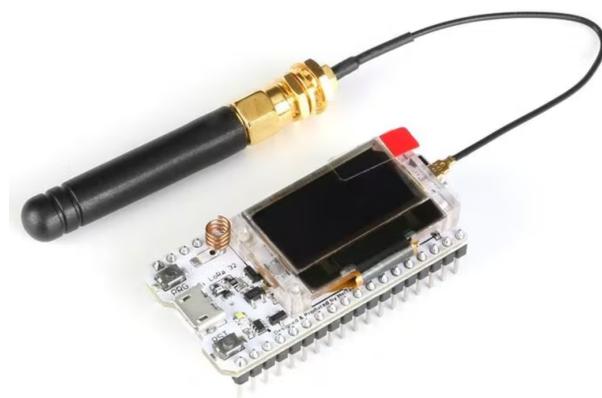
3.1 HARDWARE

3.1.1 Placa de desenvolvimento Heltec WiFi LoRa 32 v2

O desenvolvimento dos dispositivos periféricos de uma rede LoRaWAN para fins de prototipação pode ser realizado tomando como base uma pluralidade bastante ampla de componentes encontrados no mercado. Neste trabalho, priorizou-se a implementação desses dispositivos, levando em consideração critérios que seriam interessantes para o contexto do projeto, tanto no sentido de se ter uma diversidade satisfatória nas possibilidades de customização e integração de diversos tipos de sensores, quanto nas potencialidades de interações com rádios LoRa.

Considerando um amplo leque de possibilidades técnicas, encontrou-se certa conveniência no afunilamento da busca no sentido de encontrar uma placa de desenvolvimento que entregasse esses recursos de forma modularizada, comumente utilizadas e testadas pela comunidade de desenvolvimento IoT. Desta forma, poderia se ter certa garantia de suporte robusto em termos de documentação e de informações das muitas pessoas e entidades que já utilizariam este tipo de dispositivo. Nesse contexto, a placa de desenvolvimento Heltec WiFi LoRa 32 v2 se apresentou como uma alternativa interessante por características que possibilitariam a obtenção das vantagens supracitadas. Uma imagem do dispositivo é mostrada na Figura 17.

Figura 17 – Placa Heltec WiFi LoRa 32



Fonte: Pluijm (2018)

O Heltec WiFi LoRa 32 v2 é composto por pelo menos dois elementos cuja combinação é de grande conveniência para a construção dos protótipos de *hardware* deste trabalho: o microcontrolador ESP32 (produzido pela Espressif) e o transceptor SX1276/SX1278 (da Semtech). Nesta combinação, o ESP32 disponibiliza múltiplos de recursos para integração dos

sensores e outros dispositivos, como pinos GPIO, canais ADC, interfaces I2C, SPI e UART, além de conseguir se comunicar através de WiFi e Bluetooth de forma nativa (ESPRESSIF, 2022). Já o SX1276 ou o SX1278 são transceptores que possibilitam comunicação usando o padrão de transmissão LoRa em nas faixas 868-915 MHz e 433-470 MHz, respectivamente (SEMTECH, 2020b). Essas transmissões acabam sendo recurso indispensável para a implementação do dispositivo periférico, uma vez que por meio do rádio LoRa que acontecem as interações entre o *end device* e o restante da arquitetura LoRaWAN.

A placa de circuito impresso na qual o ESP32 e o SX1276/SX1278 estão fixados possibilita a interconexão física entre os dois dispositivos através do barramento SPI do microcontrolador. Logo, o que basicamente ocorre é que recursos do ESP32 são conectados diretamente às funções do SX1276/SX1278, no intuito de tornar o microcontrolador apto a utilizar a tecnologia LoRa (HELTEC, 2018). Em outras palavras, o código-fonte carregado no ESP32 consegue acessar os recursos e utilizar o rádio LoRa utilizado, de modo que essa interação possuiria suporte de bibliotecas até para lidar com o protocolo LoRaWAN, em níveis mais altos de abstração.

As principais características técnicas relacionadas com alimentação e consumo apresentam alternativas interessantes para o projeto. Isso inclui a possibilidade de alimentação através de bateria, uma vez que possa ser necessário a utilização de *end devices* em condições de independência de fontes de alimentação externas. Além disso, uma preocupação no que diz respeito a placas de desenvolvimento é possível dificuldade de controle de consumo energético não possui definições claras a respeito de consumo de corrente dados seus diferentes estados de operação. No entanto, o Heltec WiFi LoRa 32 v2 apresenta estudos importantes que podem servir de princípios para a condução dos experimentos no que diz respeito ao controle da alimentação e consumo. Essas características são demonstradas de forma resumida na Tabela 3.

Tabela 3 – Principais características técnicas da Heltec ESP32 LoRa

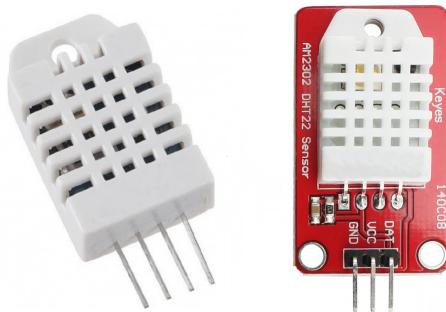
Característica Técnica Relacionadas com Alimentação e Consumo	Valor
Tensão de alimentação através da porta USB (correntes >500 mA)	4,7 até 6,0 V
Tensão de alimentação utilizando bateria	2,7 até 4,2 V
Tensão de alimentação através do pino 3,3V	2,7 até 3,5 V
Tensão de alimentação através do pino 5V	4,7 até 6,0 V
Consumo nominal utilizando WiFi modo Scan	115 mA
Consumo nominal utilizando WiFi modo AP	134 mA
Consumo nominal utilizando LoRa a 10 dB	50 mA
Consumo nominal utilizando LoRa a 12 dB	60 mA
Consumo nominal utilizando LoRa a 15 dB	110 mA
Consumo nominal utilizando LoRa a 20 dB	130 mA
Máxima corrente de saída em pinos de output	500 mA

Fonte: Adaptado de HELTEC (2018)

3.1.2 Sensor de Temperatura e Umidade DHT22

Como solução de sensor para temperatura e umidade, este trabalho propõe a utilização do DHT22 (ou AM2302), desenvolvido pela Aosong Electronics. Este dispositivo permite leituras das variáveis de temperatura e umidade através de sensoriamento resistivo e capacitivo. A faixa de operação no que diz respeito à temperatura cobre o intervalo entre -40 a +80 graus Celsius, com precisão de 0,5 °C e resolução de temperatura de 0,1 °C. Para umidade relativa, são possíveis leituras entre 0 e 100%, com precisões e resoluções de 2% RH e 0,1% RH, respectivamente (AOSONG, 2014). Ambas as faixas de operação, bem como precisões e resoluções são consideradas adequadas para o desenvolvimento de um projeto para qualidade do ar quando se tem como referência variações típicas de temperatura e umidade sazonais na região do Polo Gesseiro do Araripe. Uma imagem do DHT22, bem como de uma placa contendo o sensor para facilitar a construção de protótipos, são mostradas na Figura 18.

Figura 18 – Sensor DHT22 (ou AM2302)



Fonte: Adaptado de BAÚ DA ELETRÔNICA (2021)

Com base nas referências do fabricante, a Tabela 4 resume a funcionalidade de cada um dos quatro pinos encontrados no DHT22.

Tabela 4 – Descrição da função dos pinos do DHT22

Pino	Nome	Descrição
1	VDD	Tensão de alimentação (3,3 a 5,5 volts)
2	SDA	Barramento de dados bidirecional
3	NC	Nenhuma função
4	GND	Tensão de referência GND

Fonte: Adaptado de AOSONG (2014)

Em termos de princípio de funcionamento, o DHT22 é composto por um termistor TNC (*Negative Temperature Coefficient*) combinado com outros elementos resistivos, formando divisão de tensão para gerar saídas correlacionadas com temperatura. Para a umidade, o princípio de funcionamento é capacitivo: há a utilização de eletrodos revestidos de substrato

que fazem a retenção de umidade que causam variações de capacitância que são utilizadas como referência para medição de umidade relativa do ar à qual o sensor está exposto. Os sinais gerados são traduzidos por um circuito integrado que consegue disponibilizar esses dados digitalmente através de um dos quatro contatos entre o sensor e sistemas externos (AOSONG, 2014).

A forma que outros sistemas tem acesso às leituras de umidade e temperatura do DHT22 é feita através de uma única linha de sinal digital, modulando um protocolo de comunicação denominado *Single-Bus Communication Protocol* (AOSONG, 2014). Caso o sistema cliente do sensor não possua um suporte nativo a este tipo de comunicação, existe a possibilidade de se utilizar timers para a captura e decodificação desses sinais. No entanto, o nível de abstração nessa abordagem é baixo, havendo muitos detalhes a serem implementados pelo desenvolvedor, sendo talvez uma melhor recomendação quando não houverem opções de suporte em mais alto nível.

As principais características técnicas sobre o DHT22 relativas aos parâmetros de alimentação e funcionamento geral são destacadas na Tabela 5.

Tabela 5 – Principais características técnicas do DHT22

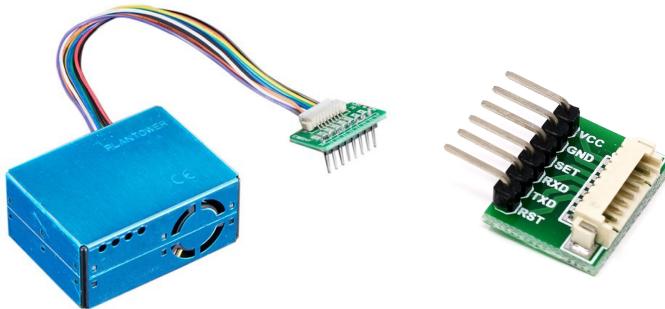
Característica Técnica	Valor
Faixa de tensão de alimentação	3,3 até 5,5 V
Corrente nominal na utilização	0,5 mA
Intervalo mínimo entre amostras	2 segundos
Faixa de temperatura	-40 °C a 80 °C
Precisão de temperatura	± 0,5 °C
Resolução de temperatura	0,1 °C
Faixa de umidade relativa	0 até 100% RH
Precisão de umidade relativa	± 2% RH
Resolução de umidade relativa	0,1% RH

Fonte: Adaptado de AOSONG (2014)

3.1.3 Sensor de Materiais Particulados PMS5003

Dada a necessidade deste projeto de quantificar a presença de materiais particulados MP₁₀ e MP_{2,5}, o sensor PMS5003, desenvolvido pela Plantower, se apresenta como uma alternativa que cumpre os requisitos propostos pelo projeto. Este sensor consegue classificar concentrações de partículas de faixas de diâmetros específicos por cada 0,1 litro de volume de ar. Essa classificação resulta na em quantificações de que podem ser apresentadas em termos de MP₁₀, MP_{2,5} e MP_{1,0} (PLANTOWER, 2016), o que é importante para cumprir requisitos mínimos de sistemas de monitoramento de qualidade no ar no que diz respeito à preocupações com exposição a poluição atmosférica causada por materiais particulados.

Figura 19 – Sensor de materiais particulados PMS5003



Fonte: Adaptado de ROBIUL ELECTRONICS (2022)

Com base nas referências do fabricante, a Tabela 6 resume a funcionalidade de cada um dos pinos encontrados no PMS5003.

Tabela 6 – Descrição da função dos pinos do PMS5003

Pino	Nome	Descrição
1	VCC	Tensão de alimentação (5,0 V)
2	GND	Tensão de referência GND
3	SET	Habilitação do sensor (3,3 V para ativo; 0 V para modo sleep)
4	RX	Recebimento serial TTL (3,3 V)
5	TX	Transmissão serial TTL (3,3 V)
6	RST	Reset do sensor (3,3 V para funcionamento normal; 0 V para reset)
7	NC	Nenhuma função
8	NC	Nenhuma função

Fonte: Adaptado de PLANTOWER (2016)

O princípio de funcionamento do PMS5003 possui base na utilização do espalhamento da luz produzida por um laser sobre partículas suspensas no ar. O grau do espalhamento

refletido é utilizado para a obtenção de uma curva que considera as alterações luminosas em função do tempo. Dessa forma, o diâmetro de partícula equivalente e o número de partículas com diferentes diâmetros por unidade de volume podem ser calculados pelo próprio sensor utilizando a teoria Lorenz-Mie, sendo esta uma solução analítica para as equações de Maxwell tratando da dispersão de radiação eletromagnética utilizando idealizações esféricas representando as partículas (PLANTOWER, 2016).

O PMS5003 se comunica com dispositivos externos através de comunicação UART no padrão TTL de 3,3 volts nas linhas de transmissão (*TX*) e recebimento (*RX*). Por padrão, essa interface é configurada para funcionar utilizando *boudrate* de 9600 bps, com 1 bit de parada. Os dados são transmitidos através de um datagrama de 32 bits contendo as leituras feitas pelo sensor, sendo seu formato demonstrado no *datasheet* disponibilizado pelo fabricante. A forma que o sensor entrega essas leituras para outros dispositivos ocorre com base em dois modos de funcionamento distintos, caracterizados como modo ativo e passivo (PLANTOWER, 2016).

No modo ativo, o sensor envia a cada determinado intervalo de tempo um datagrama contendo os dados de leitura do sensor. Neste modo, o dispositivo alterna automaticamente entre dois submodos. Quando a lógica embarcada do sensor considera que está havendo apenas pequenas variações de concentração entre suas leituras, é mantido o submodo *estável*, quando o intervalo entre as transmissões dos datagramas se dá a cada 2,3 segundos. Quando as variações de concentração são consideradas grandes o sensor alterna para o submodo *rápido*, quando o intervalo entre as transmissões passa a ser entre 200 e 800 milissegundos. Já no modo passivo, o PMS5003 não faz leituras automáticas. Neste modo, o sensor espera por requisições através da interface serial com comandos específicos para realizar suas leituras (PLANTOWER, 2016).

Independente do modo de funcionamento, seja ele passivo ou ativo ou passivo, os dados apresentados pelo sensor apresenta magnitudes de detecção de partículas considerando o diâmetro das partículas em exposição, bem como o número de partículas com diferentes diâmetros por unidade de volume. As classificações de diâmetro das partículas leva em consideração faixas de medição relativas à tabela 7.

Tabela 7 – Classificações de diâmetro de MP feitas pelo PMS5003

Classificação	Faixa de Medição
MP _{1,0}	0,3 até 1,0 μm
MP _{2,5}	1,0 até 2,5 μm
MP ₁₀	2,5 até 10,0 μm

Fonte: Adaptado de PLANTOWER (2016)

Os padrões de alimentação e consumo do PMS5003 são demonstradas na Tabela 8.

Tabela 8 – Principais características técnicas do PMS5003

Característica Técnica	Valor
Faixa de tensão de alimentação	4,5 até 5,5 V
Corrente em modo ativo	$\leq 100 \text{ mA}$
Corrente em modo standby	$\leq 200 \mu\text{A}$
Tempo de resposta única	<1 segundo
Tempo de resposta total	$\leq 10 \text{ segundos}$
Faixa de temperatura de operação	-10 até +60 °C
Eficiência de contagem para MP de até 0,3 μm	50%
Eficiência de contagem para MP $\geq 0,5 \mu\text{m}$	98%
Faixa efetiva para o padrão MP _{2,5}	0 a 500 $\mu\text{g}/\text{m}^3$
Faixa máxima para o padrão MP _{2,5}	$\leq 1000 \mu\text{g}/\text{m}^3$
Erro máximo de consistência para o padrão MP _{2,5}	$\pm 10\%$ para 100 até 500 $\mu\text{g}/\text{m}^3$

Fonte: Adaptado de PLANTOWER (2016)

3.1.4 Gateway LoRaWAN Dragino LPS8

A arquitetura de uma rede LoRaWAN prevê a utilização de *gateways* que suportem o padrão na sua composição. Para assumir o papel deste elemento na topologia, este trabalho priorizou pela busca de um equipamento de relativo baixo custo, amplamente utilizados pela comunidade em contextos de prototipação IoT com o protocolo LoRaWAN, e que fosse compatível com servidor de rede que seria selecionado. Nesse sentido, poucos equipamentos no mercado pareceram mais adequados do que o Dragino LPS8 para a realidade deste projeto.

O Dragino LPS8 (desenvolvido pela empresa chinesa Dragino Technology Co) é um dispositivo que desempenha o papel de ponte entre interfaces de rádios LoRa e redes baseadas no protocolo TCP/IP (DRAGINO, 2022). Através desta ponte, este equipamento consegue realizar a função de um *gateway* LoRaWAN, podendo ser integrado com plataformas de gerenciamento IoT, como é o caso do *The Things Network* (TTN) (DRAGINO, 2022), que será utilizado neste projeto e detalhado em uma outra seção de materiais e métodos mais à frente neste trabalho. Uma ilustração do Dragino LPS8 é mostrada na Figura 20.

Figura 20 – Gateway LoRaWAN Dragino LPS8



Fonte: THANKS BUYER (2022)

O *hardware* do Dragino LPS8 conta com um processador ARM Cortex-A8, tendo à sua disposição 512 MB de memória RAM e 4 GB de armazenamento *flash*. O dispositivo possui interfaces com redes LAN/WLAN através de um adaptador *Ethernet* de 10/100 Mbps e de conectividade WiFi 2.4 GHz, baseada no IEEE 802.11bgn (DRAGINO, 2022). Através dessas interfaces, o LPS8 consegue acessar servidores de rede LoRaWAN remotos. Em termos de

suporte à comunicações utilizando rádios LoRa, o dispositivo suporta até 8 canais LoRa simultâneos, sendo que estes são capazes de emular até 49 demoduladores de forma paralela. Essa tarefa é realizada com o auxílio de um rádio SX1308 atuando em conjunto com duas unidades do SX1257. Esses componentes compartilham o acesso a um conector SMA, destinado ao acoplamento de antenas externas (DRAGINO, 2022).

Através desses rádios LoRa, o Dragino LPS8 consegue dar suportar à funcionalidade de *gateway* LoRaWAN, operando na faixa entre 868 e 915 MHz. Isso faz com que o dispositivo possa atuar em diversos planos de frequência utilizados para ISM, o que inclui EU868, IN865, US915, AU915, AS923 e KR920 (DRAGINO, 2022). Na sua função de *gateway* LoRaWAN, o Dragino LPS8 consegue estabelecer trocas de dados através da utilização da versão 1.0.2 do protocolo (DRAGINO, 2022).

O *software* embarcado no dispositivo se baseia no sistema operacional OpenWrt, que é um sistema de código aberto baseado em Linux para dispositivos embarcados, como roteadores e *gateways* IoT (OPENWRT, 2023). Trata-se de uma base de *software* que foi projetada para fornecer uma alternativa aos sistemas pré-instalados nesse tipo de dispositivo, como *firmware* padrão de fabricantes, que geralmente têm recursos limitados e são de difícil ou impossível personalização (OPENWRT, 2023).

É através desse *software* embarcado no dispositivo que um menu de opções em um navegador de Internet para as configurações do dispositivo é apresentado, sendo que isso ocorre por meio da conexão a um *Access Point* (AP) WiFi gerado pelo dispositivo, ou pelo uso da interface *Ethernet* via cabo RJ-45. Esse acesso será utilizado por este trabalho para a especificação de detalhes de utilização do dispositivo para o contexto deste trabalho de conclusão de curso. Algumas características técnicas gerais do Dragino LPS8 são demonstradas na Tabela 9.

Tabela 9 – Principais características técnicas do Dragino LPS8

Característica Técnica	Valor
Fonte de Alimentação	5V à 2A via cabo USB-C
Processamento	Processador ARM Cortex-A8
Memória	512 MB de RAM e 4 GB de flash
Software Embarcado	OpenWrt (baseado no Linux)
Interfaces LAN/WLAN	WiFi 2.4G (802.11 bgn) e Ethernet 10/100 Mbps
Transceptores LoRa	Um SX1308 e dois SX1257
Faixas de Frequências LoRa	Entre 868 e 915 MHz
Canais Lora	8 canais LoRa programáveis
Demoduladores LoRA	49 demoduladores emulados
Alcance LoRa	Até 10 km em área aberta
Planos LoRaWAN	EU868, US915, AU915, entre outros
Versão LoRaWAN	LoRaWAN 1.0.2

Fonte: Adaptado de PLANTOWER (2016)

3.2 FIRMWARE

3.2.1 Extensão PlatformIO para Microsoft VSCode

A utilização da extensão PlatformIO, dentro do ambiente Microsoft VSCode, seria uma alternativa interessante para a gerência de projetos IoT nas funções de configuração e programação do *end device* utilizado no protótipo. O PlatformIO, é um ambiente de desenvolvimento integrado para programação de sistemas embarcados, que permite desenvolver, depurar e programar microcontroladores de diferentes plataformas, incluindo Arduino, ESP8266, ESP32, STM32, entre outros. É uma ferramenta *open source* e multiplataforma, que oferece suporte a diversas linguagens de programação, incluindo C/C++, Python e Rust (PLATFORMIO, 2023).

De forma geral, a criação de projetos dentro do PlatformIO pode adicionar grande facilidade no gerenciamento de repositórios para prototipação IoT. Através do *Project Wizard* do ambiente em questão, é possível se definir o nome do projeto, placa a ser utilizada e também a *framework* que daria suporte ao desenvolvimento. A criação de um novo projeto dentro do PlatformIO gera um diretório característico, propondo um padrão de diretórios específico, em que um arquivo denominado *platformio.ini* serve como uma forma unificada de configuração dos projetos (PLATFORMIO, 2023). Este recurso representa uma vantagem para este trabalho, dada a necessidade de isolar e tornar replicável o funcionamento de diferentes projetos em ensaios de integração de sensores, por exemplo.

3.2.2 Integração dos Sensores DHT22 e PMS5003

Como mencionado anteriormente, a placa de desenvolvimento Heltec WiFi LoRa 32 v2 será a base para a implementação do protótipo do dispositivo periférico demonstrado neste trabalho. Para que isso ocorra, é de fundamental importância que haja disponibilidade de recursos compatíveis com a placa para a integração dos sensores DHT22 e PMS5003, que serão necessários para aquisição dos dados de interesse do projeto.

Para os propósitos de desenvolvimento deste projeto, considerando o uso da placa de desenvolvimento da Heltec programada através da *framework* Arduino, existem opções de bibliotecas *open-source* disponibilizadas pela comunidade Arduino, integráveis à Arduino IDE ou PlatformIO. Essas bibliotecas poderiam abstrair a decodificação desses sinais e possibilitar um processo de prototipação mais prático, reduzindo a complexidade aparente nas implementações de código necessárias.

No caso do DHT22, ARDUINO (2022a) disponibiliza uma biblioteca que integra o sensor ao ESP32 em relativo alto nível de abstração. Através de recursos como este, é possível a utilização de chamadas de códigos para realizar as leituras de temperatura e umidade do sensor, abstraindo a necessidade de programação em baixo nível para lidar com o protocolo *Single-Bus Communication*, necessário para o estabelecimento da comunicação com o sensor e posterior aquisição de dados do mesmo. Sendo assim, fica como alternativa secundária

meios utilizando menor nível de abstração, como a utilização do controle de níveis dos pinos GPIO e *timers* do microcontrolador ESP32, por exemplo.

De forma bastante análoga, ARDUINO (2022b) apresenta uma biblioteca para integração do sensor PMS5003 com o microcontrolador ESP32. Também se tratando da disponibilização de um recurso para a realização de chamadas para lidar com o sensor de materiais particulados em questão. Sendo assim, a necessidade de implementar funcionalidades para a tradução dos datagramas gerados pelo sensor fica como alternativa secundária, para casos em que esta biblioteca apresente incompatibilidades imprevistas ou dificuldade demasiada de utilização.

3.2.3 Implementação de End-node LoRaWAN com o LMIC

Neste projeto, a implementação de um *end device* LoRaWAN se faz necessária, já este seria responsável por capturar dados através dos sensores e enviá-los para um *gateway*. Nesse sentido, a utilização de bibliotecas denominadas LMIC (*LoRaWAN MAC In C*) se apresentam como uma das principais alternativas para o desenvolvimento de tais dispositivos. Este meio acabou sendo a opção primária para a utilização neste trabalho, considerando o uso da placa de desenvolvimento Heltec WiFi LoRa 32 v2.

Desenvolvido originalmente pelo IBM Zurich Research Laboratory, o conceito de biblioteca LMIC se apresenta como um recurso de pilha de *software* para facilitar a implementação de *end devices* LoRaWAN. Trata-se de um padrão de pilha de *software*, definido sob a forma de funções abstratas a serem implementadas que seriam essenciais para o funcionamento de um dispositivo periférico LoRaWAN (IBM CORPORATION, 2015). A ideia é que a implementação dessas funções, seguindo o padrão, possibilitaria que dispositivos (microcontroladores ou placas de desenvolvimento), de *software* embarcado compilado da linguagem C, pudessem incorporar funcionalidades necessárias básicas para atuarem como *end device* de uma rede LoRaWAN. Assim, as versões publicadas de definições sobre a LMIC se apresentam como guia dessas padronizações para implementações por outros agentes, como organizações e desenvolvedores independentes (IBM CORPORATION, 2015).

