

Helal, Alan A.

EstAcqua: Proposta de solução integrada de Hardware, Software para monitoramento ambiental

Vitória - ES

2018

Helal, Alan A.

EstAcqua: Proposta de solução integrada de Hardware, Software para monitoramento ambiental

Dissertação de mestrado submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do título de Mestre em Informática. Área de concentração: Ciência da Computação.

Universidade Federal do Espírito Santo – UFES

Departamento de Informática

Programa de Pós-Graduação em Informática

Orientador: Rodolfo da Silva Villaça

Coorientador: Roberto Colistete Júnior

Vitória - ES

2018

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

Helal, Alan Afif, 1987-
H474e EstAcqua : Proposta de solução integrada de hardware,
software para monitoramento ambiental / Alan Afif Helal. – 2018.
135 f. : il.

Orientador: Rodolfo da Silva Villaça.
Coorientador: Roberto Colistete Júnior.
Dissertação (Mestrado em Informática) – Universidade
Federal do Espírito Santo, Centro Tecnológico.

1. Internet das coisas. 2. Sistemas de transmissão de dados.
3. Monitoramento ambiental. 4. Hardware. 5. Software. I. Villaça,
Rodolfo da Silva. II. Colistete Júnior, Roberto. III. Universidade
Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004



EstAcqua: Proposta de solução integrada de Hardware, Software para monitoramento ambiental

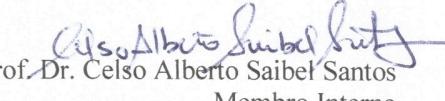
Alan Afif Helal

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

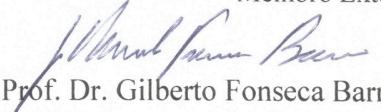
Aprovada em 27 de agosto de 2018:


Prof. Dr. Rodolfo da Silva Villaca
Orientador


Prof. Dr. Roberto Colistete Júnior
Coorientador


Prof. Dr. Celso Alberto Saibel Santos
Membro Interno


Prof. Dr. Moisés Renato Nunes Ribeiro
Membro Externo


Prof. Dr. Gilberto Fonseca Barroso
(Membro Externo)

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória-ES, 27 de agosto de 2018.

Este trabalho é dedicado à memória de minha avó, Almira Soares Freire.

Agradecimentos

Agradeço à minha mãe, que apesar de todas as dificuldades sempre torceu pro mim e me apoiou em todas as minhas decisões.

À minha amável esposa, por todo o apoio, carinho, compreensão e paciência durante esse período de muitas ausências, além de ser meu porto seguro nos momentos de fraqueza.

Ao laboratório NERDS e o Projeto FUTEBOL por terem cedido verba para a aquisição de sensores e microcontroladores utilizados neste trabalho.

Ao meu orientador, Rodolfo Villaça, e co-orientador, Roberto Colistete Jr, por toda a paciência, ajuda e direcionamento que foram fundamentais para a concretização deste trabalho, além de contribuirem com sensores, cabos, multímetros, microcontroladores, dentre outros equipamentos adquiridos com verba própria.

Ao Departamento de Oceanografia e Ecologia da UFES, em especial aos Professores Gilberto Fonseca e Fábio Garcia por toda a ajuda nas especificações da EstAcqua, além de toda a paciência nas visitas ao lago Terra Alta para teste e análise da mesma.

Ao meu grande amigo José Martins por sempre me ajudar nas horas de desespero, seja resolvendo o problema ou dando sugestões.

Aos demais amigos e familiares que de alguma forma colaboraram para que essa jornada fosse menos tortuosa.