A LMIC clássica, como é ocasionalmente referenciada a padronização original da IBM, acabou tendo seu suporte descontinuado. No entanto, a MCCI Corporation já havia adaptado o padrão para que possuisse escopo centrado em dispositivos de *software* embarcado baseado na *framework* Arduino (MCCI CORPORATION, 2018). A versão da MCCI Corporation, focada na abstração e portabilidade do ambiente Arduino, se encontra ativa e tem sido uma alternativa bastante referenciada pela própria Semtech no contexto do desenvolvimento de *end devices* customizados, capazes de se integrar à redes LoRaWAN (SEMTECH, 2020a). Neste sentido, é possível encontrar implementações seguindo o padrão MCCI Arduino LMIC em diferentes repositórios de código aberto, como é o caso dos projetos disponibilizados por MCCI CATENA (2022) e PARENTE (2022).

A ideia de Arduino LMIC, mantida pela MCCI Corporation, tem por objetivo expor abs-

trações de funções para padronizar implementações de uma pilha de software que funcione como uma *framework*, tendo como base as chamadas padronizadas da Arduino *framework* (MCCI CORPORATION, 2018). Esse conjunto de possibilidades de chamadas ofereceria suporte à implementação de *end devices* de uma rede LoRaWAN a partir de microcontroladores e placas de desenvolvimento que possuem suporte da Arduino Framework e possuem acesso direto a um módulo de rádio LoRa.

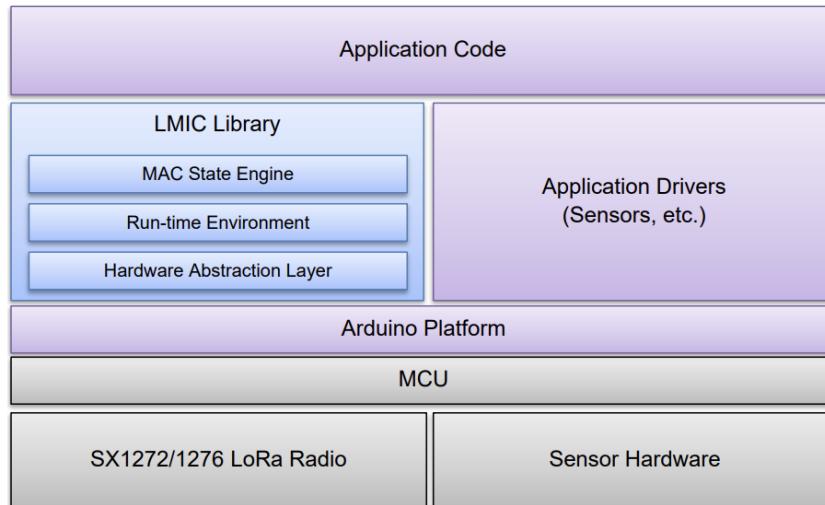
De acordo com a MCCI Corporation, a implementação de uma biblioteca LMIC de forma geral seria estruturada com módulos internos com funcionalidades específicas. Esses módulos atuariam na forma de camadas de abstração dentro do padrão, e teriam as seguintes denominações:

- ***Hardware Application Layer*** (HAL): camada de abstração que permite que o *software* LMIC acesse as funcionalidades de um *hardware* específico sem se preocupar com os detalhes do mesmo. Isso permite que outras chamadas da biblioteca LMIC utilizada possam ser executadas em várias plataformas diferentes (MCCI CORPORATION, 2018).
- **Ambiente de Tempo de Execução** (*Run-time Environment*): ambiente no qual a LMIC é executada, que inclui bibliotecas, *drivers* e outros recursos necessários para suportar o funcionamento das chamadas do padrão. Pode incluir, por exemplo, bibliotecas de temporização e interrupção, bibliotecas de comunicação serial, gerenciamento de memória, etc. O ambiente de tempo de execução é específico para a plataforma em que o LMIC está sendo executado e é geralmente fornecido pelo fabricante da plataforma. Ele deve ser compatível com a camada de abstração de hardware (HAL) do LMIC para permitir que ele acesse o *hardware* de forma adequada (MCCI CORPORATION, 2018).
- **Motor de Estado MAC** (*MAC State Engine*): parte da LMIC que gerencia o estado atual da comunicação LoRaWAN, garantindo que o protocolo de comunicação funcione por parte do *end device*. Implementa as regras de negócios e lógicas para garantir que as mensagens sejam enviadas e recebidas corretamente. Também gerencia as operações de transmissão e recepção, incluindo a agendamento de transmissões, gerenciamento de colisões e garantia de qualidade do serviço (QoS). Ele é responsável por gerenciar a comunicação entre o dispositivo final (*end device*) e a rede LoRaWAN, garantindo que o dispositivo final esteja sempre em conformidade com as regras da rede (MCCI CORPORATION, 2018).

No caso da MCCI Arduino LMIC, esta pilha de funcionalidades seria cliente das chamadas Arduino, ao mesmo tempo que disporia seus recursos para que o código da aplicação desenvolvida pudesse fazer uso da camada MAC do LoRaWAN estabelecida. Neste sentido, a aplicação seria o nível mais alto e teria à sua disposição as tanto as chamadas da LMIC para

se comunicar utilizando o protocolo LoRaWAN, quanto a leitura de sensores que o *end device* estaria portando, por exemplo. A Figura 21 mostra uma idealização do uso da Arduino LMIC na composição de um *end device* LoRaWAN.

Figura 21 – Camadas de abstração propostas pelas definições LMIC da MCCI



Fonte: MCCI CORPORATION (2018)

3.3 SOFTWARE

3.3.1 *The Things Stack (TTS) Community Edition*

Como mencionado na Seção 2.4.1 deste trabalho, em uma rede LoRaWAN, o servidor de rede desempenha um papel essencial para a comutação de mensagens entre dispositivos periféricos e servidores para aplicações associadas. Uma vez que existe a necessidade de implementação de uma aplicação voltada para o monitoramento de qualidade do ar no projeto aqui proposto, é necessário que se tenha à disposição uma forma conveniente de implementar esta parte da arquitetura de forma viável, dadas restrições orçamentárias e práticas deste trabalho.

Neste sentido, *The Things Industries* (TTI) se apresenta como uma organização cujos objetivos se relacionam principalmente com o fornecimento soluções para tornar mais prática a implantação de servidores de rede LoRaWAN. Essas soluções são desenvolvidas e mantidas pela TTI sob a forma de uma pilha de *software open source* que foi batizada de *The Things Stack* (TTS) (TTI, 2022b). A distribuição do TTS acaba sendo realizada de diferentes formas, que diferem principalmente no que diz respeito ao tipos de possibilidade de suporte técnico e também a diferentes formas de alocação de infraestrutura para a implantação desses servidores (TTI, 2022a).

Para implantações *on premise*, por exemplo, onde se considera o uso de infraestrutura de *hardware* própria para executar o servidor de rede, a TTI disponibiliza distribuições como a *Open Source* e *Enterprise*. Resumos gerais sobre essas alternativas são apresentados a seguir:

- **TTS Open Source:** é disponibilizada gratuitamente e pode ser baixada e utilizada. Trata-se de uma distribuição que é oferecida pelo TTI, mas que acaba tendo manutenção suportada pela comunidade de usuários voluntários, sendo que os mesmos podem contribuir com o código fonte e desenvolver novos recursos (TTI, 2022a). Esse modelo de distribuição permite que um dispositivo com baixo poder de processamento como o Raspberry Pi possa ser utilizados como plataforma funcional de execução do TTS (PEREZ, 2023).
- **TTS Enterprise:** é uma distribuição comercial do TTS, em que é disponibilizado suporte técnico de agentes mantenedores da TTI, assim como garantia de disponibilidade de serviço e outros recursos adicionais. Trata-se de um seguimento de distribuição geralmente voltado para empresas, onde é necessário garantir a continuidade dos serviços e escalabilidade (TTI, 2022a).

Para casos onde se deseja implantar o TTS como serviço *on cloud*, a TTI disponibiliza possibilidades para utilização de instâncias que assumem a forma de serviços de forma paga. É neste formato em que se baseiam as distribuições *Cloud*, *Dedicated Cloud* e *AWS Launcher*. Uma breve explicação do que seriam essas possibilidades é apresentada a seguir:

- **TTS Cloud:** é uma versão em nuvem paga do TTS, na qual são oferecidos benefícios principalmente relacionados a suporte técnico. Oferece a flexibilidade de usar a plataforma sem que haja a necessidade de manutenção física/lógica de uma infraestrutura, garantindo escalabilidade, disponibilidade de serviço e de suporte técnico. Essa versão é geralmente usada por empresas que buscam escalabilidade e agilidade para gerenciar suas redes LoRaWAN (TTI, 2022a).
- **TTS Dedicated Cloud:** trata-se de uma forma de distribuição paga baseada em nuvem que é operada e gerenciada pelo próprio usuário. Ele fornece aos usuários uma plataforma pronta para uso, escalável e que promete ser altamente disponível para se construir e gerenciar soluções IoT. Esta forma de implantação é instalada e gerenciada em uma infraestrutura de nuvem dedicada (como AWS, Azure ou GCP) escolhida pelo usuário (TTI, 2022a).
- **TTS AWS Launcher:** é uma categoria de distribuição do TTS que é comercializada na forma de um serviço da *Amazon Web Services* (AWS). Este formato permite a iniciação rápida de uma instância do TTS diretamente à partir de uma conta do *cloud provider* da Amazon, oferecendo uma solução pré-configurada e pronta para uso que inclui o gerenciamento de dispositivos, coleta de dados, processamento, armazenamento e visualização de dados IoT. O usuário pode escolher entre diferentes opções de configuração, como o tamanho da instância e o tipo de armazenamento, podendo personalizar essas configurações antes de iniciar a implantação (TTI, 2022a).

Como iniciativa associada à *The Things Industry*, existe também uma alternativa denominada *The Things Network* (TTN), que apresenta uma solução na nuvem, para propósitos relacionados a fins não comerciais. De forma geral, o TTN se propõe a ser um ecossistema global, que utiliza o TTS na provisão do seu serviço, chamando sua plataforma *on cloud* de TTS *Community Edition* (TTI, 2022b).

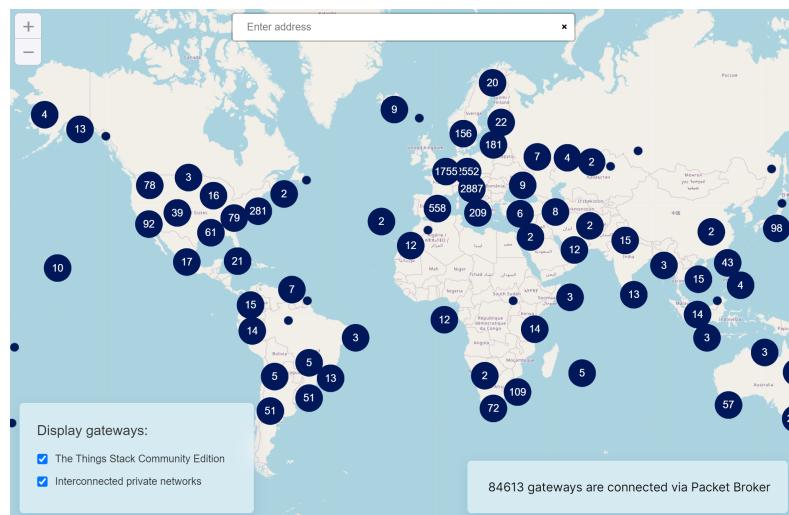
Essa forma de distribuição se faz presente como plataforma *open source* e gratuita, projetada para ser fácil de usar e acessível para pessoas de todos os níveis de habilidade e setores, permitindo que qualquer um possa construir soluções IoT e experimentar o uso da tecnologia LoRaWAN, sem necessidade de investimento significativo. A alternativa permite que os usuários construam e gerenciem interfaces abstratas com aplicações por meio do uso de um servidor de rede LoRaWAN público, de forma a não necessitar da utilização de uma infraestrutura *on premisses* ou da contratação de *cloud providers* de forma paga (TTN, 2022b).

O serviço expõe redes LoRaWAN públicas criadas e gerenciadas por usuários na forma de uma comunidade de voluntários, que colaboram para integrar e manter *gateways* LoRaWAN em localidades diversas ao redor do globo. Deste modo, o TTN acaba também fomentando a existência de uma ampla comunidade para o desenvolvimento colaborativo de

soluções IoT utilizando o protocolo LoRaWAN, viabilizando discussões e ambientes públicos para compartilhamento de conhecimento técnico sobre a tecnologia (TTN, 2022a).

Através do TTS *Community Edition* provisionado pelo TTN, no entanto, existem certas limitações e termos de uso no sentido de limitar tráfego excessivo ou indevida utilização do serviço (TTN, 2022a). Esta forma de serviço não inclui suporte técnico direto do *The Things Industries* e só pode ser executado em uma única região do provedor de nuvem selecionado. Além disso, o serviço do TTN expõe os dados obtidos na sua rede comunitária, possuindo limitações de escalabilidade, sem oferecer garantia de disponibilidade dos serviços (TTN, 2022a).

Figura 22 – Disposição geográfica de gateways públicos da TTN



Fonte: TTN (2022b)

No contexto da proposta de desenvolvimento de um protótipo para este trabalho, a distribuição *Community Edition* do TTS, através do TTN, pareceu uma alternativa bastante conveniente. Através desta plataforma, seria possível a utilização de uma abstração de uma rede LoRaWAN comunitária sem custo algum, o que se fez interessante em um contexto de prototipação para fins acadêmicos. Uma vez que não haveriam preocupações iniciais com a exposição de dados gerados pelos sensores, nem com a disponibilidade do serviço, não haveria problemas em testar o protótipo utilizando a plataforma.

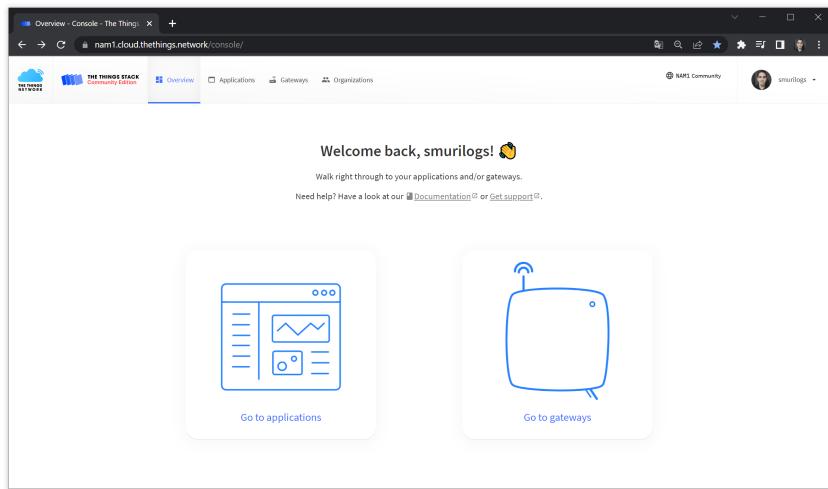
O TTS *Community Edition* se faz disponível através de um console do TTN, permitindo o registro de *gateways* e *end devices*, bem como a customização da forma que dados são entregues a abstrações de aplicações criadas. Além disso, o serviço provisionado pelo TTN apresenta uma documentação robusta, onde é encontrada grande conformidade com os conceitos apresentados na fundamentação teórica deste trabalho (TTN, 2022a). Nesse sentido, quantidade considerável de configurações relacionadas a parâmetros ajustáveis, modos de ativação, classes de dispositivos são colocadas à disposição do desenvolvedor.

Também são muitas as possibilidades disponibilizadas pela plataforma para suportar a troca de dados com aplicações externas. É possível, por exemplo, conexões diretas com

outros serviços IoT de diferentes *cloud providers*, como *AWS IoT*, *Azure IoT Hub* e *LoRa Cloud*. São também contempladas integrações dedicadas à comunicação com plataformas *no-code*, como o *Red-NODE* ou *IFTTT* (TTN, 2022a).

Além disso, através de um recurso denominado de *storage integration*, são expostos meios como gRPC (*Google Remote Procedure Calls*) ou API (*Application Programming Interface*) para o acesso aos dados produzidos por mensagens de *uplink*, sendo também possível a exposição de tópicos para consumo dos dados na plataforma por clientes MQTT (*Message Queuing Telemetry Transport*). É importante ressaltar a utilização do *storage integration* no TTN possui limitações, sendo uma das mais destacada a restrição em de 24 horas de retenção dos dados em tráfego pela plataforma (TTN, 2022a).

Figura 23 – Console do TTS *Community Edition*



Fonte: Autoria própria (2023)

Essas possibilidades se apresentam como formas pelas quais a aplicação criada por este trabalho poderá se comunicar com rede neutra com o intuito de estabelecer um sistema de monitoramento IoT. Neste sentido, o TTS permite a criação de adaptações lógicas para aplicações, onde atribuições gerais sobre as mesmas são feitas com relação a sua operação dentro de uma topologia suportada pelo protocolo LoRaWAN (TTN, 2022a).

São definidos, dentro do TTS, não somente os formatos pelos quais essas aplicações trocarão dados com os dispositivos através do servidor de rede, mas também definições essenciais para o funcionamento geral da própria aplicação como elemento da topologia, como designações de códigos e chaves *AppEUI* e *AppKey*, necessárias para o estabelecimento da comunicação com dispositivos periféricos operando no modo OTAA. O modo de ativação ABP também é possível, sendo que neste caso necessário nesses casos atribuições prévias de *NwkSkey* e *AppSKey* aos dispositivos periféricos associados à aplicação no próprio TTN. Adaptações lógicas de *gateways*, aplicações e *end devices* são criadas dentro do próprio console do ecossistema (TTN, 2022a).

3.3.2 Ecossistema da Linguagem Python

A aplicação que dará forma ao sistema de monitoramento proposto neste trabalho tem à disposição uma grande quantidade ferramentas através das quais a mesma pode ser construída. Dentro desse contexto, a priorização por uma ferramenta em detrimento de outra aqui possui critérios relacionados a menor custo, possibilidade de menores níveis de acoplamento entre sistemas e familiaridade do autor na utilização desses meios. O ecossistema da linguagem de programação Python pareceu possuir grande conformidade com esses critérios e boa capacidade de lidar com a necessidade de implementação do sistema de monitoramento de qualidade do ar aqui proposto.

Como reportando da Seção 3.3.1 anterior, este projeto opta por priorizar a utilização do recurso de *storage integration* do TTN como uma forma de obtenção dos dados trafegados. Este recurso permite que esses dados sejam acessados através de *endpoints* de uma API exposta pelo ecossistema do TTN. Deste modo, seria possível o acesso aos dados retidos para posterior persistência própria, fora do ecossistema do TTN. Uma outra opção que foi considerada para isso foi a implementação dessa integração através da utilização do protocolo MQTT (*Message Queuing Telemetry Transport*) como uma forma de baixo acoplamento na interação entre o TTN e a aplicação a ser desenvolvida. Esta ultima foi considerada uma alternativa importante à disposição.

3.3.2.1 *SQLite* e *SQLAlchemy*

No sentido de garantir a persistência dos dados de interesse deste projeto, à despeito das limitações impostas pelo uso do serviço do TTN, foi considerada como uma opção interessante o suporte à manipulação do banco de dados *SQLite*, que é garantido pela biblioteca padrão das versões mais recentes do Python, como a versão 3.11.1. Isso pode ser feito através do uso do pacote *sqlite3*, que lida com a versão 3 do banco de dados (PYTHON, 2023). Esta opção se mostra adequada considerando as vantagens que o *SQLite* apresenta em termos de simplicidade e portabilidade na sua forma de armazenamento. Os dados armazenados pelo *SQLite* podem ser dispostos em um único arquivo, comparativamente bastante leve, e que pode ser alocado dentro do diretório do projeto (SQLITE, 2023). Essas características podem se tornar ideais para a implementação de um sistema protótipo, desde que se tenha atenção para as potenciais necessidades de expansão do projeto para trabalhos futuros.

Pensando nas limitações que o uso do *SQLite* poderia trazer enquanto banco de dados minimalista, tornou-se interessante também a utilização de um ORM (*Object Relational Mapper*) disponível na linguagem Python chamado *SQLAlchemy*. Esta ferramenta permitiria viabilizar a adaptação de bancos de dados relacionais com o sistema de monitoramento, possibilitando que o sistema de persistência de dados utilizado possa ser intercambiável com relativo baixo custo técnico. Além desse desacoplamento, o *SQLAlchemy* permite a desenvolvedores trabalhar com bancos de dados relacionais utilizando uma abordagem

orientada a objetos, sendo capaz de abstrair as complexidades do SQL e oferecer uma interface mais simples e intuitiva para interagir com o banco de dados (SQLALCHEMY, 2023).

Além disso, *SQLAlchemy* possui compatibilidade com vários bancos de dados relacionais, incluindo *SQLite*, *MySQL*, *MariaDB*, *PostgreSQL*, *Oracle*, *Microsoft SQL Server*, dentre outros. Nesse sentido, desenvolvedores podem usar o ORM em projetos que envolvem diferentes tipos de bancos de dados, de forma intercambiável, sem precisar alterar a lógica de negócios da aplicação. O *SQLAlchemy* também fornece recursos de migração de banco de dados, que ajudam a manter o esquema do banco de dados sincronizado com as mudanças no modelo de dados da aplicação. Isso é particularmente útil em projetos em que o esquema do banco de dados é atualizado frequentemente (SQLALCHEMY, 2023).

3.3.2.2 *Pandas* e *Streamlit*

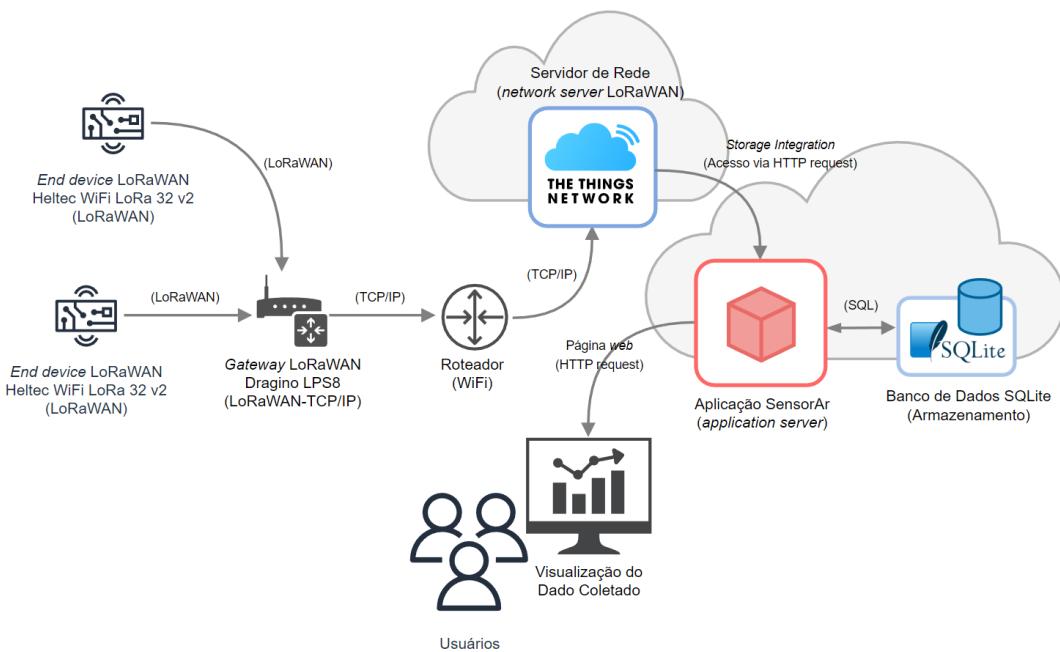
Considerando as necessidades de transformações com os dados obtidos para se chegar aos valores do IQAr e cores associadas, seria necessário facilitar cálculos e manipulação com dados em estruturas tabulares, armazenados no banco de dados SQLite criado para o projeto. Para tal, este projeto opta pela utilização da biblioteca *Pandas*, uma ferramenta de código aberto que é disponibilizada na linguagem de programação Python, utilizada para transformações para contextos diversos. A biblioteca *Pandas* fornece estruturas de dados com performance relativamente alta e ferramentas de manipulação de dados para a análise e transformação de dados de diversos formatos. Uma das principais vantagens do *Pandas* é a capacidade de trabalhar com grandes conjuntos de dados de forma rápida e eficiente. Isso é possível graças ao uso de estruturas de dados como o *dataframe*, que permite que os dados sejam organizados em tabelas bidimensionais com colunas nomeadas e linhas indexadas. Com o *dataframe*, é possível realizar operações de agregação, filtragem e transformação de dados com facilidade (PANDAS, 2023), meios se farão necessários no contexto deste trabalho.

Sendo também necessária a apresentação desses dados em uma *dashboard* capaz de apresentar informações sobre o contexto de presença de materiais particulados, a biblioteca *Streamlit* se apresentou como uma alternativa no ecossistema de ferramentas *open source* na linguagem Python. Trata-se de uma ferramenta utilizada para a criação de aplicações *web* interativas, que tem sido utilizada para análise de dados, modelagem de modelos de *machine learning* e outras aplicações. Ela foi criada com o objetivo de simplificar o processo de desenvolvimento de aplicações *web*, tornando-o mais rápida e intuitiva, sendo uma das suas principais vantagens o fato de que com poucas linhas de código é possível criar uma aplicação *web* completa, com interface de usuário interativa e visualizações de dados (STREAMLIT, 2023). Isso é possível graças à abordagem baseada em *widgets* da biblioteca, que permite que os desenvolvedores criem elementos interativos, como botões, barras de rolagem e caixas de seleção. O *Streamlit* possui também a capacidade de atualizar visualizações de dados em tempo real, permitindo que os usuários vejam as alterações nos dados conforme elas ocorrem (STREAMLIT, 2023).

3.4 ARQUITETURA PROPOSTA

Considerando as possibilidades técnicas de *hardware*, *firmware* e *software* demonstradas nas seções supracitadas, a implementação dos sistema de monitoramento de qualidade do ar terá como proposta a arquitetura demonstrada na Figura 24.

Figura 24 – Arquitetura do sistema de monitoramento proposta



Fonte: Autoria própria (2023)

Como detalhado anteriormente, esse esquema prevê a utilização da placa de desenvolvimento Heltec WiFi LoRa 32 v2 na função de *end device* e do Dragino LPS8 na função de *gateway*. Neste arranjo, é atribuído ao *end device* a utilização os sensores DHT22 e PMS5003 como forma de aquisição de dados de temperatura/umidade relativa e presença de materiais particulados MP₁₀/MP_{2,5}, bem como o seu envio desses dados até a rede neutra. Já o *gateway* desempenhará o papel de ponte de conexão entre o *end device* e o servidor de rede.

Uma vez que o serviço do TTN pudesse desempenhar a função de servidor de rede LoRaWAN neste contexto, uma aplicação criada em Python, conseguiria realizar chamadas a *endpoints* da API exposta pelo TTN (no modo *storage integration*) para acesso aos dado de tráfego na rede. Esse acesso garantirá que a aplicação seja capaz de armazenar os dados coletados em um sistema de persistência próprio, que seria materializado na forma de um banco de dados SQLite. Sendo assim, esses dados armazenados, poderiam ser utilizados pelo ecossistema do Python para criar ferramentas para a apresentação de dados na forma de uma *dashboard*.

4 RESULTADOS E DISCUSSÃO

4.1 MONTAGENS DE HARDWARE PARA O END-DEVICE

A estratégia adotada para efetuar as montagens de *hardware* neste trabalho foi a de primeiro testar individualmente a viabilidade das leituras realizadas pelos sensores selecionados para o protótipo. Neste sentido, utilizando o Heltec WiFi LoRa 32 v2 como unidade central para integração dos sensores, criou-se rotinas de ensaios individuais para a captura das variáveis de interesse e apresentação das mesmas utilizando o monitor serial, tanto para o DHT22 quanto para o PMS5003.

Essas leituras seriam importantes para capacitação do autor na tarefa de refatoração desses códigos na forma de recursos que pudessem ser integrados no código da aplicação de um *end device* LoRaWAN. Assim, poderia-se realizar essas leituras de forma simultânea e adaptá-las para o envio de *payloads* de dados através de uma biblioteca LMIC, como será relatado mais a frente neste trabalho. O progresso das montagens para a realização desses ensaios e seus resultados serão relatados nas seções subsequentes.

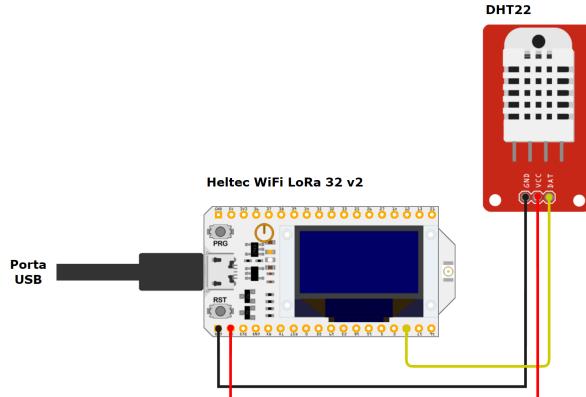
4.1.1 Ensaio de Integração do DHT22

A princípio, a placa Heltec WiFi LoRa 32 v2 foi conectada à porta USB de computador pessoal, no intuito de prover alimentação necessária para o funcionamento do dispositivo e, posteriormente, sensores que seriam acoplados. Utilizou-se os próprios terminais de saída de 5 volts e de referência GND da placa de desenvolvimento para estender a alimentação até o DHT22. Esses dois contatos foram conectados de forma correspondente aos terminais de alimentação de VCC e GND do sensor, enquanto o terminal de dados teria que ser conectado a um dos pinos GPIO que seria utilizado para o trabalho de leitura utilizando o protocolo *Single-Bus Communication*, como mencionado na seção 3.1.2 deste trabalho, relativa ao sensor em questão.

Observou-se neste momento a necessidade de lidar com escolhas de pinos utilizados para a troca de dados entre os dois dispositivos. Isso porque, vários recursos físicos da placa de desenvolvimento da Heltec, como o seu *display* OLED ou sua interface com o rádio SX1276 fixado na placa, compartilham pinos específicos do microcontrolador ESP32. Uma vez que não houvesse cuidado para lidar com esses o uso desses contatos, poderiam haver conflitos no uso dos mesmos no momento da integração dos sensores.

Foi tomado cuidado com essas possibilidades de conflitos que optou-se pela utilização do pino 4, uma vez que este pareceu uma boa alternativa. Sendo assim, o pino referido foi utilizado para a troca de dados com o sensor DHT22 e seria responsável por decodificar os pacotes trocados através do *Single-Bus Communication*. Um diagrama da montagem para o ensaio de integração da placa de desenvolvimento com o sensor é demonstrada na Figura 25.

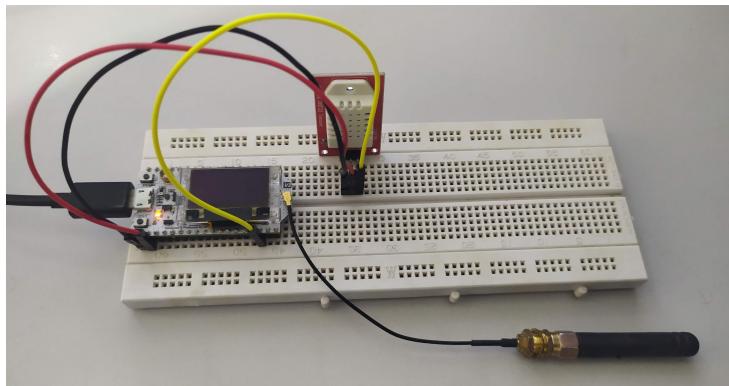
Figura 25 – Diagrama da montagem do ensaio de integração do DHT22



Fonte: Autoria própria (2023)

Uma foto da montagem real para este ensaio foi registrada pelo autor. Na imagem, as cores dos fios correspondem às cores apresentadas no diagrama da Figura 25. O registro fotográfico é mostrado na Figura 26.

Figura 26 – Montagem real do ensaio de integração do DHT22



Fonte: Autoria própria (2023)

Com a montagem pronta, o próximo passo foi instalar as bibliotecas necessárias para a manipulação do sensor. Como descrito da fundamentação teórica, a primeira opção foi utilizar a biblioteca *DHT Sensor Library*, disponibilizada para a comunidade pela Adafruit. Assim, utilizou-se a interface do PlatformIO para buscar e instalar a biblioteca em questão.

Descobriu-se nesse processo que, apesar de o projeto PlatformIO ter sido configurado para o uso da placa de desenvolvimento Heltec WiFi LoRa v2, essa definição precisa ser complementada pela inclusão de uma biblioteca de definições da placa sobre a qual a *framework* Arduino pudesse atuar. Neste sentido, observou-se que para esses casos é comum a utilização da biblioteca *Heltec ESP32 Dev-Boards* (HELTEC, 2022). Este recurso teve que ser instalado para que o PlatformIO pudesse reconhecer a placa em tarefas de construção do binário a ser carregado no ESP32 da placa.

Após essas descrições no arquivo *platformio.ini* do projeto, tendo a inclusão das bibliotecas mencionadas como incrementos mais específicos do ensaio, pôde-se então testar com sucesso o carregamento de códigos na placa, sem que houvessem erros. A Figura 27 e mostra o conteúdo do arquivo de configuração *platformio.ini* após a configuração do projeto e a instalação das dependências utilizadas. Este código também é destacado no *Apêndice A*.

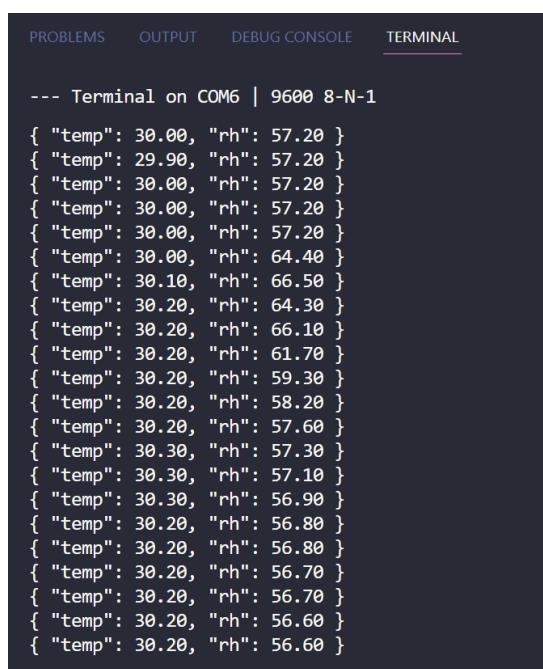
Figura 27 – Arquivo *platformio.ini* para ensaio com o DHT22

```
[env:heltec_wifi_lora_32_V2]
platform = espressif32
board = heltec_wifi_lora_32_V2
framework = arduino
lib_deps =
    heltecautomation/Heltec ESP32 Dev-Boards@^1.1.1
    adafruit/DHT sensor library@^1.4.4
```

Fonte: Autoria própria (2023)

Escreveu-se então um código para o ensaio, demonstrando a capacidade do sensor de realizar leituras periódicas consistentes das variáveis de umidade relativa do ar (em porcentagem) e de temperatura (em graus Celcius). O *script* criado para realizar essa tarefa está disponível no *Apêndice B*. Este algoritmo realiza uma breve exibição dos resultados das leituras do sensor foi programada para ser apresentada utilizando a interface UART conectada à porta USB da placa de desenvolvimento da Heltec. Essa exibição no monitor serial utilizado pode ser visualizada na Figura 28.

Figura 28 – Monitoramento serial das leituras realizada no ensaio com o DHT22



The screenshot shows a terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, displaying the following text:

```
--- Terminal on COM6 | 9600 8-N-1
{
  "temp": 30.00, "rh": 57.20
}
{
  "temp": 29.90, "rh": 57.20
}
{
  "temp": 30.00, "rh": 64.40
}
{
  "temp": 30.10, "rh": 66.50
}
{
  "temp": 30.20, "rh": 64.30
}
{
  "temp": 30.20, "rh": 66.10
}
{
  "temp": 30.20, "rh": 61.70
}
{
  "temp": 30.20, "rh": 59.30
}
{
  "temp": 30.20, "rh": 58.20
}
{
  "temp": 30.20, "rh": 57.60
}
{
  "temp": 30.30, "rh": 57.30
}
{
  "temp": 30.30, "rh": 57.10
}
{
  "temp": 30.30, "rh": 56.90
}
{
  "temp": 30.20, "rh": 56.80
}
{
  "temp": 30.20, "rh": 56.80
}
{
  "temp": 30.20, "rh": 56.70
}
{
  "temp": 30.20, "rh": 56.70
}
{
  "temp": 30.20, "rh": 56.60
}
{
  "temp": 30.20, "rh": 56.60
}
```

Fonte: Autoria própria (2023)

4.1.2 Ensaio de Integração do PMS5003

De forma similar ao que foi feito com o DHT22, um ensaio de integração para o sensor PMS5003 também foi realizado a fim de entender o funcionamento do sensor, bem como definir os recursos necessários para a integração do mesmo. O trabalho desenvolvido no código do ensaio possibilitaria o incremento do código de ensaio de integração criado para o DHT22 a fim de formar uma unidade de lógica de sensores conveniente para a finalidade do projeto. Isso porque, uma vez que essa unidade pudesse ser construída, poderia-se utilizar uma biblioteca LMIC para implementar um dispositivo periférico que enviasse os dados de leituras dos dois sensores em conjunção.

O primeiro passo da integração do PMS5003 foi conectar placa Heltec WiFi LoRa 32 v2 a um computador pessoal utilizando conexão USB, com o propósito de alimentar o dispositivo e o sensor que seria acoplado. Os terminais de saída de 5 volts e de referência GND da placa foram usados para a alimentação do PMS5003, sendo esses dois terminais conectados aos terminais *VCC* e *GND* do sensor, respectivamente.

Como forma de estabelecer comunicação entre o sensor e a placa de desenvolvimento, os pinos de transmissão *TX* e recebimento *RX* do PMS5003 teriam que ser conectados a uma interface UART da placa de desenvolvimento. Como é possível observar no diagrama de pinos mostrado em sua documentação, a Heltec WiFi LoRa 32 v2 expõe duas dessas interfaces disponíveis pelo seu microcontrolador ESP32 através de contatos de pinos da placa de desenvolvimento (HELTEC, 2018). No entanto, um desses pares de pinos, relativos à *UART0* (referenciados como *TX* e recebimento *RX* no diagrama de pinos), são utilizados pelo *bootloader* da placa para que a mesma consiga ser programada através do conector USB. Ou seja, utilizar essa interface, causaria riscos de compartilhamento indevido das funções desses contatos em tarefas muito básicas, relacionadas ao carregamento de código na placa.

A *UART2* da placa (exposta através dos pinos 16 e 17) pareceu a alternativa imediata. No entanto, a documentação da Heltec chama atenção para o fato de que o pino 16 compartilha funcionalidades com o controle do *display* OLED presente na placa (HELTEC, 2018). Isso poderia representar mais uma situação de risco de compartilhamento indevido de recursos do microcontrolador ESP32, que poderiam impedir a efetividade das leituras do PMS5003.

Novas alternativas foram cogitadas no sentido de tentar contornar esses obstáculos. Uma delas foi a utilização de bibliotecas para implementação de interfaces seriais via *software* apresentada por ARDUINO (2023). No entanto, essa opção traria incertezas: uma vez que esse tipo de implementação requer, por vezes, a utilização de recursos como interrupções e *timers*, esse uso poderia acabar impedindo a coexistência dos códigos de leitura com as funcionalidades de biblioteca LMIC, caso as duas implementações competisse pelos mesmos recursos.

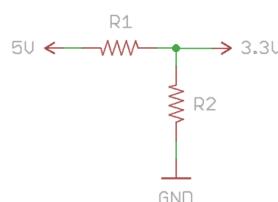
Uma solução considerada tecnicamente mais segura foi encontrada em por meio de uma funcionalidade do microcontrolador ESP32, que permite suas interfaces UART possam ser roteadas para outros contatos do dispositivo, diferentes dos estabelecidos por

padrão. Na placa de desenvolvimento da Heltec, isso significaria uma possibilidade de se redesignar os pinos utilizados por uma interface UART do microcontrolador fixado na placa. Essa redesignação pôde ser realizada através da passagem de parâmetros como argumentos na criação do objeto *Serial*, utilizando a *framework* Arduino. Neste sentido, TX e RX da *UART2* puderam ser redirecionados para os pinos 32 e 33, respectivamente, disposição na qual não haveriam riscos de colisões na utilização desses contatos.

Com a interface *UART2* configurada para ser acessada através de pinos alternativos, mais uma ponderação teve que ser feita: a avaliação das especificações da placa de desenvolvimento e do sensor trouxe dúvida à capacidade do pino RX do PMS5003 de suportar tensões acima de 3,3 volts. Considerando que os níveis lógicos dos pinos atribuídos à placa de desenvolvimento da Heltec alcançam 5 volts, isso poderia ser um potencial causador de danos ao sensor. Não havendo margens para falhas nesse sentido, optou-se então por assegurar uma adequação da interface onde essas tensões pudessem ser respeitadas. Essa adequação envolveu o dimensionamento de um circuito divisor de tensão para evitar que a linha RX do sensor tivesse contato com 5 volts alcançados pelo pino TX da UART utilizada.

O divisor de tensão dimensionado, garantiria que o TX da interface *UART2* da placa de desenvolvimento alcançaria no máximo 3,3 volts, o que adequaria o sinal a um nível tolerável para o contato RX do PMS5003. O circuito adicional, necessitou de apenas dois resistores, conectados em série, como mostrado no esquema na Figura 29. Neste tipo de arranjo, a tensão de saída é medida entre os terminais dos resistores, sendo que a Equação (4.1) é usada para calcular a tensão de saída, dada uma tensão de entrada e as resistências R_1 e R_2 dos resistores.

Figura 29 – Esquema do divisor de tensão a ser utilizado



Fonte: Autoria própria (2023)

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2} \quad (4.1)$$

onde:

V_{out} é a tensão de saída

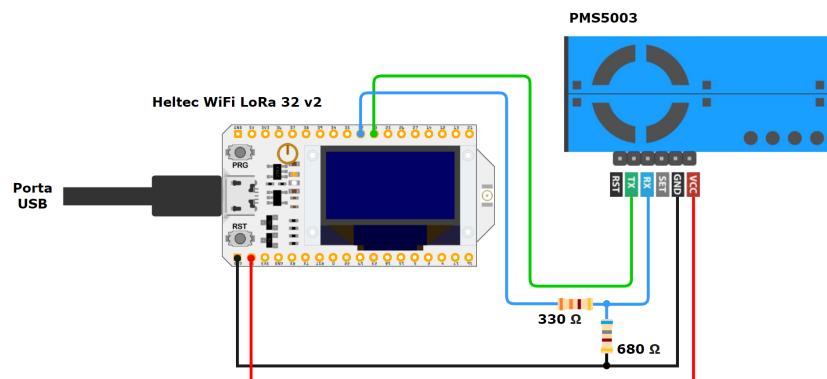
V_{in} é a tensão de entrada

R_1 é a resistência do resistor superior

R_2 é a resistência do resistor inferior

De forma geral, para que se houvesse uma redução de tensão de 5 para 3,3 volts seria necessário que R_1 possuísse metade do valor de R_2 , sendo que esta, em termos absolutos, não precisariam passar de $1\text{k}\Omega$, pois não havia a necessidade de interferir na corrente de forma muito brusca. Os resistores utilizados na prática acabaram assumindo valores de 330Ω e 680Ω para as resistências R_1 e R_2 , respectivamente. Considerando que o valor máximo apresentado de entrada seria de 5 volts e que este seria o valor de V_{in} na Equação (4.1), o cálculo para a saída máxima V_{out} resultou em 3,37 volts. Este resultado pareceu adequado para se utilizar no condicionamento desse sinal e o experimento seguiu utilizando esses valores. O diagrama da montagem em *protoboard* para o ensaio com o sensor é demonstrado na Figura 30.

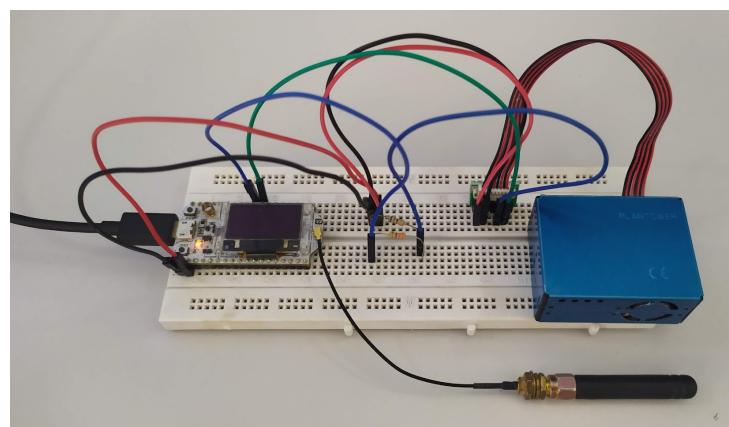
Figura 30 – Ensaio de integração do PMS5003



Fonte: Autoria própria (2023)

Foi registrada uma foto da montagem real deste ensaio em *protoboard*, com as cores dos fios respeitando as cores utilizadas nas conexões do diagrama da Figura 30. O registro fotográfico é mostrado na Figura 31.

Figura 31 – Montagem real do ensaio de integração do PMS5003



Fonte: Autoria própria (2023)

O próximo passo após a montagem para este ensaio foi instalar as bibliotecas que seriam utilizadas pelo código escrito para o ensaio de leituras pelo PMS5003. Essas instalações

foram feitas em um novo projeto PlatformIO, para que se isolasse a forma como foram feitas as configurações e gerenciamento de dependências em cada um dos ensaios. Assim, partiu-se para a instalação de recursos necessários para a manipulação do PMS5003 e para o reconhecimento da placa de desenvolvimento da Heltec para a construção do binário e carregamento do mesmo no ESP32 fixado na placa.

No caso das necessidades relativas às leituras de variáveis sobre presença de materiais particulados utilizando o PMS5003, foram instaladas as bibliotecas *Adafruit PM25 AQI Sensor* e *Adafruit Unified Sensor*. Essas bibliotecas se apresentam como soluções complementares para a manipulação do sensor, criada pela Adafruit com popularidade aparentemente satisfatória junto à comunidade Arduino. O combo consegue atuar em conjunto para expor métodos relacionados a configuração e leituras de dados coletados pelo sensor de uma forma geral. Além disso, Também foi instalada a *PMS Library* uma biblioteca *wrapper* que utiliza as bibliotecas da Adafruit já citadas para disponibilizar métodos com maior nível de abstração, com ênfase na leitura simplificada de dados relativos à valores de MP₁₀, MP_{2,5} e MP_{1,0} e controle de modos de funcionamento do sensor.

Além disso, assim como no caso do ensaio anterior com o DHT22, houve neste ensaio com o PMS5003 a necessidade de instalar a biblioteca *Heltec ESP32 Dev Boards*. De forma geral, essa biblioteca acaba sendo utilizada como um complemento das configurações de projeto utilizando a placa de desenvolvimento da Heltec para que o projeto PlatformIO consiga reconhecimento do dispositivo para tarefas de construção do binário e carregamento do mesmo no seu microcontrolador.

Com a instalação dessas dependências dentro do novo projeto PlatformIO criado, houve o incremento automático do arquivo *platformio.ini*. No arquivo, é possível observar as bibliotecas mencionadas inclusas, sendo que utilizando essas configurações e dependências pôde-se testar com sucesso o carregamento de códigos de demonstração, sem que houvessem erros de carregamento desses recursos. A Figura 32 mostra o conteúdo do arquivo de configuração *platformio.ini* após a configuração do projeto e as instalações das dependências utilizadas. Este código também é referenciado no *Apêndice C* deste trabalho.

Figura 32 – Arquivo *platformio.ini* para ensaio com o PMS5003

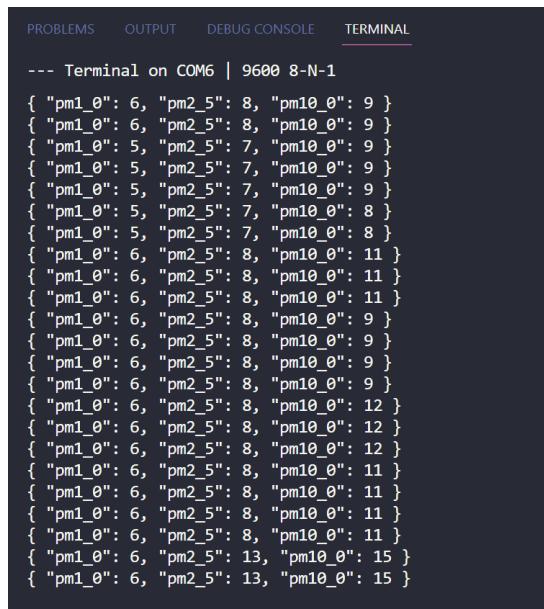
```
[env:heltec_wifi_lora_32_V2]
platform = espressif32
board = heltec_wifi_lora_32_V2
framework = arduino
lib_deps =
    heltecautomation/Heltec ESP32 Dev-Boards@^1.1.1
    adafruit/Adafruit PM25 AQI Sensor@^1.0.6
    adafruit/Adafruit Unified Sensor@^1.1.7
    fu-hsi/PMS Library@^1.1.0
```

Fonte: Autoria própria (2023)

A partir deste ponto, foi possível escrever um código simples, demonstrando a capacidade do sensor de realizar leituras periódicas consistentes das variáveis de presença de

materiais particulados que seriam interessantes para o projeto, sendo que estas foram de MP₁₀, MP_{2,5} e MP_{1,0}, sendo cada uma dadas em $\mu\text{g}/\text{m}^3$. O *script* criado para realizar essa tarefa está disponível no *Apêndice D*. Este algoritmo realiza uma breve exibição dos resultados das leituras do sensor foi programada para ser apresentada utilizando a interface UART conectada à porta USB da placa de desenvolvimento da Heltec. Essa exibição é demonstrada na Figura 33.