*"There are so many different worlds
So many different suns
And we have just one world
But we live in different ones"*
(Mark Knopfler)

Resumo

Com a expansão da Internet das Coisas, diversas soluções para monitoramento ambiental estão disponíveis no mercado. Entretanto, a maioria das soluções utilizam software proprietário, de custo elevado e não oferecem monitoramento online, dificultando o acesso aos dados e impedindo que ações sejam tomadas de forma preventiva. Na literatura encontram-se vários estudos realizados sobre diversas formas de transmissão de dados sem fio, contudo a grande maioria são simulações ou apenas testes de transmissão em ambientes controlados. Este trabalho apresenta a EstAcqua, uma solução integrada de hardware e software de baixo custo, utilizando conceitos de Internet das Coisas com transmissão de dados sem fio via LoRaWAN. O objetivo principal dessa estação é o monitoramento de dados ambientais e oceanográficos de sensores de superfície e submersos, com baixo consumo de energia e podendo ser acessada remotamente em tempo real. Com isso, o propósito é disponibilizar uma solução simples de implementar e realizar manutenções, que possibilita a integração com dispositivos de diferentes fabricantes e permite ao usuário decidir quais componentes deseja utilizar, independentemente do fabricante. Foram realizados testes em ambientes controlados e reais para analisar o comportamento da EstAcqua durante o seu desenvolvimento. Esse processo foi incremental, no qual cada teste contribuiu para a detecção de falhas, servindo de base para melhorias na EstAcqua. Após um mês de funcionamento da EstAcqua em um ambiente real, verificou-se que é possível a instalação em ambientes de difícil acesso. A interface de monitoramento online mostrou-se suficiente para suprir as demandas da EstAcqua. Além disso, apresentou alta autonomia de bateria. Comparando a acurácia e resolução dos sensores utilizados na EstAcqua com sensores comerciais, observa-se que são similares, entretanto os utilizados na EstAcqua chegam a ter seu custo quase cem vezes menor. Além da alta autonomia de bateria, ainda é possível recarregá-la ou até mesmo substituí-la, algo não permitido pela maioria dos produtos atualmente comercializados. Pelos resultados apresentados, podemos concluir que a EstAcqua possui sensores de resolução e acurácia próximos dos sensores comerciais utilizados, além de permitir que os dados sejam coletados e transferidos para a nuvem, possibilitando o acesso aos mesmos em qualquer lugar do mundo. A EstAcqua mostrou-se como uma solução viável, não somente para fins acadêmicos, mas também como uma solução de menor custo, confiável, com maior integração e mais funcionalidades do que a maioria das soluções comerciais.

Palavras-chave: LoRa. IdC. internet das coisas. monitoramento ambiental.

Abstract

With the expansion of Internet of Things, several solutions for environmental monitoring are available in the market. However, most solutions use proprietary and costly software and do not provide online monitoring, making data access difficult, and preventing actions from being taken in a preventative manner. In the literature there are several studies carried out on various forms of wireless data transmission, but most are simulations or only transmission tests in controlled environments. This work presents EstAcqua, an integrated solution of hardware and software of low cost, using concepts of Internet of Things with wireless data transmission via LoRaWAN. The main objective of this station is the monitoring of environmental and oceanographic data of surface and submersible sensors, with low energy consumption and can be accessed remotely in real time. The purpose is to provide a simple solution to implement and perform maintenance, which allows integration with devices from different manufacturers and allows the user to decide which components to use regardless of the manufacturer. Tests were performed in controlled and real environments to analyze the behaviour of EstAcqua during its development. This process was incremental, in which each test contributed to the detection of failures, serving as the basis for improvements in EstAcqua. After a month of operation of EstAcqua in a real environment, it was verified that it is possible to install in difficult to access environments. The online monitoring interface proved sufficient to meet the demands of EstAcqua. In addition, it presented high battery autonomy. Comparing the accuracy and resolution of the sensors used in EstAcqua with commercial sensors, it is observed that they are similar, however the ones used in EstAcqua have their cost almost one hundred times smaller. In addition to the high battery life, it is still possible to recharge or even replace it, something not allowed by most of the products currently marketed. From the results presented, we can conclude that EstAcqua has resolution and accuracy sensors close to the commercial sensors used, as well as allowing the data to be collected and transferred to the cloud, allowing access to them anywhere in the world. EstAcqua has proven to be a viable solution, not only for academic purposes, but also as a lower-cost, more reliable, more integrated and more functional solution than most solutions found in the market.