Figura 33 – Monitoramento serial das leituras realizada no ensaio com o PMS5003



The screenshot shows a terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, displaying the following text:

```
--- Terminal on COM6 | 9600 8-N-1
{ "pm1_0": 6, "pm2_5": 8, "pm10_0": 9 }
{ "pm1_0": 6, "pm2_5": 8, "pm10_0": 9 }
{ "pm1_0": 5, "pm2_5": 7, "pm10_0": 8 }
{ "pm1_0": 5, "pm2_5": 7, "pm10_0": 8 }
{ "pm1_0": 6, "pm2_5": 8, "pm10_0": 11 }
{ "pm1_0": 6, "pm2_5": 8, "pm10_0": 9 }
{ "pm1_0": 6, "pm2_5": 8, "pm10_0": 9 }
{ "pm1_0": 6, "pm2_5": 8, "pm10_0": 9 }
{ "pm1_0": 6, "pm2_5": 8, "pm10_0": 12 }
{ "pm1_0": 6, "pm2_5": 8, "pm10_0": 12 }
{ "pm1_0": 6, "pm2_5": 8, "pm10_0": 12 }
{ "pm1_0": 6, "pm2_5": 8, "pm10_0": 11 }
{ "pm1_0": 6, "pm2_5": 13, "pm10_0": 15 }
{ "pm1_0": 6, "pm2_5": 13, "pm10_0": 15 }
```

Fonte: Autoria própria (2023)

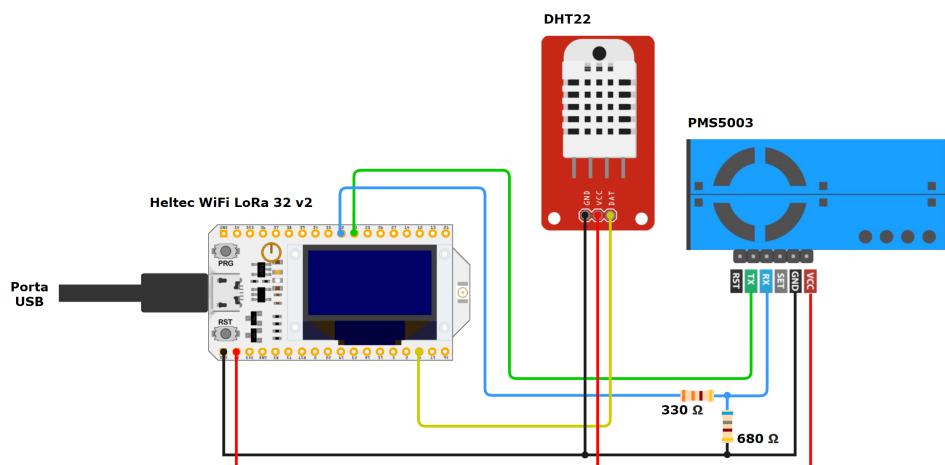
4.1.3 Ensaio Unificado dos Sensores

Através dos ensaios realizados para os testes de leitura utilizando os sensores DHT22 e PMS5003, pôde-se ter uma ideia mais ampla acerca das capacidades no funcionamento desses dispositivos de forma individual. Esse conhecimento acabou permitindo que houvesse uma melhor noção sobre como os dados provenientes dessas leituras poderiam ser reunidas em uma única captura de dados que seriam transmitidos utilizando o protocolo LoRaWAN, através de uma biblioteca LMIC.

Os ensaios individuais também foram importantes para que se tivesse certeza de que o funcionamento em conjunto dos dois sensores não utilizaria recursos limitados de forma sobreposta, garantindo que a construção de um dispositivo periférico LoRaWAN não incorresse em conflitos no acesso aos meios disponibilizados pela placa Heltec WiFi LoRa 32 v2. Assim, para confirmar de forma prática a viabilidade de reunir as leituras dos dois sensores ao mesmo tempo, criou-se um terceiro ensaio. Desta vez, com o objetivo de efetivar leituras a partir dos dois sensores de forma a coletar os dados necessários para o projeto e apresentá-los por um único meio, de forma unificada.

De forma geral, as mesmas decisões tomadas nos ensaios anteriores foram trazidas para este ensaio geral, sendo que a alimentação do Heltec WiFi LoRa 32 v2 foi feita utilizando um computador pessoal, sendo o a alimentação de 5 volts estendida aos dois sensores a partir dos pinos VCC e GND da placa de desenvolvimento. Enquanto o pino 4 foi utilizado para a realização da interface com o DHT22 por meio do protocolo *Single-Bus Communication*, os pinos 32 e 33 designados para operarem as linhas TX e RX da *UART2* da placa da Heltec para que fosse efetivada a comunicação serial com o PMS5003. Vale lembrar que o mesmo dimensionamento de divisor de tensão foi utilizado nessa interface, impedindo que o contato RX do sensor de presença de materiais particulados não tivesse contato direto com 5 volts e sim 3,3 volts. O diagrama dessas escolhas de conexões é demonstrada na Figura 34.

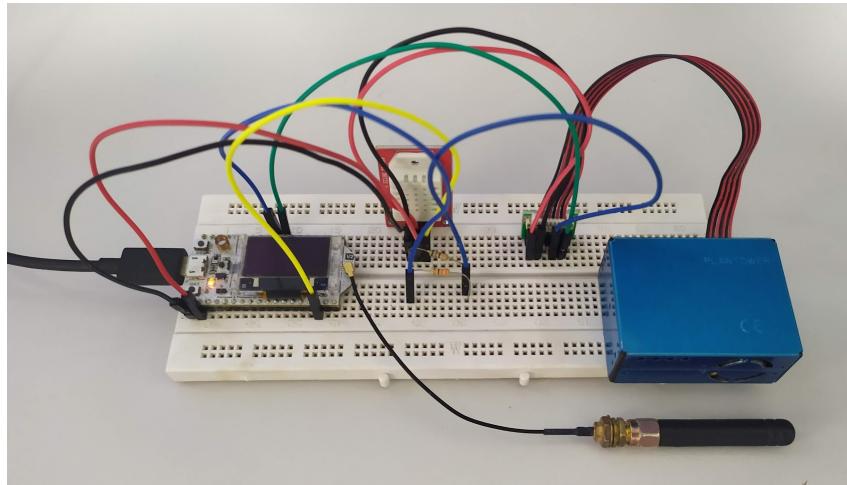
Figura 34 – Montagem do protótipo de hardware



Fonte: Autoria própria (2023)

Uma foto da montagem real do ensaio geral também foi registrada pelo autor. Na imagem, as cores dos fios correspondem às cores apresentadas no diagrama da Figura 34. O registro do protótipo em *protoboard* é mostrado na Figura 35.

Figura 35 – Montagem real do ensaio de integração simultânea do DHT22 e PMS5003



Fonte: Autoria própria (2023)

Para este terceiro ensaio, também foi criado um novo projeto PlatformIO, no qual foram utilizadas as funcionalidades da plataforma para configurar o projeto e instalar todas as bibliotecas que foram necessárias nos ensaios anteriores. Assim, enquanto a biblioteca *DHT Sensor Library* foi instalada para a manipulação do sensor DHT22, as bibliotecas *Adafruit PM25 AQI Sensor*, *Adafruit Unified Sensor* e *PMS Library* foram instaladas para a manipulação do PMS5003. Também foi necessário instalar a *Heltec ESP32 Dev Boards*, pelas mesmas razões justificadas nos ensaios anteriores. Após essas instalações, o arquivo de configurações *platformio.ini* obteve a configuração mostrada na Figura 36 para o projeto criado para este ensaio. O arquivo contendo este código está disponível no *Apêndice E*.

Figura 36 – Arquivo *platformio.ini* para ensaio unificado

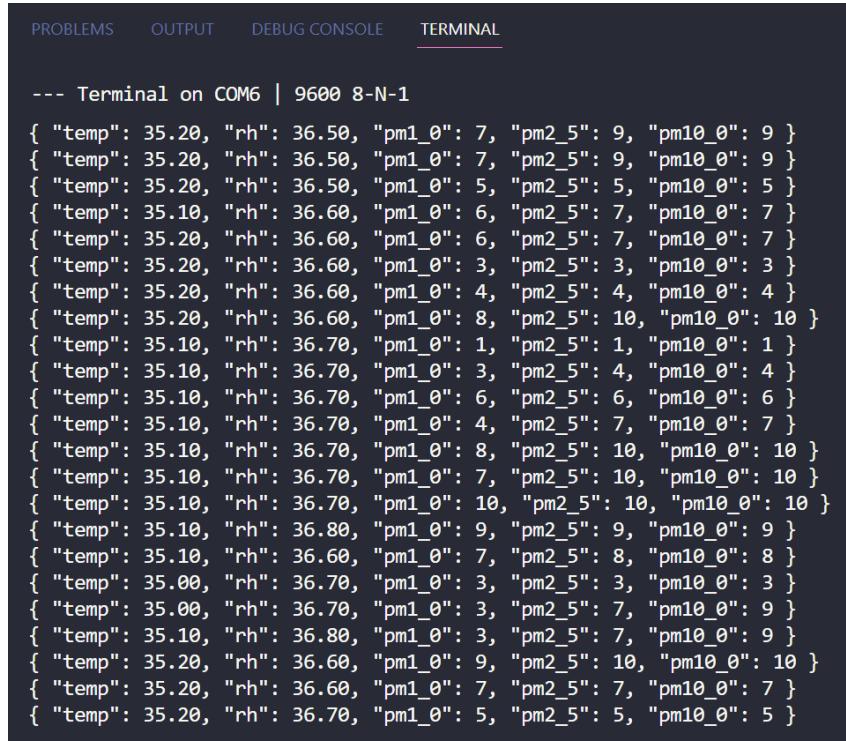
```
[env:heltec_wifi_lora_32_V2]
platform = espressif32
board = heltec_wifi_lora_32_V2
framework = arduino
lib_deps =
    heltecautomation/Heltec ESP32 Dev-Boards@^1.1.1
    adafruit/DHT sensor library@^1.4.4
    adafruit/Adafruit PM25 AQI Sensor@^1.0.6
    adafruit/Adafruit Unified Sensor@^1.1.7
    fu-hsi/PMS Library@^1.1.0
```

Fonte: Autoria própria (2023)

A partir deste ponto, foi possível escrever um código simples, demonstrando a capacidade do sensor de realizar leituras periódicas intercaladas e das variáveis de temperatura (em Celcius), umidade relativa do ar (em porcentagem) e presença de materiais particulados MP₁₀,

$\text{MP}_{2,5}$ e $\text{MP}_{1,0}$, sendo estes valores dados em $\mu\text{g}/\text{m}^3$. O *script* criado para realizar essa tarefa está disponível no *Apêndice F* deste trabalho. Este algoritmo realiza uma exibição simplificada dos resultados das leituras dos sensores e foi programada para ser apresentada utilizando a interface UART conectada à porta USB da placa de desenvolvimento da Heltec. Essa exibição é demonstrada na Figura 37.

Figura 37 – Monitoramento serial das leituras realizada no ensaio unificado



```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

--- Terminal on COM6 | 9600 8-N-1
{
  "temp": 35.20, "rh": 36.50, "pm1_0": 7, "pm2_5": 9, "pm10_0": 9
}
{
  "temp": 35.20, "rh": 36.50, "pm1_0": 7, "pm2_5": 9, "pm10_0": 9
}
{
  "temp": 35.20, "rh": 36.50, "pm1_0": 5, "pm2_5": 5, "pm10_0": 5
}
{
  "temp": 35.10, "rh": 36.60, "pm1_0": 6, "pm2_5": 7, "pm10_0": 7
}
{
  "temp": 35.20, "rh": 36.60, "pm1_0": 6, "pm2_5": 7, "pm10_0": 7
}
{
  "temp": 35.20, "rh": 36.60, "pm1_0": 3, "pm2_5": 3, "pm10_0": 3
}
{
  "temp": 35.20, "rh": 36.60, "pm1_0": 4, "pm2_5": 4, "pm10_0": 4
}
{
  "temp": 35.20, "rh": 36.60, "pm1_0": 8, "pm2_5": 10, "pm10_0": 10
}
{
  "temp": 35.10, "rh": 36.70, "pm1_0": 1, "pm2_5": 1, "pm10_0": 1
}
{
  "temp": 35.10, "rh": 36.70, "pm1_0": 3, "pm2_5": 4, "pm10_0": 4
}
{
  "temp": 35.10, "rh": 36.70, "pm1_0": 6, "pm2_5": 6, "pm10_0": 6
}
{
  "temp": 35.10, "rh": 36.70, "pm1_0": 4, "pm2_5": 7, "pm10_0": 7
}
{
  "temp": 35.10, "rh": 36.70, "pm1_0": 8, "pm2_5": 10, "pm10_0": 10
}
{
  "temp": 35.10, "rh": 36.70, "pm1_0": 7, "pm2_5": 10, "pm10_0": 10
}
{
  "temp": 35.10, "rh": 36.70, "pm1_0": 10, "pm2_5": 10, "pm10_0": 10
}
{
  "temp": 35.10, "rh": 36.80, "pm1_0": 9, "pm2_5": 9, "pm10_0": 9
}
{
  "temp": 35.10, "rh": 36.60, "pm1_0": 7, "pm2_5": 8, "pm10_0": 8
}
{
  "temp": 35.00, "rh": 36.70, "pm1_0": 3, "pm2_5": 3, "pm10_0": 3
}
{
  "temp": 35.00, "rh": 36.70, "pm1_0": 3, "pm2_5": 7, "pm10_0": 9
}
{
  "temp": 35.10, "rh": 36.80, "pm1_0": 3, "pm2_5": 7, "pm10_0": 9
}
{
  "temp": 35.20, "rh": 36.60, "pm1_0": 9, "pm2_5": 10, "pm10_0": 10
}
{
  "temp": 35.20, "rh": 36.60, "pm1_0": 7, "pm2_5": 7, "pm10_0": 7
}
{
  "temp": 35.20, "rh": 36.70, "pm1_0": 5, "pm2_5": 5, "pm10_0": 5
}

```

Fonte: Autoria própria (2023)

4.1.4 Prototipação do *End-Device LoRaWAN* com o *lnlp/LMIC-node*

A partir do ensaio unificado dos sensores, o próximo passo para a implementação de um *end device* LoRaWAN seria o envio dos dados capturados em amostras para um *gateway*, utilizando uma biblioteca que conseguisse implementar as definições LMIC. No caso deste projeto, onde se tem utilizado a *framework* Arduino para todos os ensaios de *hardware* realizados, foi deduzida a necessidade de se buscar por uma implementação de biblioteca LMIC baseada no Arduino, como é o caso das versões publicadas pela MCCI, conforme explicado na Seção 3.2.2 deste trabalho.

Um projeto *open source* denominado *lnlp/LMIC-node* (PARENTE, 2022) se mostrou ser uma alternativa viável para emprego neste trabalho por simplificar a configuração e uso das funcionalidades da pilha Arduino LMIC para múltiplos dispositivos. A ideia geral em torno deste projeto se baseia na disponibilização de um repositório *template* que facilita a implementação de *end devices* LoRaWAN, considerando opções de placas de desenvolvimento e microcontroladores comuns no mercado (dentre eles a própria placa *Heltec WiFi LoRa 32 v2*, utilizada neste projeto). Para isso, a *lnlp/LMIC-node* também adapta a opção pelo uso de uma biblioteca que implementa as definições do LMIC no padrão clássico IBM ou por uma outra biblioteca baseada na *framework* Arduino, seguindo o padrão MCCI (PARENTE, 2022).

A forma de distribuição original do *lnlp/LMIC-node* é encontrada no GitHub e seu uso é guiado por instruções apresentadas pela documentação do repositório em (PARENTE, 2022). O projeto, em seu estado original, possui a forma de um repositório PlatformIO, sendo que a estrutura do diretório criada acaba sendo reconhecida por ter traços característicos de um projeto criado utilizando a plataforma. Um desses traços acaba sendo a presença do arquivo *platformio.ini*, que é peça central de configuração para o utilização do *template* que o repositório dispõe para ajudar na implementação de *end devices* LoRaWAN em projetos customizados, derivados do projeto original.

Considerando as possibilidades que o *template* disponibilizado pelo *lnlp/LMIC-node*, os passos para a customização do mesmo foram de clonar o projeto, configurá-lo e realizar as intervenções de código necessária para que o mesmo pudesse incorporar as leituras dos sensores deste trabalho na forma de um *end device* LoRaWAN. Nas seções subsequentes, serão descritas as principais ações e decisões que foram tomadas na utilização do *lnlp/LMIC-node* para que fosse efetuada a adaptação do *template* original ao contexto deste trabalho.

4.1.4.1 Definição de Dispositivo e Modo de Operação

Após o *template* original do *lnlp/LMIC-node* do GitHub ser clonado, no arquivo *platformio.ini* do projeto encontram-se várias possibilidades de configuração baseadas em uma variedade de dispositivos IoT no mercado suportada pela iniciativa. Nesse arquivo, por padrão, tem-se uma série de grupos de configurações, alguns deles sendo essenciais a qualquer forma de utilização do *template* e outros desses grupos seriam dispensáveis, dependendo do

dispositivos a serem programado ou configurações utilizadas. Assim, a preocupação inicial foi a de realizar essas configurações de forma satisfatória para que o projeto customizado pudesse ser compilado em um binário e carregado na placa *Heltec WiFi LoRa 32 v2*.

A primeira modificação realizada à partir do *template* original foi definir a placa de desenvolvimento que seria utilizada no projeto customizado derivado. Nesse sentido, o arquivo *platformio.ini* do *template* original apresenta o grupo de configurações *[platformio]*, onde foi possível escolher como *default_env* o identificador da placa da Heltec utilizada, que é apresentado por *heltec_wifi_lora_32_v2* na documentação (PARENTE, 2022). A presença desse identificador sinaliza para o PlatformIO qual dos grupos de configurações relativas à placas específicas (iniciados com *env*) deve ser levado em consideração. Neste caso, o grupo de configurações *[env:heltec_wifi_lora_32_v2]* é selecionado, onde podem ser configuradas suas especificidades no contexto deste trabalho.

Figura 38 – Grupo de configurações *[platformio]* na customização do *end device*

```
[platformio]
default_envs =
    heltec_wifi_lora_32_v2
```

Fonte: Autoria própria (2023)

Com essa definição feita no arquivo *platformio.ini*, foi notada a possibilidade de remoção de todos os outros grupos de configurações relativos às placas que não seriam utilizadas. Por consequência, uma vez que cada um desses grupos de configurações faz a adaptação de um arquivo *header* presente na pasta *src/boards* referente à placa do grupo (PARENTE, 2022), percebeu-se também que poderia se realizar a remoção de todos os arquivos deste diretório, com exceção de *bsf_heltec_wifi_lora_32_v2.h*.

Uma outra configuração essencial feita foi a de escolher o modo de funcionamento do *end device*. As opções ABP ou OTAA são suportadas pelo *lnlp/LMIC-node*, sendo que a escolha por uma dessas opções pode ser feita através inclusão ou não da *build flag* -D *ABP_ACTIVATION*, no grupo de configurações *[common]* do arquivo *platformio.ini* (PARENTE, 2022). A presença dessa sinalização faz com que o modo escolhido seja o ABP, enquanto que a ausência dela, configura o *end device* no modo OTAA. Neste trabalho, optou-se por utilizar o modo ABP para fins de simplificação, dado o cenário prototipação. Nesse caso, a *build flag* referida é mantida no arquivo de configurações do projeto PlatformIO.

Como exposto na fundamentação teórica, o modo ABP exige pré-configuração de chaves sessão *NwkSKey* e *AppSKey*, assim como o endereço do *end device* na forma do identificador *DevAddr*, para que haja comunicação com o servidor de rede LoRaWAN. De forma coerente com esses conceitos, o *lnlp/LMIC-node* disponibiliza um arquivo no *template*, chamado *keyfiles lorawan-keys.h*, para inclusão dessas chaves e identificadores para o modo ABP. De forma a complementar essa configuração, o servidor de rede LoRaWAN necessita de um cadastro prévio desse dispositivos através dessas chaves e identificadores também

de forma prévia. Logo, a necessidade de preenchimento do *template* para inclusão desses valores teria que ser postergada até o processo de cadastro do *end device* no servidor de rede LoRaWAN utilizado, como veremos em uma seção dedicada a isso.

Figura 39 – Alocação reservada para identificador e chaves de sessão do modo ABP do *end device* no arquivo *keyfiles lorawan-keys.h*

```
// End-device Address (u4_t) in uint32_t format.
// Note: The value must start with 0x
// (current version of TTN Console does not provide this).
#define ABP_DEVADDR 0x00000000

// Network Session Key (u1_t[16]) in msb format
#define ABP_NWKSKEY 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
| | | | | | | |
| | | | | | | |
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

// Application Session K (u1_t[16]) in msb format
#define ABP_APPSKY 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
| | | | | | | |
| | | | | | | |
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

Fonte: Autoria própria (2023)

Além disso, no grupo *[common]* também é necessária a inclusão da *build_flag* -D DO_WORK_INT ERVAL_SECONDS, que recebe um valor inteiro relativo à quantidade de segundos de espera entre uma iteração do *template* e outra. Este ponto ficará mais claro à frente, onde essas haverão intervenções de código relativas a essas iterações. No mesmo grupo de configurações ainda são incluídos na lista *lib_deps* os identificadores de bibliotecas que serão utilizadas pelo PlatformIO, independente do dispositivo escolhido. Nesses moldes, o *template* inclui as dependências *olikraus/U8g2* e *lnlp/EasyLed*. Isso é feito através da herança dessas configurações que o grupo de configurações relativo ao dispositivo utilizado faz, invocando variáveis de *[common]*.

Figura 40 – Grupo de configurações *[common]* e *[esp32]* na customização do *end device*

```
[common]
monitor_speed = 115200
build_flags =
    -D ABP_ACTIVATION
    -D DO_WORK_INTERVAL_SECONDS=1800
lib_deps =
    olikraus/U8g2
    lnlp/EasyLed

[esp32]
build_flags =
    -D hal_init=LMICHAL_init
```

Fonte: Autoria própria (2023)

4.1.4.2 Definições Relacionadas ao Suporte LMIC

O *template* original do *lnlp/LMIC-node* (PARENTE, 2022) disponibiliza duas opções de uso de bibliotecas LMIC para uso. Uma delas é baseada em uma implementação baseada nas definições clássicas da IBM, mas adaptadas para a *framework* Arduino, sendo esta opção disponibilizada pelo repositório *matthijskooijman/arduino-lmic* no GitHub (KOOIJMAN, 2020). No entanto, essa opção, assim como o suporte IBM para sua padronização LMIC, foi descontinuada e não garante mais suporte. A outra opção disponibilizada apresentada pelo *template* tem como base um repositório do GitHub denominado *mcci-catena/arduino-lorawan* (MCCI CATENA, 2022), que segue as definições LMIC da MCCI e que será a opção empregada neste trabalho.

Nesse contexto, é possível encontrar no arquivo *platformio.ini* do *lnlp/LMIC-node* dois grupos de configuração que são relativos às duas opções de uso de bibliotecas supracitadas, sendo esses grupos nomeados originalmente de *[classic_lmic]* e *[mcci_lmic]*, respectivamente. A opção por uma opção de LMIC ou outra no *template* acaba sendo dependente do dispositivo escolhido, sendo que a placa da Heltec utilizada neste trabalho opta automaticamente pela distribuição baseada nas definições LMIC. Neste caso o grupo de configurações *[classic_lmic]* pode ser removido, a fim de simplificação nos arquivos do projeto.

No grupo *[mcci_lmic]*, que permanece na customização, foram realizadas algumas configurações que acabam tendo grande importância na forma de uso do dispositivo. Apesar de a variável *lib_deps* ser fixa, apontando para a versão LMIC, a variável *build_flags* traz definições que determinam:

- **Plano de Frequência US915:** Dentre as *build_flags* desse grupo, uma delas define o plano de frequência a ser utilizado pelo dispositivo criado. Um desses planos deve ser necessariamente selecionado. Neste sentido, é necessário optar por um plano de frequências que o rádio LoRa utilizado pelo dispositivo alcança. No caso deste trabalho, a placa da Heltec utilizada operaria em frequências entre 868 e 915 MHz (HELTEC, 2018), que conseguiria suportar planos como o EU868, US915 ou AU915, por exemplo. Neste trabalho, optou-se por utilizar US915 pela proximidade com a alocação para a classificação de dispositivos radiação restrita definida pela ANATEL, sendo suficiente pra isso a presença da *build_flags* -D CFG_us915=1 (PARENTE, 2022).
- **Dispositivo da Classe A:** As definições LMIC dadas pela MCCI especificam que as implementações do padrão teriam suporte apenas às classes A e B (MCCI CORPORATION, 2018). Neste trabalho, a opção pela classe A já se faz suficiente, dada a ideia de protótipo aqui concebida. Neste sentido, o grupo de configurações *[mcci_lmic]*, no arquivo *platformio.ini* foi customizada, de modo que a presença das *build_flags* -D DISABLE_PING e -D DISABLE_BEACONS acaba fazendo atribuindo ao *end device* criado a operação na classe A.

- **SX1276 como Rádio Utilizado:** O suporte aos rádios SX1272 e SX1276 são disponíveis quando se utiliza o LMIC especificado pela MCCI, sendo necessário o entendimento de qual desses rádios está sendo utilizado para que o código customizado consiga configurar o *end device* criado de forma correta. A placa de desenvolvimento da Heltec utilizada neste trabalho utilizar o SX1276 (HELTEC, 2018), sendo que a *build_flag* -D CFG_sx1276_radio=1 foi inclusa para efetivar essa sinalização.
- **Modo de Debug e Uso do AES Original:** No referido grupo de configurações também foram inseridas *build_flags* -D USE_ORIGINAL_AES e -D LMIC_DEBUG_LEVEL=0. Enquanto a primeira define a forma de uso da AES (*Advanced Encryption Standard*) (MCCI CORPORATION, 2018), a segunda omite detalhes do aparato de detalhes do *template* disponibilizado pelo *lnlp/LMIC-node* (MCCI CORPORATION, 2018).

A Figura 41 apresenta a forma final do grupo de configurações [*mcci_lmic*] para as configurações que foram descritas.

Figura 41 – Grupo de configurações [*mcci_lmic*] na customização do *end device*

```
[mcci_lmic]
lib_deps =
    mcci-catena/MCCI LoRaWAN LMIC library
build_flags =
    -D CFG_us915=1
    -D DISABLE_PING
    -D DISABLE_BEACONS
    -D CFG_sx1276_radio=1
    -D USE_ORIGINAL_AES
    -D LMIC_DEBUG_LEVEL=0
```

Fonte: Autoria própria (2023)

4.1.4.3 Utilização do Dispositivo e Instalação das Dependências

Diante da necessidade de efetuar a manipulação dos sensores DHT22 e PMS5003, o processo de instalação utilizado nos ensaios foi feito utilizando o console do PlatformIO. Foram instaladas no projeto *lnlp/LMIC-node* clonado todas as dependências que foram necessárias para realizar o ensaio geral envolvendo os dois sensores operando em conjunto, com exceção de *Heltec ESP32 Dev Boards*, que tem sua função já incorporada nas configurações do *lnlp/LMIC-node* para a placa de desenvolvimento da Heltec utilizada.

Foram instaladas, a *DHT Sensor Library* para a manipulação do DHT22, assim como as bibliotecas *Adafruit PM25 AQI Sensor*, *Adafruit Unified Sensor* e *PMS Library* para a manipulação do PMS5003. Após a instalação, o arquivo *platformio.ini* exibe nos grupo de configurações relativos ao dispositivo utilizado, a lista dessas dependências instaladas como atribuição para a variável *lib_deps*. No caso desta customização, essa variável é alocada no

grupo `[env:heltec_wifi_lora_32_v2]`. Essa utilização herda as bibliotecas definidas em outros grupos de configuração comuns, como `[common]` e `[mcci_lmic]`.

Como configurações adicionais, o grupo de configurações relativo ao dispositivo utilizado também de `build_flags`. Algumas destas também são herdadas dos grupos `[common]` e `[mcci_lmic]`, sendo a herança do grupo `[esp32]` a novidade neste sentido. Essa herança basicamente acrescenta a `build_flag -D hal_init=LMICHAL_init`, que especifica a utilização de um função de chamada para HAL específica para dispositivos baseados no microcontrolador ESP32. Além disso outras `build_flag` foram especificadas como forma de definir o arquivo que especifica definições de pinos para a placa de desenvolvimento da Heltec utilizada e questões relacionadas a formas de `debug` utilizando monitor serial e LED na placa de desenvolvimento, por exemplo. A Figura 42 apresenta a forma final do grupo de configurações `[env:heltec_wifi_lora_32_v2]` para as configurações que foram descritas.

Figura 42 – Grupo de configurações `[env:heltec_wifi_lora_32_v2]` na customização do *end device*

```
[env:heltec_wifi_lora_32_v2]
platform = espressif32
board = heltec_wifi_lora_32_V2
framework = arduino
upload_speed = 921600
monitor_speed = ${common.monitor_speed}
lib_deps =
    ${common.lib_deps}
    ${mcci_lmic.lib_deps}
    adafruit/DHT sensor library@^1.4.4
    adafruit/Adafruit Unified Sensor@^1.1.7
    adafruit/Adafruit PM25 AQI Sensor@^1.0.6
    fu-hsi/PMS Library@^1.1.0
build_flags =
    ${common.build_flags}
    ${mcci_lmic.build_flags}
    ${esp32.build_flags}
    -D BSFILE=\"boards/bsf_heltec_wifi_lora_32_v2.h\"
    -D MONITOR_SPEED=${common.monitor_speed}
    -D LMIC_PRINTF_TO=Serial
    -D USE_SERIAL
    -D USE_LED
```

Fonte: Autoria própria (2023)

Todos os grupos de configurações do arquivo de configuração `platformio.ini` na customização do *end device* estão disponíveis no *Apêndice G* deste trabalho.

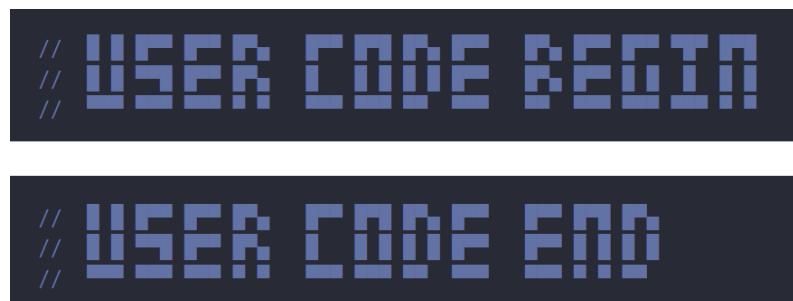
4.1.4.4 Intervenção de Código para Leitura dos Sensores

O projeto *lnlp/LMIC-node* apresenta uma estrutura padrão onde o código do usuário que clonou o projeto deve ser inserido para que a customização funcione para os propósitos do *end device* LoRaWAN criado. Neste sentido, o arquivo `src/LMIC-node.cpp` expõe três seções

onde isso deve ocorrer por padrão. Serão aqui descritas a função dessas seções e o uso delas feitas para a customização no contexto deste projeto.

Na sua forma original, o arquivo *src/LMIC-node.cpp* apresenta um ensaio sugestivo onde é feita a demonstração da transmissão um valor incremental utilizando o dispositivo que está sendo configurado. O código do arquivo serve tanto como referência para o aplicação demonstrativa, sendo que a mesma foi utilizada junto à documentação do projeto para melhor conhecimento de zonas de intervenção no código. Essas zonas estão contidas em identificadores na forma de blocos de comentários sugestivos, mostrados na Figura 43.

Figura 43 – Indicadores do arquivo *src/LMIC-node.cpp* para inclusão de código do usuário



Fonte: PARENTE (2022)

A existência desses identificadores comentados em código no arquivo *src/LMIC-node.cpp* original indica os pontos onde poderiam haver essas inclusões de código. As especificações do que foi feito em cada uma delas são descritas nos pontos a seguir.

- **Definições em Nível Global:** A primeira aparição de uma seção para a inclusão de código pelo usuário do código está relacionada à inclusões de bibliotecas e definições globais como declarações de variáveis e instância de objetos que se deseja obter acesso à nível global. Nesta seção foi alocadas essas estruturas criadas no contexto do ensaio unificado realizado na seção anterior, onde inclusões de bibliotecas para manipulação dos sensores DHT22 e PMS5003 foram feitas, bem como a declaração de variáveis e objetos globais para que fosse possível efetivar a leitura por esses sensores. O trecho de código atribuído a esta alocação está disponível no *Apêndice H* deste trabalho.
- **Instruções de Inicialização:** Uma outra alocação de código nessa estrutura está presente para que chamadas e declarações que acontecem uma única vez em determinados momentos. Por outro lado, essa estrutura também entrega as bases para um fluxo de controle utilizando recorrentes chamadas customizadas na forma de *loop*. Esta alocação foi utilizada neste trabalho para um trecho de código responsável por inicializar os sensores e viabilizar o controle das leituras dos sensores acoplados. Este trecho está disponível no *Apêndice I*.

- **Instruções em Loop:** Uma outra alocação de código de usuário permite a inclusão de trecho de código que se necessite chamar de forma recorrente, em um determinado intervalo de tempo fixo. Este intervalo é definido no arquivo *platformio.ini* do projeto, dentro do grupo de configurações [*common*], pela *build_flag* -D DO_WORK_INTERVAL_SECONDS. A esta definição é atribuído um valor inteiro que faz referência à quantidade de segundos entre um *loop* e outro. Esse *loop* foi utilizado como forma de efetivar as leituras dos sensores DHT22 e PMS5003, assim como montar um *payload* de dados e enviá-lo através do protocolo LoRaWAN. Para tal, é importante mencionar que, considerando o template utilizando do projeto *lulp/LMIC-node*, isso deve ser feito de uma função chamada *processWork()*, que em meio a outros tipos de controle, consegue agendar o envio de mensagens de *uplink* através da chamada da função *scheduleUplink()*. O trecho alocado para realizar essa função no contexto desse trabalho se faz disponível no *Apêndice J*.

Deste ponto em diante, tinha-se um dispositivo capaz de atuar como *end device* LoRaWAN. Este dispositivo, configurado no modo ABP, desde que tivesse suas definições incrementadas com as chaves de sessão *NwkSKey* e *AppSKey*, assim como o endereço do *end device* na forma do identificador *DevAddr*, seria capaz de enviar leituras dos seus sensores para um servidor de rede LoRaWAN entre intervalos de tempo fixo. Esta última configuração precisaria ser retomada no arquivo *keyfiles lorawan-keys.h*, como explicado na Seção 4.1.4.1. O projeto PlatformIO completo para a customização do *end device* LoRaWAN está disponível no *Apêndice L*.

4.2 CONFIGURAÇÃO DA REDE NEUTRA

Com o *end device* configurado e pronto para realizar interações utilizando o protocolo LoRaWAN, seria necessário provisionar os demais elementos essenciais para o funcionamento da topologia requerida pelo protocolo. Neste sentido um *gateway* em associação com um *network server* LoRaWAN possibilitariam a integração do *end device* com a aplicação que seria desenvolvida mais à frente. Na seções subsequentes, serão relatadas as configurações desses elementos para atender a construção do protótipo no contexto deste trabalho.

4.2.1 Configuração do *Gateway* Dragino LPS8

O primeiro elemento da rede neutra LoRaWAN aqui considerado é o *gateway*. Como mencionado na seção de materiais e métodos deste trabalho, optou-se por utilizar o Dragino LPS8 (operando em torno de 915 MHz) como dispositivo *gateway*, dentro da topologia básica requerida pelo protocolo LoRaWAN. À partir das definições de fábrica, as configurações desse dispositivo para cumprir os requisitos levantados por este trabalho são pontuadas nesta seção, tendo sido a documentação do dispositivo a principal referência para tal (DRAGINO, 2022).

Ao conectar o Dragino LPS8 à sua fonte de alimentação, é esperado que o dispositivo atue como AP (*Access Point*) e crie uma rede WLAN padrão à sua volta. Para a unidade utilizada, foi possível identificar o SSID do *gateway* como "dragino-211520" através de um computador pessoal. Estabeleceu-se a conexão com esta rede, utilizando o *password* padrão "dragino+dragino", sendo esta senha temporária de substituição recomendada após o primeiro acesso. A interface com o *software* do LPS8 ocorre por meio de um navegador, por meio do acesso ao endereço de IP 10.130.1.1, quando conectado com a WLAN criada. Esse acesso apresenta um modal para login, onde o *username* "root" e o *password* "dragino" devem ser utilizados no primeiro acesso. Assim, é possível o acesso ao menu do Dragino LPS8, onde as configurações do dispositivo ficam disponíveis para modificações.

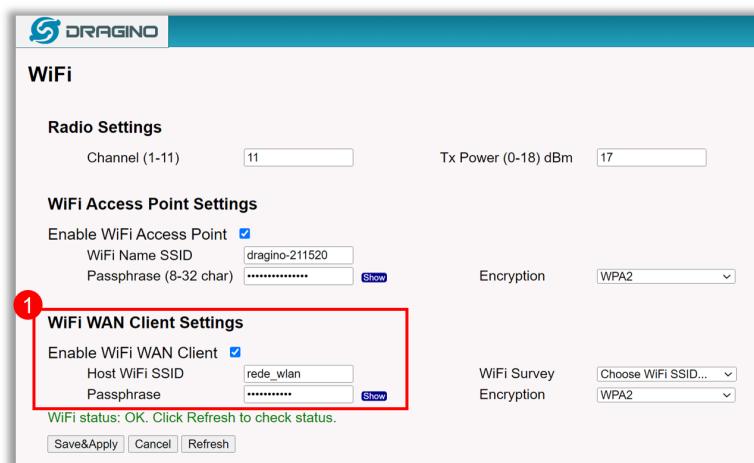
Figura 44 – Menu inicial do Dragino LPS8, acessível através de um navegador



Fonte: Autoria própria (2023)

Partindo para as configurações de WiFi, tem-se a possibilidade de modificar o SSID e *password* padrão utilizado para acessar a rede criada pelo LPS8. Além disso, essa mesma tela de configuração é utilizada para estabelecer a conexão entre o dispositivo *gateway* e a rede WiFi local. Essa configuração efetiva uma relação essencial entre o LPS8 e a Internet, através da qual é possível a realização da ponte de comunicações entre transmissões LoRa e o roteador. Essa ponte é utilizada pelo dispositivo para encaminhar dados recebidos ou destinados ao *network server* LoRaWAN associado através do protocolo TCP/IP. No Dragino LPS8, isso também poderia ser realizado utilizando interface *Ethernet*, através de um cabo RJ-45. A tela das configurações relacionadas ao WiFi é demonstrada na Figura 45.

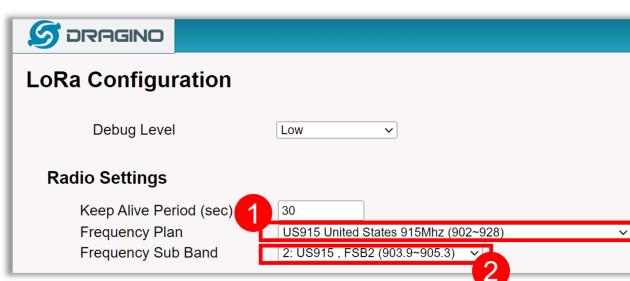
Figura 45 – Opções do Dragino LPS8 relacionadas ao WiFi



Fonte: Autoria própria (2023)

Nas configurações relacionadas à tecnologia LoRa, dentro da zona *Radio Settings*, foram selecionados o plano de frequências US915 (que abrange de 902 a 928 MHz - padrão de uso de utilização do espectro que será utilizado neste projeto) e a sub banda de frequência FSB2 (que utiliza canais entre 903,9 e 905,3 MHz e é sub banda padrão da US915 no TTN). A escolha por essas opções terá impacto nas configurações do servidor de aplicação, como veremos na próxima seção. Um registro do formulário onde essas escolhas feitas é mostrado na Figura 46.

Figura 46 – Opções do Dragino LPS8 relacionadas ao LoRa



Fonte: Autoria própria (2023)

O menu de acesso às configurações do Dragino LPS8 também é utilizado para exibir opções relacionadas ao uso do protocolo LoRaWAN. Na opção de menu principal atribuída ao protocolo, em uma área denominada *Primary LoRaWAN Server* (servidor de rede LoRaWAN primário), pode-se especificar qual o tipo do provedor do serviço do protocolo deve ser esperado como opção primária através do campo "Service Provider". No caso deste projeto, trata-se da opção The Things Network V3, referenciando a mais nova versão do TTN, também conhecida como sendo a distribuição TTS *Community Edition*. Já no campo "Server Address" especifica-se o endereço do servidor de rede LoRaWAN exposto, a qual o LPS8 deve se conectar. No caso deste projeto, opta-se pelo endereço relacionado ao *cluster* de servidores de rede utilizado pelo TTN na América do Norte, comumente referenciado como NAM1 (ou North America 1).

Na mesma tela, uma outra zona bastante importante no contexto das configurações LoRaWAN é apresentada como *General Settings*. Nesta área é possível a verificação de um campo chamado de *Gateway ID*. O valor deste campo é automaticamente gerado pelo dispositivo e o registro dele é importante pois será utilizado para associar a unidade do Dragino LPS8 ao *network server*, sendo isso um ponto-chave nas configurações demonstradas na próxima seção, sobre como preparar o TTN para utilização. As escolhas de configurações relativas ao protocolo LoRaWAN dão demonstradas na Figura 47.

Figura 47 – Opções do Dragino LPS8 relacionadas ao LoRaWAN



Fonte: Autoria própria (2023)

É importante ressaltar que o Dragino LPS8 apresenta possibilidades de configurações que vão além do que foi descrito nesta seção. No entanto, partindo das configurações de fábrica ressaltou-se aqui as modificações necessárias e pontos de atenção importantes para preparar o dispositivo *gateway* para uma operação simplificada no contexto deste trabalho.

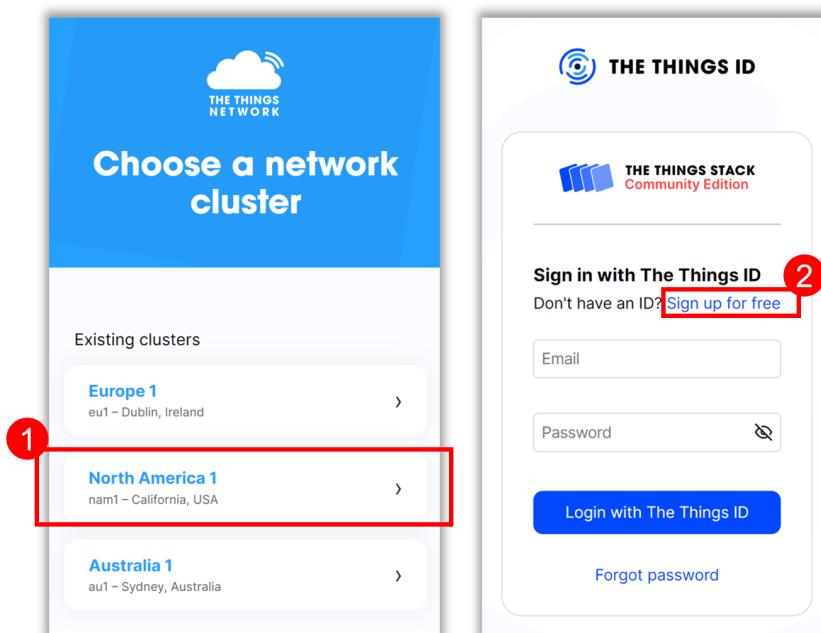
4.2.2 Configuração do Servidor de Rede TTN

Uma vez que se tinha o *gateway LoRaWAN* configurado para realizar sua função dentro da topologia, o próximo passo seria configurar o servidor de rede. Como descrito na Seção 3.3.1 deste trabalho, optou-se aqui por utilizar o serviço TTN, baseado em uma instância do TTS *Community Edition*. Nesta seções subsequentes será relatado como ocorreram as configurações dentro do serviço para que se tivesse as abstrações necessárias registradas na plataforma para cumprir os requisitos de operação deste projeto. Nessas seções também serão demonstradas as escolhas técnicas em torno do uso da plataforma no que diz respeito à forma de tratar os dados em tráfego até que os mesmos pudessem ser disponibilizados para a aplicação que seria criada mais à frente.

4.2.2.1 Criação de Conta e Localização de Serviço do TTN

Um primeiro acesso à *console.cloud.thethings.network*, através de um navegador, direciona o usuário para a escolha de uma das três instâncias de *clusters* baseados no TTS *Community Edition* que são disponibilizadas para acesso pelo console do TTN. É recomendável a utilização da opção mais próxima da região de onde se está desenvolvendo o projeto. No caso deste trabalho, a recomendação de escolha para projetos no Brasil é o North America 1 (ou NAM1). Sendo assim, foi através dessa opção que se criou uma conta do autor utilizada como a base das configurações feitas no TTN. A Figura 48 mostra o partes das telas onde esses procedimentos foram realizados.

Figura 48 – Telas de escolha de cluster e cadastro no TTN



Fonte: Autoria própria (2023)

4.2.2.2 Registro do *Gateway* no TTN

Após a criação de uma conta pessoal no TTN, a primeira ação tomada foi a de efetivar uma associação entre o dispositivo *gateway* e o servidor de rede disponibilizado pelo serviço. Para tal, o console do TTN apresenta em seu menu uma opção chamada *Gateways*, onde a lista de *gateways* registrados e indicações de status de atividades dos mesmos são apresentados. Essa tela também disponibiliza um botão para o registro de um novo dispositivo *gateway*, sendo que este recurso foi utilizado para tal. A Figura 49 mostra parte da referida tela.

Figura 49 – Registro de *gateway* no console do TTN

The screenshot shows the 'Register gateway' interface in The Things Stack Community Edition. At the top, there's a navigation bar with icons for Overview, Applications, and Gateways. The 'Gateways' tab is active. Below the header, the title 'Register gateway' is displayed. A sub-instruction says: 'Register your gateway to enable data traffic between nearby end devices and the network.' It also links to 'Adding Gateways'. The main form contains the following fields:

- Gateway EUI** (Gateway ID): A green button labeled 'Gateway ID' with a 'Reset' button next to it.
- Gateway ID**: An input field containing 'eui-' followed by a placeholder 'Gateway ID'.
- Gateway name**: An input field containing 'sensorar-gateway-1'.
- Frequency plan**: A dropdown menu set to 'United States 902-928 MHz, FSB 2 (used by TTN)'.
- Require authenticated connection**: A checkbox with a note: 'Choose this option eg. if your gateway is powered by LoRa Basic Station'.
- Share gateway information**: A note: 'Select which information can be seen by other network participants, including Packet Broker'.
- Share status within network**: A checked checkbox.
- Share location within network**: A checked checkbox.

At the bottom is a blue 'Register gateway' button.

Fonte: Autoria própria (2023)

Durante registro, o preenchimento de campo denominado *Gateway EUI* é requerido, sendo que o valor inserido nesse campo deve ser o mesmo que o valor hexadecimal presente no campo *Gateway ID* apresentado pela opção de menu LoRaWAN do dispositivo Dragino LPS8. Também é possível atribuir ao dispositivo um identificador paralelo para facilitar o reconhecimento da unidade registrada. Neste trabalho, a unidade do Dragino LPS8 utilizada recebeu como identificador o nome *sensorar-gateway-1*.

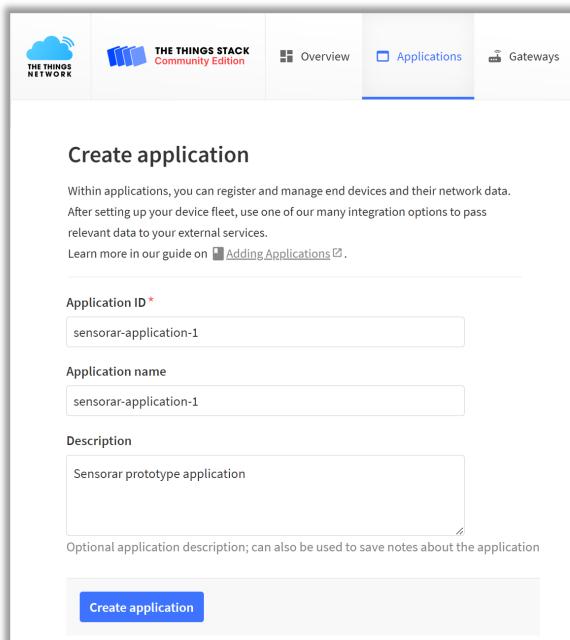
Nessas configurações também houve a necessidade de se manter a coerência em relação às definições de plano de frequência e sub banda de operação definidas no Dragino LPS8. Desta forma, foi necessário que o que as opções por US915 e FSB2 nas configurações do tivesse correspondência direta neste ponto, o que também é mostrado no campo *Frequency plan* da Figura 49.

A inclusão dessa abstração de *gateway* no TTN associa o dispositivo diretamente à região utilizada. Ou seja, uma vez se utilize o console relacionado com o *cluster* NAM1, o *gateway* registrado será vinculado ao servidor relativo a esse servidor. Assim, o dispositivo adicionado possuiria relação com o servidor LoRaWAN exposto pelo endereço *nam1.cloud.thethings.network*, sendo isso coerente com a configuração LoRaWAN feita no Dragino LPS8.

4.2.2.3 Registro de uma Aplicação e *End Devices* Associados

Assim como ocorre com o registro de *gateways*, para a delimitação de representações de aplicações, console do TTN apresenta em seu menu uma opção chamada *Applications*, onde é mostrada uma lista de aplicações registradas, também com indicações sobre status de atividades de cada uma delas. É por meio dessa opção que a plataforma permite ao usuário a criação de novas abstrações de aplicações. A Figura 50 apresenta parte da tela onde isso é feito.

Figura 50 – Criação da abstração de uma aplicação no TTN



Fonte: Autoria própria (2023)

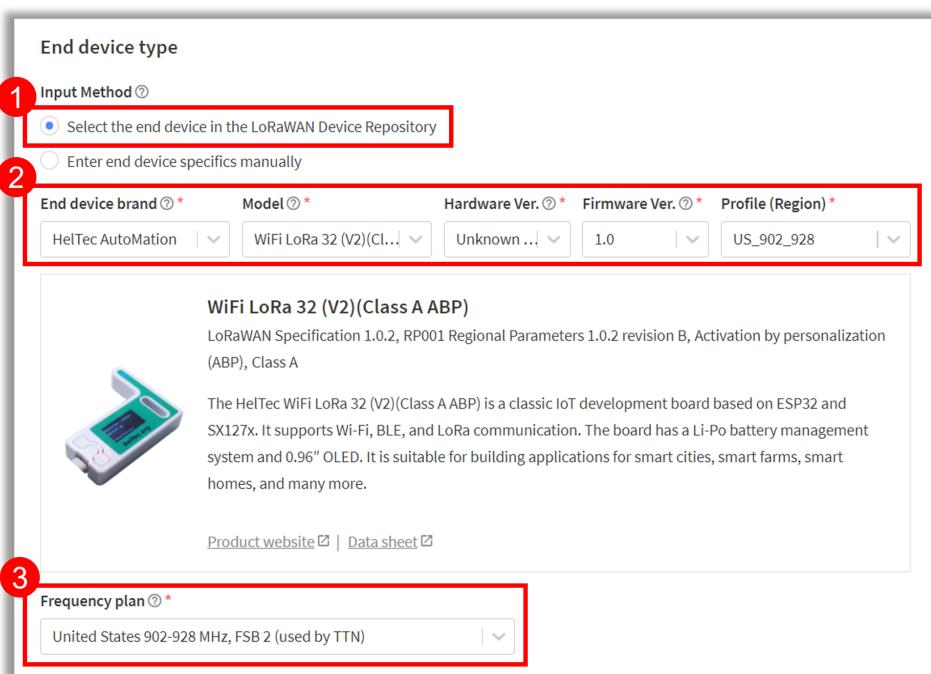
A criação dessas abstrações é bastante simples e direta, sendo apenas necessário apenas a atribuição de um *Application ID* que identifica de forma única a abstração de aplicação dentro da conta utilizada. Também são possíveis atribuições opcionais relacionadas ao nome da aplicação e sua descrição. No caso deste trabalho, criou-se uma aplicação com o identificador *sensorar-application-1*, mantendo o padrão de nomenclatura até aqui empregado.

Além de criação dessas abstrações em si, é também por meio da opção *Applications*, dentro do console do TTN, que a plataforma permite ao usuário fazer associações entre uma aplicação criada e selecionada com os seus *end devices*. Assim, para cada aplicação

criada, é disponsta uma lista onde são exibidos os dispositivos periféricos cadastrados e onde pode-se optar pela opção de cadastrar novos dispositivos. Existe nesta opção a possibilidade de escolha por equipamentos de configuração conhecida pelo TTN ou pela especificação manual de configurações LoRaWAN. Para o caso deste projeto, não foi necessário utilizar a segunda opção pois a placa de desenvolvimento Heltec WiFi LoRa 32 v2 é suportada pela plataforma em diferentes formas de uso.

De forma coerente com as configurações atribuídas ao *end device* criado neste trabalho, foi necessário optar pelas seleções pela classe A, e modo ABP. Também se especificou o uso da versão de *firmware* 1.0, compatível com o TTN V3 e perfil regional de operação entre 902 e 928 MHz. Também foi especificado aqui o plano de frequências US915, utilizando a sub banda FSB 2, mantendo as especificações de hardware e de configurações do equipamento utilizado na composição do *end device*. Um registro das configurações especificadas na opção pelo equipamento é mostrado na Figura 51, como parte da tela de registro de um dispositivo periférico para a aplicação criada.

Figura 51 – Registro do *end device* no console do TTN



Fonte: Autoria própria (2023)

A escolha por essas opções exige que o usuário provisione informações sobre a composição de configurações utilizada. O modo ABP requisitando acaba necessitando de atribuições *DevEUI*, *Device address*, *NwkSKey* e *AppSKey*, enquanto o campo *End device ID* é preenchido automaticamente. Todos esses valores podem ser gerados automaticamente na zona de registro do *end device*, sendo que esse recurso foi utilizado. Mais tarde as configurações do dispositivo foram acessadas para nomear o dispositivo de *sensorar-enddevice-1*. Uma imagem do console do TTN nessas atribuições é mostrada na Figura 52.

Figura 52 – Provisão de identificação e chaves de sessão do modo ABP no console do TTN

The screenshot shows the 'Provisioning information' section of the TTN console. It includes fields for:

- DevEUI**: A 64-bit identifier with a 'Generate' button and a note: "2/50 used".
- Device address**: A 16-bit identifier with a 'Generate' button.
- AppSKey**: An 80-bit session key with a 'Generate' button.
- NwksKey**: A 64-bit network session key with a 'Generate' button.
- End device ID**: A text input field containing "my-new-device". Below it, a note says: "This value is automatically prefilled using the DevEUI".

Fonte: Autoria própria (2023)

Tanto o valor gerado para o campo *Device address*, quanto as chaves de sessão *NwksKey* e *AppSKey* possuem importância fundamental no complemento final da implementação do *end device*. Esses valores foram copiados para as especificações de chaves da customização feita à partir do *template* do *lnlp/LMIC-node*. Como mencionado anteriormente, é atribuído ao arquivo *keyfiles/lorawan-keys.h* a função de especificação da identificação e chaves de sessão do modo ABP. Neste momento, tendo esses valores sido gerados no registro do *end device* no TTN, pôde-se realizar tal preenchimento, sendo que a partir desse momento, tinha-se o dispositivo pronto para usar o protocolo LoRaWAN para transmitir dados até aplicação.

4.2.2.4 Configuração *Payload Formatters*

Uma vez que se havia configurado o *end device* customizado para o envio de um *payload* contendo valores sobre os dados coletados utilizando os sensores, seria necessário a interpretação desses conjuntos de *bytes* para a recuperação desses valores na recepção. Isso poderia ser feito tanto pela própria aplicação que seria criada posteriormente, ou no próprio console do TTN através de um recurso chamado *payload formatters* que a plataforma disponibiliza. Essa funcionalidade permite o uso de *descripts* para realizar transformações complementares em cima dos *payload* trafegados. A utilização de *payload formatters* foi a estratégia utilizada por este projeto para que fosse abstraída a complexidade desta tarefa na perspectiva do que viria a ser a aplicação final.

Através dos *payload formatters* seria possível intervir nos dados brutos transmitidos pelo dispositivos para que os mesmos pudessem ser disponibilizados em formato interpretável pelo servidor de rede de forma variáveis com nomenclaturas de maior significado. Assim como é interessante em contextos de necessidade de decodificação de dados codificados, esse recurso pode ser útil na conversão de unidades, ou adição de informações de metadados.

dos, por exemplo. No escopo do TTN, um *payload formatter* pode utilizado para formatar mensagens de *uplink* ou *downlink*, sendo que a atribuição desses *scripts* pode ser feita para determinados *end devices* registrados ou para todos os dispositivos de uma determinada abstração de abstração.

O *payload formatter* utilizado nesse projeto foi escrito na linguagens de programação JavaScript, sendo definido de forma prévia dentro do console do TTS e atribuído à mensagens de *uplink* destinadas à aplicação *sensorar-application-1* criada. Essa pré-definição buscou intervir em pacotes que utilizarem o valor 10 no campo de *fPort*, já que este foi o valor atribuído a este campo no contexto anterior de envio de dados pelo dispositivo periférico. O código utilizado como *payload formatter* é mostrado na Figura 53.

Figura 53 – Payload Formatter para transformação dos pacotes enviados pelo end device

The screenshot shows the THE THINGS STACK Community Edition interface. In the top navigation bar, 'THE THINGS STACK Community Edition' is selected. Below it, the 'Applications' tab is active, showing 'sensorar-application-1'. Under this, the 'Uplink > Payload formatters' path is visible. The main content area is titled 'Default uplink payload formatter' and has a 'Setup' section. A dropdown menu for 'Formatter type' is open, showing 'Custom Javascript formatter'. The 'Formatter code' section contains the following JavaScript code:

```

1 function decodeUplink(input) {
2     var data = {};
3     var warnings = [];
4
5     if (input.fPort == 10) {
6
7         data.temp = ((input.bytes[0] << 8) + input.bytes[1])
8             + (((input.bytes[2] << 8) + input.bytes[3]) / 100.0);
9
10        data.rh = ((input.bytes[4] << 8) + input.bytes[5])
11            + (((input.bytes[6] << 8) + input.bytes[7]) / 100.0);
12
13        data.pm1_0 = ((input.bytes[8] << 8) | input.bytes[9]);
14
15        data.pm2_5 = ((input.bytes[10] << 8) | input.bytes[11]);
16
17        data.pm10_0 = ((input.bytes[12] << 8) | input.bytes[13]);
18
19    } else {
20
21        warnings.push("Unsupported fPort");
22
23    }
24
25    return {
26        data: data,
27        warnings: warnings
28    };
}

```

A red box encloses the sidebar under 'Payload formatters' (labeled 1) and the code editor area (labeled 2).

Fonte: Autoria própria (2023)

O *script* criado tem como principal objetivo realizar operações de deslocamento de bits para decodificar a sequencia de *bytes* enviada no *payload* e atribuir os valores decodificados à pacotes decodificados que terão utilidade evidente na interpretação dos dados de *uplink*. Assim, os dados capturados conseguem assumir registros associados à chave *decoded_payload* nos registros de atividade de *uplink* da aplicação. Isso terá sua importância na captura dos dados trafegados por sistemas externos utilizando o recurso de *storage integration*, como será demonstrado mais a diante.

4.2.2.5 Teste de Tráfego de Mensagens de *Uplink*

A utilização do TTN como servidor de rede LoRaWAN para atender às necessidades técnicas deste trabalho precisava passar por uma verificação de funcionamento mais direta. Nessa verificação, as implementações, customizações e configurações descritas até aqui deveriam ser testadas em conjunto a fim de demonstrar a efetividade do uso do protocolo LoRaWAN na comunicação entre os elementos da arquitetura desenvolvidos até o momento. A realização desse teste seria possível através da observação de indicações das conexões e trocas de dados entre esses elementos utilizando os próprios recursos disponibilizados pelo console do TTN.

Desde que o dispositivo periférico customizado conseguisse transmitir dados, utilizando o Dragino LPS8 configurado, para a abstração de aplicação criada no TTN, seria possível observar o funcionamento adequado dessas partes. Assim, considerando a conexão do Dragino LPS8 à sua fonte de alimentação e a verificação do seu registro *sensorar-gateway-1* no TTN tendo *status* de "conectado", pôde-se confirmar efetividade desse registro. Essa indicação é mostrada na Figura 54.

Figura 54 – Indicação de conexão do *gateway* registrado no console do TTN

ID	Name	Gateway EUI	Status	Created at
eui-...	Gateway ID	sensorar-gateway-1	Connected	Jan 2, 2023

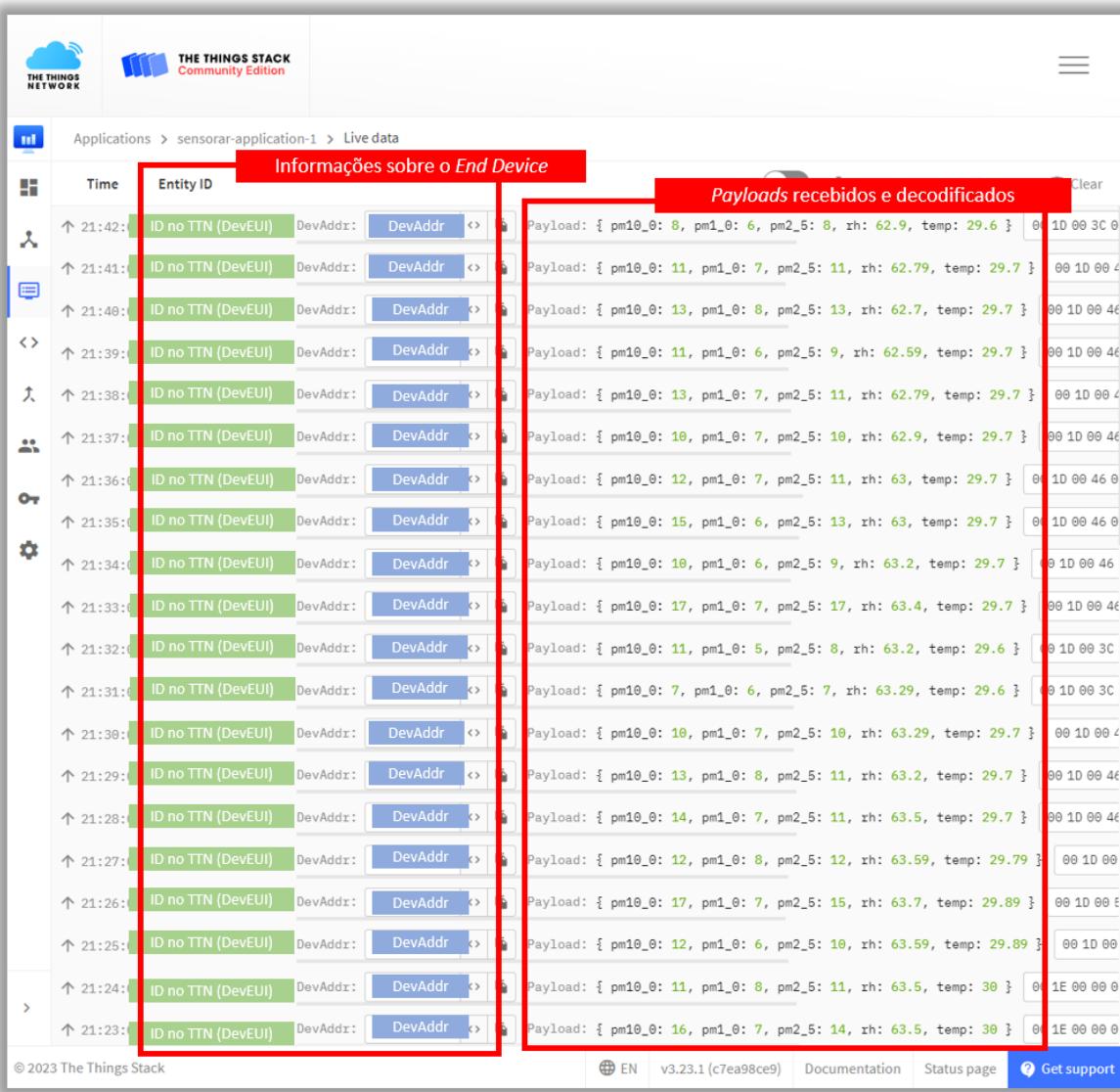
Fonte: Autoria própria (2023)

De forma similar, a criação e configuração da aplicação *sensorar-application-1* e do dispositivo periférico *sensorar-enddevice-1* teriam que sua confirmação verificada de alguma forma. E isso acabou sendo feito também através de recursos disponibilizados pelo próprio console do TTN. Para tal, considerando o funcionamento confirmado do *gateway* junto ao TTN, a foi feita a conexão do *end device* à sua fonte de alimentação para possibilitar que o dispositivo fizesse o trabalho de realizar as leituras dos sensores e envio dos dados capturados para o servidor de rede e a aplicação criada, utilizando o protocolo LoRaWAN.

Com esses elementos em atividade, ao selecionar no console a aplicação *sensorar-application-1* em uma zona chamada *Live data* apresenta as mensagens de *uplink* e *downlink* na troca de dados entre a aplicação e o *end device* nela registrado, sendo este o *sensorar-enddevice-1*. Através dessa opção, foi possível realizar verificações visuais dessas transmissões e confirmar ou não a coerência dessas apresentações de registros com o envio de dados de leituras dos sensores pelo *end device* criado.

Ao fazer isso, foi possível notar o aparecimento de registros relacionados ao mensagens de *upload* que tinham como remetente o dispositivo periférico registrado. Além isso, para cada um desses registros, era possível notar que o *payload formatter* utilizado conseguia decodificar os de forma bastante coerente com a sequência de variáveis de interesse proposta, apresentando as chaves *temp*, *rh*, *pm1_0*, *pm2_5* e *pm10_0*, bem como os valores capturados relativos a cada uma das leituras correspondentes. A representação dessa exibição é mostrada na Figura 55.

Figura 55 – Atividades de *uplink* pelo *sensorar-application-1* no console do TTN



Fonte: Autoria própria (2023)

4.3 APLICAÇÃO DE MONITORAMENTO

Uma vez que se havia verificado o funcionamento dos elementos na composição da rede LoRaWAN criada para este trabalho, o passo seguinte seria a utilização dos dados capturados para o desenvolvimento de uma aplicação externa que pudesse consumir esses registros, e dar forma à aplicabilidade proposta pelo projeto. Se tratando de um sistema de monitoramento da presença de materiais particulados, isso acabou sendo feito por meio da criação de uma aplicação *web* para a apresentação dos valores de IQAr e suas representações nas cores atribuídas pelo CONAMA, relativa aos intervalos do valor adimensional assumido pelo índice. O processo de criação dessa aplicação de visualização de dados é relatado nas seções subsequentes.

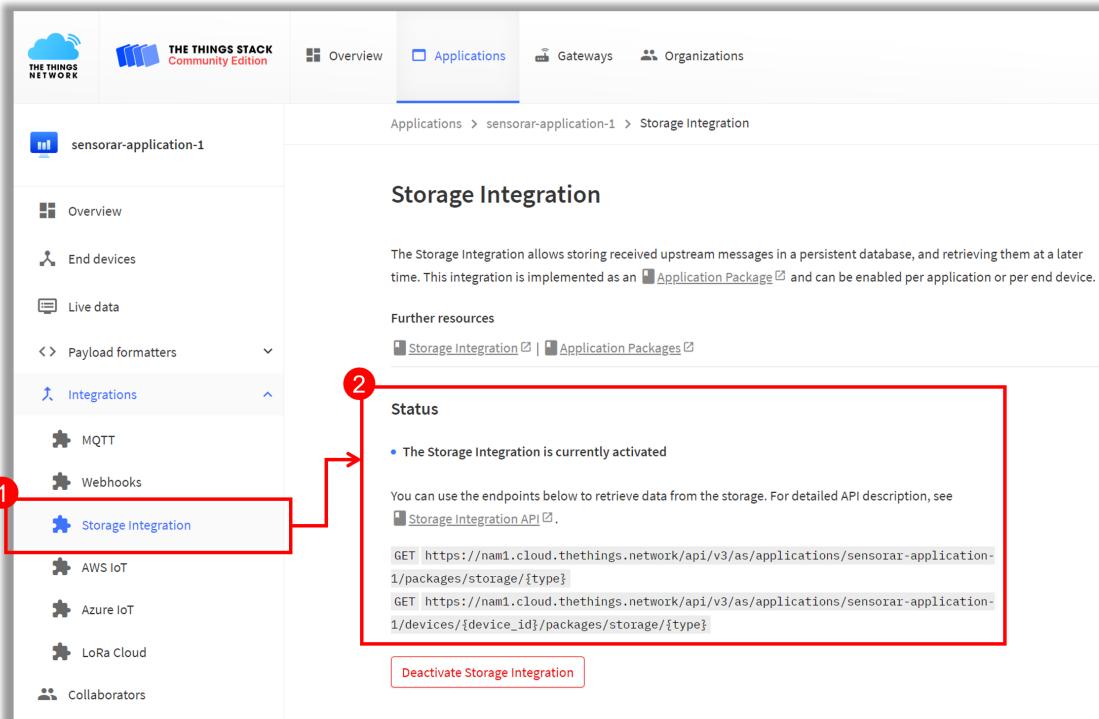
4.3.1 Ingestão de Dados e Persistência

Como dito anteriormente, a utilização do TTN, na forma de instâncias do TTS *Community Edition*, apresenta limitações que impactam em outras decisões técnicas deste trabalho. Uma dessas limitações acaba sendo refletida na volatilidade dos dados que trafegam pela plataforma. Como os dados relativos às leituras dos sensores integrados ao *end device* possuem tempo limitado de retenção no TTN, houve a necessidade de criar mecanismos de ingestão dos dados produzidos pelo *end device* a fim que os mesmos não estivessem submetidos a um tipo de armazenamento volátil.

Como já mencionado, a abordagem deste problema acabou considerando a utilização do recurso chamado de *storage integration*, disponível no console do TTN, como forma de obtenção desses dados por elementos de *software* externos à rede neutra. Como apresentado na Seção 3.3.1, essa opção oferece ao usuário a possibilidade de utilizar chamadas de uma API para acesso aos dados temporariamente armazenados na plataforma, associados a uma aplicação. Assim, ao criar uma aplicação dentro da plataforma TTN, como é o caso do que foi feito para gerar a abstração *sensorar-application-1*, esse recurso pode ser ativado para expor *endpoints* de uma API através da qual requisições HTTP podem ser utilizadas para acesso aos dados de *uplink* produzidos pelo *end device* desenvolvido.

Nesse sentido, seria necessário a montagem de requisições HTTP que faria acessariam e capturariam todas as mensagens de *uplink* armazenadas temporariamente pela aplicação no TTN. A ação de ativação do recurso de *storage integration* é facilmente feito através do próprio menu da abstração e aplicação criada, na opção *Storage Integration*. Isso faz com que o TTN exponha em API os *endpoints* necessários para acesso aos dados de tráfego, podendo isso ser feito por no escopo de todos os *end devices* de uma aplicação, ou no escopo de um *end device* específico. Após a ativação, o *status* de ativação do recurso para *sensorar-application-1* é mostrado na Figura 56.

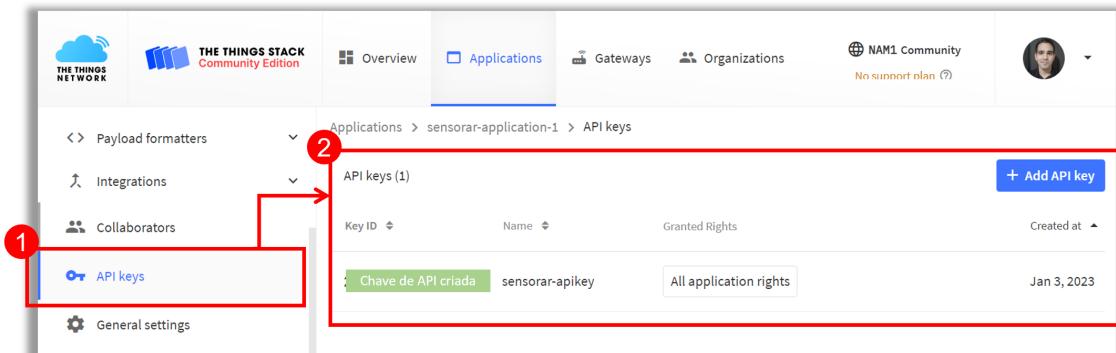
Figura 56 – Caminho para a ativação do recurso de *Storage Integration* no console do TTN



Fonte: Autoria própria (2023)

Para que um cliente HTTP consiga acessar os *enddevices* expostos pelo TTN, é necessário também que se crie uma chave de acesso para a API. Esta chave é introduzida no corpo da requisição HTTP para que se tenha controle sobre quais aplicações externas teriam acesso aos dados trafegados na plataforma, sendo facilmente criada pelo menu da abstração de aplicação criada no TTN, na opção *API Keys*. A Figura 57 mostra onde isso é feito dentro do console do TTN.

Figura 57 – Caminho para a criação de uma chave de API no console do TTN



Fonte: Autoria própria (2023)

Utilizando o acesso a esses *endpoints*, foi possível a obtenção de resposta de requisição acabavam retornando estruturas de dados em formato JSON, contendo detalhes da atividade

de *uplink de sensorar-application-1*. Esses detalhes possibilitaram a criação de uma aplicação de ingestão, capaz de utilizar a biblioteca padrão do Python e o SQLAlchemy para transformar esses registros em objetos armazená-los em uma tabela de banco de dados relacionado SQLite. Os eventos de ingestão ocorrem de forma agendada, podendo acontecer múltiplas vezes ao dia.

Considerando tal cenário, logo ficou claro que a probabilidade de sobreposição de registros de transmissões extraídos seria bastante alta. Caso isso não fosse levado em consideração, isso causaria o preenchimento do banco de dados com múltiplos registros duplicados, o que seria indesejável para este cenário. Para lidar com esse problema, optou-se por criar mecanismo que pudesse utilizar o conteúdo do banco de dados com as extrações realizadas até o momento para indicar quais seriam os novos registros a serem inseridos no banco de dados SQLite, de modo a impedir entradas duplicadas. Isso foi feito com a ajuda de transformações de dados utilizando a biblioteca *Pandas*. Esta implementação se faz disponível no *Apêndice L*.

4.3.2 Apresentação dos Dados em *Dashboard*

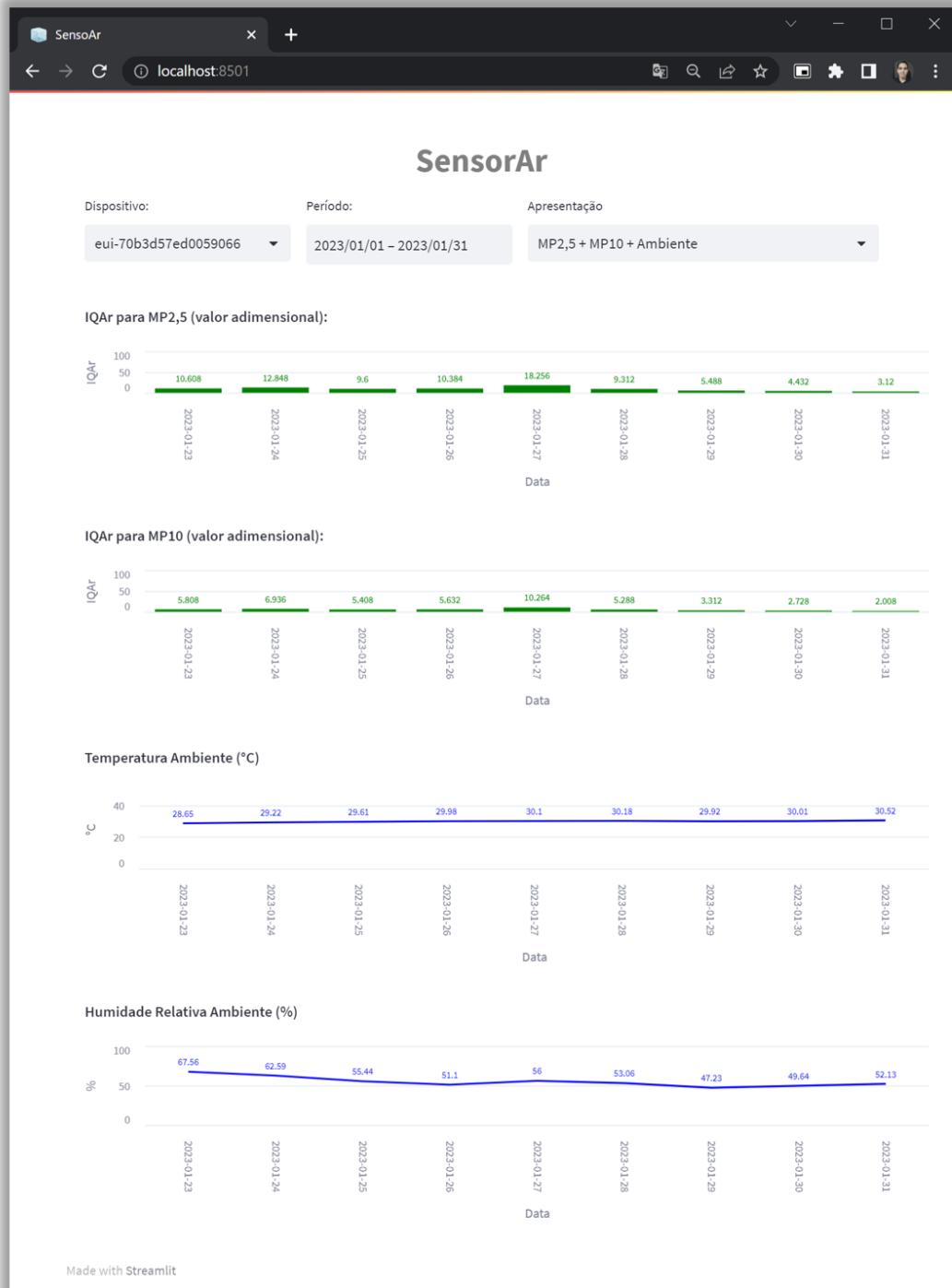
Uma vez que os dados extraídos puderam ser persistidos em uma das tabelas do banco de dados SQLite3, os mesmos puderam ser facilmente carregados em um *dataframe Pandas* para que manipulações fossem aplicadas. Essas manipulações seriam grande importância para a transformação desses registros para formatos mais adequados para a apresentação pela aplicação.

Operações de agregação puderam ser utilizadas para o cálculo de médias diárias para os valores relativos às chaves *temp*, *rh*, *pm1_0*, *pm2_5* e *pm10_0*. Em seguida, operações definidas em funções foram criadas para calcular e adicionar novas colunas relativas ao IQAr. Enquanto uma dessas colunas armazenaria o índice alcançado pelas médias diárias de cada um dos valores de MP₁₀ e MP_{2,5}, uma outra coluna foi gerada para atribuição da cor dada à classificação do índice a cada um das datas. Os cálculos e atribuições feitos nessa etapa se basearam nas definições do IQAr, explicadas na Seção 2.2.2.1 deste trabalho.

A utilização da biblioteca Streamlit permitiu que fosse possível utilizar o *dataframe Pandas* transformado para renderizar elementos visuais em uma aplicação *web* demonstrativa. Nesta demonstração, filtros relacionados à escolha do dispositivo, delimitação do período da amostragem e forma de apresentação dos gráficos são expostos para que o usuário consiga especificar melhor suas análises. A aplicação *web* foi nomeada de *SensoAr* e nela é possível a visualização do IQAr para MP₁₀ e MP_{2,5} em gráficos de barras, cujas cores de cada uma das barras corresponde à cor de classificação dada para o índice alcançado.

Na aplicação também é possível visualizar gráficos de linha correspondentes à temperatura e umidade relativa do ambiente para as datas que foram filtradas. Uma visualização de amostras reais do sistema, realizadas entre os dias 23 e 32 de fevereiro de 2023, são apresentada nas Figura 58.

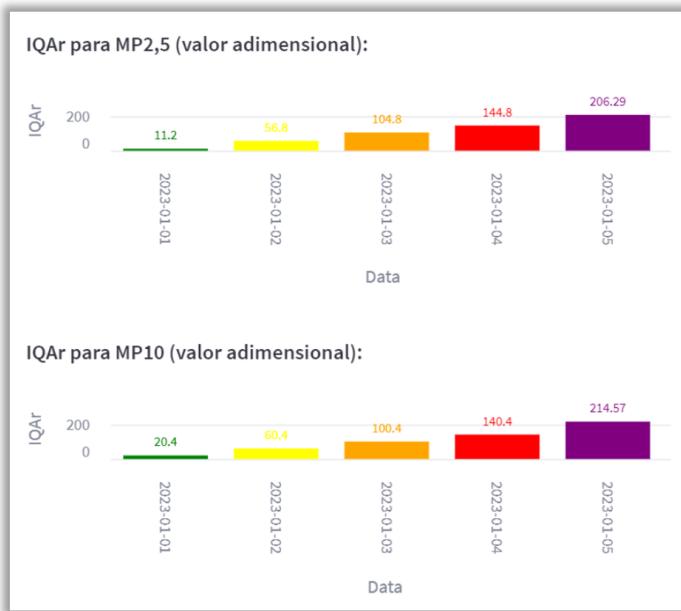
Figura 58 – Aplicação web criada para apresentação dos dados do SensorAr



Fonte: Autoria própria (2023)

Um exemplo fictício da representação do IQAr no gráfico de barras proposto, apresentando valores do índice na faixa de cada uma das possíveis cores da escala, é mostrado na Figura 59.

Figura 59 – Exemplo fictício da representação de barras com os valores do IQAr e representação de cores associadas



Fonte: Autoria própria (2023)

O código em Python para o desenvolvimento da referida aplicação de visualização se encontra disponível no *Apêndice M*.

5 CONSIDERAÇÕES FINAIS

5.1 CONCLUSÃO

Este trabalho considerou a utilização das tecnologias LoRa e LoRaWAN para o desenvolvimento de um sistema IoT capaz servir de base para a tarefa de monitoramento da presença de materiais particulados para áreas impactadas pelo pó de gesso em suspensão, como o Polo Gesseiro do Araripe. Partindo desse princípio, foi fundamentada uma proposta de solução embasada no estudo de conceitos básicos para abordagem do problema e em meios práticos que poderiam ser utilizados para a materialização, em ambiente controlado, de um protótipo para este tipo de monitoramento.

Neste sentido, o conhecimento sobre o que precisaria ser mensurado se apresentou como necessidade essencial, sendo que métricas relacionadas ao diâmetro de materiais particulados em suspensão, como MP₁₀ e MP_{2,5}, foram levadas em consideração no embasamento da proposta de solução, dada sua importância reconhecida globalmente. Em conformidade com essas definições, o IQAr (Índice de Qualidade do Ar), indicado pelo *Guia de Monitoramento de Qualidade do Ar* do CONAMA como referência nacional, foi entendido como forma interessante de promover a democratização do entendimento sobre métricas relacionadas à presença de agentes poluentes no ar atmosférico, incluindo materiais particulados.

Através de ensaios envolvendo tecnologias propostas, foi possível o desenvolvimento protótipos de *hardware* e *firmware* para capturas recorrentes de dados sobre agentes poluentes MP₁₀ e MP_{2,5}, bem como condições pontuais de umidade relativa e temperatura, que seriam importantes no contexto de monitoramento do projeto. Essas implementações e ensaios foram essenciais para verificação da viabilidade técnica da integração dos sensores no posterior desenvolvimento de um dispositivo periférico LoRaWAN capaz de realizar leituras utilizando os mesmos meios.

Considerando o conhecimento adquirido nesses ensaios, foi possível incorporar as funcionalidades de aquisição de dados já testadas a uma implementação de biblioteca LMIC (*LoRaWAN MAC in C*). Isso possibilitou que as leituras realizadas pelos sensores pudessem ser transmitidas por meio de uma rede neutra LoRaWAN formada por um *gateway* e um servidor de rede que tiveram seus processos de configuração expostos neste trabalho. Deste modo, foi possível o desenvolvimento e teste de um protótipo de dispositivo periférico capaz de integrar a uma arquitetura e utilizá-la para trafegar dados até uma abstração de aplicação utilizando o protocolo LoRaWAN.

Considerando limitações relacionadas à volatilidade de armazenamento imposta pelo serviço TTN (*The Things Network*) como servidor de rede LoRaWAN, foi necessário criar um aplicação externa capaz de realizar a ingestão desses dados. Isso foi feito para promover a persistência dos dados produzidos pelo dispositivo periférico LoRaWAN desenvolvido, o que

acabava sendo um requisito essencial para este projeto. Por meio do acesso a esses dados armazenados, foi possível o desenvolvimento de uma exemplo de aplicação na forma de *dashboard* para visualização e análise dos dados coletados pelo sistema de monitoramento. Este sistema é capaz de apresentar, não só médias diárias de temperatura e umidade, como também calcular e exibir o IQAr relativo aos poluentes MP₁₀ e MP_{2,5} para cada data, apresentando-os em gráficos de barras coloridas com as classificações de cores do indicador.