Keywords: LoRa. IoT. internet of things. environmental monitoring.

Listas de Figuras

Figura 1 – Localização da EstAcqua no lago Terra Alta (Linhares, ES) e experimento de alcance de dados via LoRaWAN	22
Figura 2 – Exemplo de dois dispositivos da Internet das Coisas usando o modelo de comunicação Dispositivo para Dispositivo	27
Figura 3 – Exemplo de dois dispositivos da Internet das Coisas usando o modelo de comunicação Dispositivo para Nuvem	28
Figura 4 – Exemplo do modelo de comunicação por compartilhamento de dados no <i>Back-End</i>	29
Figura 5 – Exemplo do modelo de comunicação Dispositivo para Gateway	30
Figura 6 – Comparação entre a largura de banda e o alcance das principais tecnologias de transmissão sem fio	31
Figura 7 – Pilha do protocolo LoRa	34
Figura 8 – Tempo necessário ao incrementar em uma unidade o SF, mantendo o BW fixo	36
Figura 9 – Relação entre <i>Spreading Factor</i> , <i>Bandwidth</i> e T_s	37
Figura 10 – Faixa de frequência 902-915 MHz demonstrando os canais de <i>uplink</i> e <i>downlink</i> juntamente com as larguras de banda de 200 kHz e 1,6 MHz .	41
Figura 11 – Típica topologia de rede LoRaWAN	43
Figura 12 – Classes de dispositivos que operam em uma rede LoRaWAN	45
Figura 13 – LoPy4 da Pycom	51
Figura 14 – Placa de Expansão da Pycom	52
Figura 15 – Sensor de luz ambiente MAX44009, produzido pela <i>MAXIM Integrated</i> .	54
Figura 16 – Diagrama de blocos do MAX44009	54
Figura 17 – Sensor de umidade, pressão atmosférica e temperatura BME280, produzido pela Bosch	55
Figura 18 – Diagrama de blocos do BME280	56
Figura 19 – Sensor de temperatura DS18B20 com encapsulamento para medições subaquáticas de temperatura, produzido pela <i>MAXIM Integrated</i>	56
Figura 20 – Diagrama de blocos do DS18B20	57
Figura 21 – Curva de erro típica do DS18B20	58
Figura 22 – <i>Gateways</i> LoRaWAN conectados a <i>The Things Network</i>	59
Figura 23 – Mapa com a distância percorrida pelo sinal transmitido até a chegada no <i>gateway</i>	60
Figura 24 – Sinal transmitido por um nó localizado no leste da Inglaterra é recebido na Holanda	60
Figura 25 – Visão geral da <i>Things Network Stack V3</i>	61