Assim, além de possibilitar o monitoramento dessas variáveis em ambiente controlado, o sistema se mostrou capaz de armazenar esses dados e disponibilizá-lo para que pudessem ser consultados e analisados posteriormente. Desta forma, desde que novas extensões deste trabalho pudesse abordar necessidades práticas sobre a operação fora de ambiente controlado, as considerações e implementações apresentadas neste trabalho poderiam ser utilizadas para o estabelecimento de sistemas para o acompanhamento da exposição do meio ambiente e pessoas aos poluentes monitorados.

5.2 TRABALHOS FUTUROS

Tomando como base este trabalho de conclusão de curso, muitas possibilidades de expansão do estudo de desenvolvimento prático poderiam ser exploradas. Essas potencialidades devem ser consideradas válidas, uma vez que o projeto desenvolvido aqui consegue garantir a viabilidade de utilização dos meios utilizados considerando um cenário restrito, de ambiente controlado.

A expansão para um projeto que possibilitasse o monitoramento de qualidade do ar, com representatividade adequada para fins científicos, exigiria a validação dos meios de captura de dados utilizados. Neste sentido, a comparação dessas capturas com a de meios homologados pelo órgão nacional responsável para tal permitiria a realização de ensaios e captura de dados para a validação desse uso. A possibilidade de implementação de sistemas de calibração poderiam se fazer um recurso interessantes neste tipo de validação.

Além disso, as capacidades de uso do protocolo LoRaWAN se mostrariam um terreno vasto de estudos de desempenho. Neste sentido, seriam interessantes ensaios utilizando diferentes classes de dispositivos, diferentes modos de operação, múltiplos *end devices* ou capacidade de operação de múltiplos *gateways* associados a uma aplicação. Além disso, estudos sobre o funcionamento físico do dispositivos periféricos em ambientes não controlados específicos poderiam embasar diferentes estudos de aplicabilidades. Isso inclui estudos de autonomia de bateria, considerando o uso dos dispositivos aqui propostos e análises de alcance de transmissões em cenários onde este tipo de estudo se mostrasse conveniente.

REFERÊNCIAS

- ANATEL. Resolução Nº 454, de 11 de dezembro de 2006.** 2006. Disponível em: <<https://informacoes.anatel.gov.br/legislacao/resolucoes/2006/89-resolucao-454>>. Citado na página 31.
- ANATEL. Internet das Coisas.** 2020. Disponível em: <<https://www.gov.br/anatel/pt-br/regulado/outorga/internet-das-coisas>>. Citado na página 31.
- ANATEL. Aspectos Regulatórios da Internet das Coisas (IoT) e Sistemas de Comunicação Máquina a Máquina.** 2021. Disponível em: <<https://sistemas.anatel.gov.br/anexar-api/publico/anexos/download/a028ab5cc4e3f97442830bba0c8bd1dd>>. Citado na página 31.
- AOSONG. AM2302 Product Manual.** 2014. Disponível em: <https://www.filipeflop.com/img/files/download/Datasheet_DHT22_AM2302.pdf>. Citado 2 vezes nas páginas 46 e 47.
- ARBEX, M. A. et al. Queima de Biomassa e Efeitos sobre a Saúde.** 2004. Disponível em: <<https://www.scielo.br/j/jbpneu/a/VNXXmdyPSjxJDCStkYrZSzz>>. Citado na página 20.
- ARDUINO. DHT sensor library for ESPx.** 2022. Disponível em: <<https://www.arduino.cc/reference/en/libraries/dht-sensor-library-for-espx>>. Citado na página 53.
- ARDUINO. PMS Library.** 2022. Disponível em: <<https://www.arduino.cc/reference/en/libraries/pms-library>>. Citado na página 54.
- ARDUINO. SoftwareSerial Library.** 2023. Disponível em: <<https://docs.arduino.cc/learn/built-in-libraries/software-serial>>. Citado na página 67.
- BAÚ DA ELETRÔNICA. Sensor de Temperatura e Umidade DHT22.** 2021. Disponível em: <<https://www.baudaelectronica.com.br/sensor-de-temperatura-e-umidade-dht22.html>>. Citado na página 46.
- BOR, M. et al. Do LoRa Low-Power Wide-Area Networks Scale?** 2016. Disponível em: <<https://uu.diva-portal.org/smash/get/diva2:1044681/FULLTEXT01.pdf>>. Citado 3 vezes nas páginas 33, 34 e 35.
- BOUGUERA, T. et al. Energy Consumption Model for Sensor Nodes Based on LoRa and LoRaWAN.** [S.l.]: Sensors, 2018. Citado na página 43.
- BRASIL. Lei Nº 6.938, de 31 de Agosto de 1981.** 1981. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/16938.htm>. Citado na página 25.
- BRASIL. Constituição da República Federativa do Brasil.** 1988. Disponível em: <<https://www2.camara.leg.br/atividade-legislativa/legislacao/constituicao1988>>. Citado na página 25.
- CETESB. Qualidade do Ar no Estado de São Paulo.** 2017. Disponível em: <<https://cetesb.sp.gov.br/ar/wp-content/uploads/sites/28/2019/05/Relatório-de-Qualidade-do-Ar-2017.pdf>>. Citado 2 vezes nas páginas 27 e 28.

CONAMA. Resolução Nº 005, de 15 de Junho de 1989. 1989. Disponível em: <<http://www.ibama.gov.br/sophia/cnia/legislacao/MMA/RE0005-150689.PDF>>. Citado na página 25.

CONAMA. RESOLUÇÃO Nº 491, DE 19 DE NOVEMBRO DE 2018. 2018. Disponível em: <https://www.in.gov.br/web/guest/materia/-/asset_publisher/Kujrw0TZC2Mb/content/id/51058895/do1-2018-11-21-resolucao-n-491-de-19-de-novembro-de-2018-51058603>. Citado na página 25.

CPRM, S. G. D. B. Avaliação dos Recursos Minerários do Brasil - Projeto Gipsita no Sudeste da Bacia Sedimentar do Araripe. 2019. Disponível em: <https://www.researchgate.net/publication/340849550_PROJETO_GIPSITA_NO_SUDOESTE_DA_BACIA_DO_ARARIPE>. Citado na página 18.

CSDN. LoRa signal synchronization principle. 2021. Disponível em: <<https://www.csdn.net/tags/0tTaQg2sNTU00TktYmxvZw0000000000.html>>. Citado na página 34.

DNPM, D. N. D. P. M. Balanço Mineral Brasileiro 2001. 2001. Disponível em: <<https://www.gov.br/anm/pt-br/centrais-de-conteudo/dnpm/paginas/balanco-mineral/arquivos/balanco-mineral-brasileiro-2001-gipsita>>. Citado na página 19.

DRAGINO. LPS8 Indoor LoRaWAN Gateway. 2022. Disponível em: <<https://www.dragino.com/products/lora-lorawan-gateway/item/148-lps8.html>>. Citado 3 vezes nas páginas 51, 52 e 83.

EPA, U. Particulate Matter (PM) Basics. 2022. Disponível em: <<https://www.epa.gov/pm-pollution/particulate-matter-pm-basics>>. Citado na página 22.

ESPRESSIF. ESP32 Series Datasheet. 2022. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Citado na página 45.

EUROPE, W. R. O. for. Air quality guidelines: global update 2005: particulate matter, ozone, nitrogen dioxide and sulfur dioxide. 2006. Disponível em: <<https://apps.who.int/iris/handle/10665/107823>>. Citado 2 vezes nas páginas 22 e 23.

EUROPE, W. R. O. for. WHO Guidelines For Indoor Air Quality: Selected Pollutants. 2010. Disponível em: <<https://apps.who.int/iris/handle/10665/260127>>. Citado na página 24.

GRANJA, C. V. A. et al. Características Temporais da Concentração de Material Particulado na Atmosfera da Cidade de São Carlos - SP. 2017. Citado na página 16.

HEALTH, N. Y. D. of. Fine Particles (PM 2.5) Questions and Answers. 2018. Disponível em: <https://www.health.ny.gov/environmental/indoors/air/pmq_a.htm>. Citado na página 23.

HELTEC. WiFi LoRa 32 (V2.1). 2018. Disponível em: <<https://heltec.org/project/wifi-lora-32/>>. Citado 4 vezes nas páginas 45, 67, 78 e 79.

HELTEC. Heltec ESP32 Dev-Boards. 2022. Disponível em: <<https://www.arduinolibraries.info/libraries/heltec-esp32-dev-boards>>. Citado na página 65.

IBM CORPORATION. **IBM LoRaWAN in C (LMiC) version 1.5.** 2015. Disponível em: <<http://wiki.lahoud.fr/lib/exe/fetch.php?media=lmic-v1.5.pdf>>. Citado na página 54.

ITU. **Radio Regulations 1 Articles.** 1998. Disponível em: <<https://search.itu.int/history/HistoryDigitalCollectionDocLibrary/1.14.48.en.101.pdf>>. Citado na página 30.

KOOIJMAN, M. **Arduino-LMIC library.** 2020. Disponível em: <<https://github.com/matthijskooijman/arduino-lmic>>. Citado na página 78.

LINKLABS. **A Comprehensive Look at Low Power, Wide Area Networks.** 2016. Disponível em: <<http://www.link-labs.com/hubfs/LPWANWhitepaper-5.pdf>>. Citado na página 29.

LISBOA, H. de M.; KAWANO, M. **Controle da Poluição Atmosférica.** 2007. Disponível em: <<http://repositorio.asces.edu.br/handle/123456789/418>>. Citado na página 26.

LORA ALLIANCE. **What is LoRaWAN.** 2015. Disponível em: <<https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf>>. Citado 4 vezes nas páginas 37, 41, 42 e 43.

LORA ALLIANCE. **LoRaWAN 1.0.3 Specifications.** 2018. Disponível em: <<https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf>>. Citado na página 37.

LORA ALLIANCE. **RP002-1.0.1 LoRaWAN Regional Parameters.** 2020. Disponível em: <https://lora-alliance.org/wp-content/uploads/2020/11/rp_2-1.0.1.pdf>. Citado na página 31.

LORA ALLIANCE. **TR007 Developing LoRaWAN® Devices.** 2021. Disponível em: <https://lora-alliance.org/wp-content/uploads/2021/05/TR007_Developing_LoRaWAN_Devices-v1.0.0.pdf>. Citado 2 vezes nas páginas 38 e 39.

MARTINS, S. M. S. de A. L. A. M. **A Indústria Extrativa Mineral do Pólo Gesseiro do Araripe e seus Impactos Sócio-Ambientais.** 2011. Disponível em: <<https://periodicos.ufpe.br/revistas/revistageografia/article/view/228966/23374>>. Citado 2 vezes nas páginas 19 e 20.

MAUDET, S. *et al.* **Refined Node Energy Consumption Modeling in a LoRaWAN Network.** [S.I.]: Sensors, 2021. Citado 2 vezes nas páginas 32 e 38.

MCCI CATENA. **Arduino-LMIC library ("MCCI LoRaWAN LMIC Library").** 2022. Disponível em: <<https://github.com/mcci-catena/arduino-lmic>>. Citado 2 vezes nas páginas 54 e 78.

MCCI CORPORATION. **Arduino LoRaWAN MAC in C (LMiC).** 2018. Disponível em: <<https://redmine.laas.fr/attachments/download/1505/LMIC-v2.3.pdf>>. Citado 5 vezes nas páginas 54, 55, 56, 78 e 79.

MEDEIROS, M. S. de; SILVA, L. G. A.; HURTADO-GUERRERO, J. C. **A Saúde no Contexto do Polo Gesseiro de Araripina-Pernambuco, Brasil.** 2010. Citado 3 vezes nas páginas 16, 18 e 20.

MMA. Guia Técnico para a Avaliação e Monitoramento da Qualidade do Ar. 2019. Disponível em: <<https://www.gov.br/mma/pt-br/centrais-de-conteudo/mma-guia-tecnico-qualidade-do-ar-pdf>>. Citado 2 vezes nas páginas 16 e 26.

MPT-PE. Polo Gesseiro | Em São Paulo, MPT e OIT propõem a empresas pacto contra trabalho precário. 2019. Disponível em: <<https://www.prt6.mpt.mp.br/informe-se/noticias-do-mpt-go/1541-polo-gesseiro-em-sao-paulo-mpt-e-oit-propoem-a-empresas-pacto-contra-trabalho-precario>>. Citado na página 18.

MUTEBA, F; DJOUANI, K; OLWAL, T. A comparative Survey Study on LPWA IoT Technologies: Design, considerations, challenges and solutions. 2019. Citado na página 17.

NSW. Particulate matter (PM10 and PM2.5). 2020. Disponível em: <<https://www.health.nsw.gov.au/environment/air/Pages/particulate-matter.aspx>>. Citado na página 23.

OIT. Cadeia Produtiva do Gesso - Avanços e desafios rumo à promoção do Trabalho Decente: análise situacional. 2021. Disponível em: <<https://bibliotecadigital.fgv.br/dspace/handle/10438/31059>>. Citado 2 vezes nas páginas 18 e 19.

OPENWRT. Welcome to the OpenWrt Project. 2023. Disponível em: <<https://openwrt.org/start>>. Citado na página 52.

PANDAS. Pandas. 2023. Disponível em: <<https://pandas.pydata.org/>>. Citado na página 62.

PARENTE, L. L. LMIC-node. 2022. Disponível em: <<https://github.com/lnlp/LMIC-node>>. Citado 5 vezes nas páginas 54, 75, 76, 78 e 81.

PEREZ, X. The Things Stack LoRaWAN Network Server - Open Source. 2023. Disponível em: <<https://hub.docker.com/r/xoseperez/the-things-stack>>. Citado na página 57.

PLANTOWER. PMS5003 Series Data Manual. 2016. Disponível em: <<https://www.digikey.jp/htmldatasheets/production/2903006/0/0/1/PMS5003-Series-Manual.pdf>>. Citado 4 vezes nas páginas 48, 49, 50 e 52.

PLATFORMIO. Professional collaborative platform for embedded development. 2023. Disponível em: <<https://docs.platformio.org/en/latest/>>. Citado na página 53.

PLUIJM, A. van der. Single Channel TTN LoRa Gateway and nodes with ESP32 SX1276. 2018. Disponível em: <<https://www.hackster.io/Arn/single-channel-ttn-lora-gateway-and-nodes-with-esp32-sx1276-709612>>. Citado na página 44.

PYTHON. sqlite3 - DB-API 2.0 interface for SQLite databases. 2023. Disponível em: <<https://docs.python.org/3/library/sqlite3.html>>. Citado na página 61.

ROBIUL ELECTRONICS. PMS5003 PM2.5 Air Quality Sensor. 2022. Disponível em: <<https://www.robiulelectronics.com/site/prdView/1236>>. Citado na página 48.

SANTOS, J. P. de O.; EL-DEIR, S. G. Produção de Gesso no Araripe Pernambucano: Impactos Ambientais e Perspectivas Futuras. 2019. Citado 3 vezes nas páginas 16, 20 e 21.

SEMTECH. AN1200.22 LoRa™ Modulation Basics. 2015. Disponível em: <<https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf>>. Citado 2 vezes nas páginas 30 e 32.

SEMTECH. LoRa® and LoRaWAN®: A Technical Overview. 2019. Disponível em: <https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf>. Citado 10 vezes nas páginas 29, 31, 33, 34, 35, 36, 37, 40, 41 e 42.

SEMTECH. SX1272/73 - 860 MHz to 1020 MHz Low Power Long Range Transceiver. 2019. Disponível em: <https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/440000001NCE/v_VBhk1IolDgxwn0pcS_vTFxPfSEPQbuneK3mWsX1U>. Citado na página 30.

SEMTECH. Building a LoRa®-based Device End-to-End with Arduino. 2020. Disponível em: <https://lora-developers.semtech.com/uploads/documents/files/Building_a_LoRa-based_Device_End-to-End_with_Arduino_Dnld_Final.pdf>. Citado na página 54.

SEMTECH. SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver. 2020. Disponível em: <https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiaBqcpqh9qSjE>. Citado 2 vezes nas páginas 31 e 45.

SILVA, J. A. A. da et al. Produtividade Volumétrica de Clones de Eucalyptus spp. no Polo Gesseiro do Araripe, Pernambuco. 2013. Disponível em: <<http://www.journals.ufrpe.br/index.php/apca/article/view/314>>. Citado na página 20.

SINDUSGESSO. Potencialidades do polo gesseiro do araripe: Simpósio polo gesseiro do araripe: potencialidades, problemas e soluções. 2014. Citado 2 vezes nas páginas 16 e 18.

SQLALCHEMY. SQLAlchemy 2.0 Documentation. 2023. Disponível em: <<https://docs.sqlalchemy.org/en/20/>>. Citado na página 62.

SQLITE. SQLite3 - DB-API 2.0 interface for SQLite Documentation. 2023. Disponível em: <<https://www.sqlite.org/docs.html>>. Citado na página 61.

STREAMLIT. Streamlit documentation. 2023. Disponível em: <<https://docs.streamlit.io/>>. Citado na página 62.

THANKS BUYER. DRAGINO LPS8 Indoor IoT Gateway For LoRaWAN SX1308 LoRa Concentrator Version 915 For US915 AU915. 2022. Disponível em: <<https://www.thanksbuyer.com/dragino-lps8-indoor-iot-gateway-for-lorawan-sx1308-lora-concentrator-version-915-for-us915-au915-73788>>. Citado na página 51.

TTI. The Things Stack plans. 2022. Disponível em: <<https://www.thethingsindustries.com/tts-plans/>>. Citado 2 vezes nas páginas 57 e 58.

TTI. What Is The Things Stack? 2022. Disponível em: <<https://www.thethingsindustries.com/docs/getting-started/what-is-tts/>>. Citado 2 vezes nas páginas 57 e 58.

TTN. LoRaWAN Overview. 2022. Disponível em: <<https://www.thethingsnetwork.org/docs/lorawan/>>. Citado 2 vezes nas páginas 59 e 60.

TTN. The Things Network - Quick Start. 2022. Disponível em: <<https://www.thethingsnetwork.org/docs/quick-start/>>. Citado 2 vezes nas páginas 58 e 59.

WAI, W. T. *et al.* **A Study of the Air Pollution Index Reporting System.** 2012. Disponível em: <http://www.aqhi.gov.hk/pdf/related_websites/APIreview_report.pdf>. Citado na página 27.

WHO. **WHO Global Air Quality Guidelines.** 2021. Disponível em: <<https://apps.who.int/iris/handle/10665/345329>>. Citado 2 vezes nas páginas 16 e 22.

Apêndices

APÊNDICE A – ARQUIVO *PLATFORMIO.INI* PARA O ENSAIO COM O DHT22

Conteúdo do arquivo *platformio.ini* utilizado no projeto PlatformIO criado para o ensaio de integração do sensor DHT22. O arquivo com este conteúdo pode ser encontrado em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-trials/dht22-trial/platformio.ini>>.

```
[env:heltec_wifi_lora_32_V2]
platform = espressif32
board = heltec_wifi_lora_32_V2
framework = arduino
lib_deps =
    heltecautomation/Heltec ESP32 Dev-Boards@^1.1.1
    adafruit/DHT sensor library@^1.4.4
```

APÊNDICE B – CÓDIGO FONTE PARA ENSAIO COM O DHT22

Código fonte criado para o ensaio de integração do sensor DHT22. O arquivo com este conteúdo pode ser encontrado em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-trials/dht22-trial/src/main.cpp>>.

```
// dependências sobre framework arduino e placa Heltec
#include <Arduino.h>
#include <heltec.h>

// dependências sobre sensores Adafruit
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

// definição de modelo do sensor e de pino utilizado para dados
#define DHTPIN 4
#define DHTTYPE DHT22

// instância manipulador do sensor DHT22
DHT_Unified dht(DHTPIN, DHTTYPE);

void setup()
{
    // inicializa o funcionamento da porta serial
    Serial.begin(9600);

    // configura DHT22 antes da inicialização
    sensor_t sensor;
    dht.temperature().getSensor(&sensor);
    dht.humidity().getSensor(&sensor);

    // inicializa funcionamento do DHT22
    dht.begin();
}

void loop()
{
    // estabelece delay entre leituras
    delay(3000);

    // declara objeto evento para a captura das leituras
    sensors_event_t event;

    // declara variáveis auxiliáres temporárias e flags
```

```
float temp, rh;
bool hasReadTemp = false, hasReadRH = false;

// tenta capturar leituras de temperatura no objeto evento
dht.temperature().getEvent(&event);
delay(1000);
if (!isnan(event.temperature))
{
    temp = event.temperature;
    hasReadTemp = true;
}

// tenta capturar leituras de umidade relativa no objeto evento
dht.humidity().getEvent(&event);
delay(1000);
if (!isnan(event.relative_humidity))
{
    rh = event.relative_humidity;
    hasReadRH = true;
}

// verifica se os dois valores foram capturados com sucesso
// e imprime-os na tela
if(hasReadTemp && hasReadRH)
{
    Serial.print(F("{ \"temp\": "));
    Serial.print(temp);
    Serial.print(F(", \"rh\": "));
    Serial.print(rh);
    Serial.println(F(" }"));
} else
    Serial.println(F("Erro na leitura do sensor!"));
}
```

APÊNDICE C – ARQUIVO *PLATFORMIO.INI* PARA O ENSAIO COM O PMS5003

Conteúdo do arquivo *platformio.ini* utilizado no projeto PlatformIO criado para o ensaio de integração do sensor PMS5003. Este arquivo pode ser encontrado em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-trials/pms5003-trial/platformio.ini>>.

```
[env:heltec_wifi_lora_32_V2]
platform = espressif32
board = heltec_wifi_lora_32_V2
framework = arduino
lib_deps =
    adafruit/Adafruit PM25 AQI Sensor@~1.0.6
    adafruit/Adafruit Unified Sensor@~1.1.7
    heltecautomation/Heltec ESP32 Dev-Boards@~1.1.1
    fu-hsi/PMS Library@~1.1.0
```

APÊNDICE D – CÓDIGO FONTE PARA ENSAIO COM O PMS5003

Código fonte criado para o ensaio de integração do sensor PMS5003. O arquivo com este conteúdo pode ser encontrado em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-trials/pms5003-trial/src/main.cpp>>.

```
// dependências sobre framework arduino e placa Heltec
#include <Arduino.h>
#include <heltec.h>

// dependências para manipulação do PMS5003
#include <Adafruit_Sensor.h>
#include <Adafruit_PM25AQI.h>
#include <PMS.h>

// declaração de objetos necessários para a manipulação do sensor
PMS pms(Serial2);
PMS::DATA data;

void setup()
{
    // inicializa interface serial utilizada para comunicação
    // com o terminal
    Serial.begin(9600);

    // inicializa interface serial utilizada para comunicação
    // com o PMS5003
    Serial2.begin(9600, SERIAL_8N1, 33, 32);

    // inicializa e configura o PMS5003 para atuar no modo passivo
    pms.passiveMode();
}

void loop()
{
    // acorda o PMS5003, ativando seu fluxo de ar por 30 segundos para
    // preparar o sensor para a próxima leitura
    pms.wakeUp();
    delay(30000);

    // requisita a leitura do PMS5003
    pms.requestRead();

    // espera por um segundo a chegada da resposta com a leitura
    if (pms.readUntil(data))
}
```

```
{  
    // imprime valores obtidos pela leitura do PMS5003  
  
    Serial.print(F("{ \"pm1_0\": "));  
    Serial.print(data.PM_AE_UG_1_0);  
  
    Serial.print(F(", \"pm2_5\": "));  
    Serial.print(data.PM_AE_UG_2_5);  
  
    Serial.print(F(", \"pm10_0\": "));  
    Serial.print(data.PM_AE_UG_10_0);  
    Serial.println(F(" }"));  
}  
else  
    Serial.println("Erro na leitura do sensor!");  
  
// coloca o PMS no modo sleep por 30 segundos  
pms.sleep();  
delay(30000);  
}
```

APÊNDICE E – ARQUIVO *PLATFORMIO.INI* PARA O ENSAIO UNIFICADO

Conteúdo do arquivo *platformio.ini* utilizado no projeto PlatformIO criado para o ensaio de integração unificando leituras dos sensores DHT22 e PMS5003. Este arquivo pode ser encontrado em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-trials/unified-trial/platformio.ini>>.

```
[env:heltec_wifi_lora_32_V2]
platform = espressif32
board = heltec_wifi_lora_32_V2
framework = arduino
lib_deps =
    heltecautomation/Heltec ESP32 Dev-Boards@~1.1.1
    adafruit/DHT sensor library@~1.4.4
    adafruit/Adafruit Unified Sensor@~1.1.7
    adafruit/Adafruit PM25 AQI Sensor@~1.0.6
    fu-hsi/PMS Library@~1.1.0
```

APÊNDICE F – CÓDIGO FONTE PARA ENSAIO UNIFICADO

Código fonte criado para o ensaio de integração unificando leituras dos sensores DHT22 e PMS5003. O arquivo com este conteúdo pode ser encontrado em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-trials/unified-trial/src/main.cpp>>.

```
// dependências sobre framework arduino e placa Heltec
#include <Arduino.h>
#include <heltec.h>

// dependências para manipulação dos sensores
#include <Adafruit_Sensor.h>
#include <Adafruit_PM25AQI.h>
#include <DHT.h>
#include <DHT_U.h>
#include <PMS.h>

// definição de modelo do sensor e de pino utilizado para dados
#define DHTPIN 4
#define DHTTYPE DHT22

// objetos globais para manipulação do DHT22
DHT_Unified dht(DHTPIN, DHTTYPE);
sensor_t sensor;

// objetos globais para manipulação do PMS5003
PMS pms(Serial2);
PMS::DATA data;

void setup()
{
    // inicializa o funcionamento da porta serial
    Serial.begin(9600);

    // utiliza o objeto sensor para configurar e inicializar
    // o funcionamento do DHT22
    dht.temperature().getSensor(&sensor);
    dht.humidity().getSensor(&sensor);
    dht.begin();

    // inicializa interface UART2 do ESP32, utilizando os
    // pinos 32 (TX) e 33 (RX)
    Serial2.begin(9600, SERIAL_8N1, 33, 32);

    // configura o PMS5003 no modo passivo inicia sua operação
```

```

// no modo awake
pms.passiveMode();
pms.wakeUp();
}

void loop()
{
    // declara objeto evento para a captura das leituras
    sensors_event_t event;

    // declara variáveis utilizadas para armazenamento
    // temporário e flags
    float temp, rh;
    uint16_t pm10_0, pm2_5, pm1_0;
    bool hasReadTemp = false, hasReadRH = false, hasReadPM = false;

    // acorda o PMS5003, ativando seu fluxo de ar por 15 segundos
    // para preparar o sensor para a próxima leitura
    pms.wakeUp();
    delay(15000);

    // tenta capturar leituras de temperatura no objeto evento
    dht.temperature().getEvent(&event);
    delay(1000);
    if (!isnan(event.temperature))
    {
        temp = event.temperature;
        hasReadTemp = true;
    }

    // tenta capturar leituras de umidade relativa no objeto evento
    dht.humidity().getEvent(&event);
    delay(1000);
    if (!isnan(event.relative_humidity))
    {
        rh = event.relative_humidity;
        hasReadRH = true;
    }

    // espera por um segundo a chegada da resposta
    // com a leitura
    if(pms.readUntil(data))
    {
        pm1_0 = data.PM_AE_UG_1_0;
        pm2_5 = data.PM_AE_UG_2_5;
        pm10_0 = data.PM_AE_UG_10_0;
        hasReadPM = true;
    }
}

```

```
// verifica se os dois valores foram capturados com
// sucesso e apresenta-os no terminal serial
if(hasReadTemp && hasReadRH && hasReadPM)
{
    Serial.print(F("{ \"temp\": "));
    Serial.print(temp);
    Serial.print(F(", \"rh\": "));
    Serial.print(rh);
    Serial.print(F(", \"pm1_0\": "));
    Serial.print(pm1_0);
    Serial.print(F(", \"pm2_5\": "));
    Serial.print(pm2_5);
    Serial.print(F(", \"pm10_0\": "));
    Serial.print(pm10_0);
    Serial.println(F(" }"));
} else
    Serial.println(F("Erro na leitura do sensor!"));

// coloca o PMS5003 no modo sleep por 15 segundos
pms.sleep();
delay(15000);
}
```

APÊNDICE G – ARQUIVO *PLATFORMIO.INI* NA CUSTOMIZAÇÃO DO *END DEVICE*

Conteúdo do arquivo *platformio.ini* utilizado no projeto PlatformIO para a customização do *end device*. Este arquivo pode ser encontrado em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-enddevice/platformio.ini>>.

```
[platformio]
default_envs =
    heltec_wifi_lora_32_v2

[common]
monitor_speed = 115200
build_flags =
    -D ABP_ACTIVATION
    -D DO_WORK_INTERVAL_SECONDS=1800
lib_deps =
    olikraus/U8g2
    lnlp/EasyLed

[esp32]
build_flags =
    -D hal_init=LMICHAL_init

[mcci_lmic]
lib_deps =
    mcci-catena/MCCI LoRaWAN LMIC library
build_flags =
    -D CFG_us915=1
    -D DISABLE_PING
    -D DISABLE_BEACONS
    -D CFG_sx1276_radio=1
    -D USE_ORIGINAL_AES
    -D LMIC_DEBUG_LEVEL=0

[env:heltec_wifi_lora_32_v2]
platform = espressif32
board = heltec_wifi_lora_32_V2
framework = arduino
upload_speed = 921600
monitor_speed = ${common.monitor_speed}
lib_deps =
    ${common.lib_deps}
    ${mcci_lmic.lib_deps}
    adafruit/DHT sensor library@^1.4.4
    adafruit/Adafruit Unified Sensor@^1.1.7
```

```
adafruit/Adafruit PM25 AQI Sensor@^1.0.6
fu-hsi/PMS Library@^1.1.0
build_flags =
${common.build_flags}
${mcci_lmic.build_flags}
${esp32.build_flags}
-D BSFILE=\"boards/bsf_heltec_wifi_lora_32_v2.h\""
-D MONITOR_SPEED=${common.monitor_speed}
-D LMIC_PRINTF_TO=Serial
-D USE_SERIAL
-D USE_LED
```

APÊNDICE H – CUSTOMIZAÇÃO DO *END DEVICE* LORAWAN - DECLARAÇÕES GLOBAIS

Trecho de código fonte para declarações globais utilizadas na customização do *end device*. Este trecho está disponível em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-enddevice/src/LMIC-node.cpp>>

```
// dependências da Adafruit para os sensores
#include <Adafruit_Sensor.h>
#include <Adafruit_PM25AQI.h>
#include <DHT.h>
#include <DHT_U.h>
#include <PMS.h>

// definição de modelo do sensor e de pino utilizado para dados
#define DHTPIN 4
#define DHTTYPE DHT22

// objeto global para manipulação do DHT22
DHT_Unified dht(DHTPIN, DHTTYPE);
sensor_t sensor;

// objeto global para manipulação do PMS5003
PMS pms(Serial2);
PMS::DATA data;

// definição do tamanho máximo do buffer para payload
const uint8_t payloadBufferLength = 14;
```

APÊNDICE I – CUSTOMIZAÇÃO DO *END DEVICE* LORAWAN - TRECHO DE INICIALIZAÇÃO

Trecho de código fonte para inicializações de sensores utilizadas na customização do *end device*. Este trecho está disponível em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-enddevice/src/LMIC-node.cpp>>

```
// utiliza o objeto sensor para configurar e
// inicializar o funcionamento do DHT22
dht.temperature().getSensor(&sensor);
dht.humidity().getSensor(&sensor);
dht.begin();

// inicializa interface UART2 do ESP32, utilizando
// os pinos 32 (TX) e 33 (RX)
Serial2.begin(9600, SERIAL_8N1, 33, 32);

// configura o PMS5003 no modo passivo inicia sua
// operação no modo awake
pms.passiveMode();
pms.wakeUp();
```

APÊNDICE J – CUSTOMIZAÇÃO DO *END DEVICE* LORAWAN - TRECHO EM LOOP

Trecho de código fonte para ações de recorrência utilizadas na customização do *end device*. Este trecho está disponível em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-enddevice/src/LMIC-node.cpp>>

```

void processWork(ostime_t doWorkJobTimeStamp)
{
    // se DEVADDR é um valor já definido
    if (LMIC.devaddr != 0)
    {
        // quando não for possível o agendamento de
        // uma mensagem de uplink ...
        if(LMIC.opmode & OP_TXRXPEND);

        // quando for possível o agendamento de
        // uma mensagem de uplink ...
        else
        {
            // declara objeto evento para a captura
            // das leituras
            sensors_event_t event;

            // captura leituras de temperatura no objeto evento
            dht.temperature().getEvent(&event);
            delay(500);

            // se a captura ocorrer ...
            if(!isnan(event.temperature))
            {
                // declara variáveis auxiliares para a
                // parte inteira e decimal do valor
                // de temperatura capturado
                uint16_t interTemp = (int) event.temperature;
                uint16_t decimTemp = (event.temperature
                    - (int) event.temperature)*100.0;

                // aloca a parte inteira e decimal do valor
                // capturado no buffer para payload
                payloadBuffer[0] = interTemp >> 8;
                payloadBuffer[1] = interTemp & 0xFF;
                payloadBuffer[2] = decimTemp >> 8;
                payloadBuffer[3] = decimTemp & 0xFF;
            }
        }
    }
}

```

```

// captura leituras de umidade relativa no
// objeto evento
dht.humidity().getEvent(&event);
delay(500);

// se a captura de umidade estiver pronta ...
if(!isnan(event.relative_humidity))
{
    // declara variáveis auxiliares para a
    // parte inteira e decimal do valor
    // de umidade capturado
    uint16_t interHumid = (int) event.relative_humidity;
    uint16_t decimHumid = (event.relative_humidity
        - (int) event.relative_humidity)*100.0;

    // aloca a parte inteira e decimal do
    // valor capturado no buffer para payload
    payloadBuffer[4] = interHumid >> 8;
    payloadBuffer[5] = interHumid & 0xFF;
    payloadBuffer[6] = decimHumid >> 8;
    payloadBuffer[7] = decimHumid & 0xFF;
}

delay(500);
pms.requestRead();
delay(500);

// se a captura de presença de MP estiver pronta ...
if (pms.readUntil(data))
{
    // utiliza captura de leitura de MP1,0
    // no incremento da montagem do payload
    uint16_t pmVal = (uint16_t) data.PM_AE_UG_1_0;
    payloadBuffer[8] = pmVal >> 8;
    payloadBuffer[9] = pmVal & 0xFF;

    // utiliza captura de leitura de MP2,5
    // no incremento da montagem do payload
    pmVal = (uint16_t) data.PM_AE_UG_2_5;
    payloadBuffer[10] = pmVal >> 8;
    payloadBuffer[11] = pmVal & 0xFF;

    // utiliza captura de leitura de MP10
    // no incremento da montagem do payload
    pmVal = (uint16_t) data.PM_AE_UG_10_0;
    payloadBuffer[12] = pmVal >> 8;
    payloadBuffer[13] = pmVal & 0xFF;
}

```

```
// prepara o payload com mensagem de uplink
uint8_t fPort = 10;
uint8_t payloadLength = 14;

// agenda transmissão do payload montado
scheduleUplink(fPort, payloadBuffer, payloadLength);
}

}
```

APÊNDICE K – PAYLOAD FORMATTER INSERIDO NO TTN

Código utilizado como *Payload Formatter* dentro do TTN para decodificação dos *payloads* enviados pelo end device customizado neste trabalho. Um arquivo com este conteúdo pode ser encontrado em <<https://github.com/smurilogs/univasf-tcc-sensorar/blob/main/sensorar-enddevice/payload-formatters/lmic-node-uplink-formatters.js>>.

```

function decodeUplink(input) {
    var data = {};
    var warnings = [];

    if (input.fPort == 10) {
        data.temp = ((input.bytes[0] << 8) + input.bytes[1])
            + (((input.bytes[2] << 8) + input.bytes[3])/100.0);

        data.rh = ((input.bytes[4] << 8)
            + input.bytes[5]) + (((input.bytes[6] << 8)
            + input.bytes[7])/100.0);

        data.pm1_0 = ((input.bytes[8] << 8) | input.bytes[9]);
        data.pm2_5 = ((input.bytes[10] << 8) | input.bytes[11]);
        data.pm10_0 = ((input.bytes[12] << 8) | input.bytes[13]);
    }
    else {
        warnings.push("Unsupported fPort");
    }
    return {
        data: data,
        warnings: warnings
    };
}

```

APÊNDICE L – INGESTÃO DA DADOS E PERSISTÊNCIA

Diretório onde se encontra o código fonte da aplicação de ingestão para a persistência dos dados. Disponível em <<https://github.com/smurilogs/univasf-tcc-sensorar/tree/main/sensorar-webapp/src/ingestor>>.

APÊNDICE M – APRESENTAÇÃO DOS DADOS EM *DASHBOARD*

Diretório onde se encontra o código fonte da aplicação *web* para a apresentação dos dados na forma de dashboard. Disponível em <<https://github.com/smurilogs/univasf-tcc-sensorar/tree/main/sensorar-webapp/src/dashboard>>.