Figura 26 – Local da realização do teste e as respectivas distâncias entre os nós e o <i>gateway</i>	67
Figura 27 – Tempo de ar do pacote enviado por configuração	68
Figura 28 – Perda de pacotes por configuração nas distâncias de 500 m e 2,23 km	69
Figura 29 – Vazão, em <i>bits</i> por segundo, por configuração nas distâncias de 500 m e 2,23 km	70
Figura 30 – Porcentagem da vazão em relação ao cálculo de transferência de <i>bits</i> teórico para cada configuração	71
Figura 31 – <i>Data Extraction Rate</i> para cada simulação	75
Figura 32 – Pacotes transmitidos e pacotes que sofreram colisão para cada simulação	75
Figura 33 – Arquitetura da EstAcqua	77
Figura 34 – <i>Hardware</i> da EstAcqua	79
Figura 35 – Primeira versão da EstAcqua	81
Figura 36 – Lagoa da UFES situada a 128 m de distância do Departamento de Oceanografia	82
Figura 37 – EstAcqua V2 coletando dados na lagoa situada dentro da UFES	83
Figura 38 – EstAcqua V2 instalada na balsa no lago Terra Alta	83
Figura 39 – Terceira versão da EstAcqua sendo testada na lagoa da UFES	84
Figura 40 – Gráfico de umidade gerado pelo <i>Cayenne</i>	85
Figura 41 – Quarta versão da EstAcqua instalada na balsa no lago Terra Alta	86
Figura 42 – Placa de circuito impresso desenvolvida para a EstAcqua	87
Figura 43 – Consumo de 94.8 μA no modo <i>deep sleep</i>	87
Figura 44 – Quinta versão da EstAcqua instalada na balsa no Lago Terra Alta	88
Figura 45 – Diagrama Lógico do funcionamento do nó	92
Figura 46 – Interface <i>Web</i> do Cayenne com os dados acessados <i>online</i>	94
Figura 47 – Interface <i>mobile</i> do Cayenne com os dados acessados <i>online</i>	94
Figura 48 – Gráfico de variação de temperatura externa entre as 6 h e 13 h	95
Figura 49 – Gráfico de variação de temperatura da lagoa entre as 6 h e 13 h	95
Figura 50 – Alarme enviado automaticamente pelo <i>Cayenne</i> alertando que uma condição crítica, definida pelo usuário, ocorreu.	96
Figura 51 – Balsa localizada no lago Terra alta e o nó na balsa coletando dados	97
Figura 52 – Diagrama profundidade/tempo para a variação da temperatura na coluna de água do lago Terra Alta	98
Figura 53 – Variação da iluminância ao decorrer do dia 23/06/2018	99
Figura 54 – Gráficos de iluminância, umidade e pressão atmosférica gerados pelo Cayenne no período de 13/06/2018 a 13/07/2018	99
Figura 55 – Versão final da EstAcqua instalada no tripé existente na balsa situada no lago Terra Alta	100

Figura 56 – Tempo de carregamento dos <i>drivers</i> dos sensores e leitura dos dados no período de 13/06/2018 a 13/07/2018	101
Figura 57 – Autonomia da bateria de 2.600 mAh versus ciclo de trabalho para correntes de modo ativo $I_a = 50,0 \text{ mA}$, <i>deep sleep</i> $I_d = 0,0948 \text{ mA}$ e <i>Duty Cycle</i> $D \simeq 0,583\%$	102
Figura 58 – Autonomia da bateria de 2.600 mAh versus ciclo de trabalho para correntes de modo ativo $I_a = 50,0 \text{ mA}$, <i>deep sleep</i> $I_d = 0,0948 \text{ mA}$ e <i>Duty Cycle</i> $D \simeq 0,0972\%$	102
Figura 59 – Preços dos componentes do <i>NanoGateway</i>	105
Figura 60 – Preços dos componentes do Nό	105
Figura 61 – Equipamento necessário para efetuar a leitura do TidBit	106
Figura 62 – Valor total dos equipamentos para efetuar a leitura do TidBit	106
Figura 63 – Formato do pacote LPP	111

Lista de quadros

Quadro 1 – Comparativo entre microcontroladores	49
Quadro 2 – Comparativo com sensores comerciais	103
Quadro 3 – Codificação utilizada pelo LPP	110

Lista de tabelas

Tabela 1 – Taxa de <i>bits</i> transmitidos por segundo em função de SF e BW	37
Tabela 2 – Tamanho máximo do <i>payload</i> , em <i>Bytes</i> , por <i>Data Rate</i>	39
Tabela 3 – Comparativo das limitações LoRa na Europa e Estados Unidos da América	40
Tabela 4 – Comparativo entre os métodos de autenticação OTAA e ABP	44
Tabela 6 – Tempo de conversão por resolução do DSB18B20	57
Tabela 7 – Parâmetros selecionados para cada configuração do teste	68
Tabela 8 – Resolução e precisão por faixa de corrente contínua para o Minipa ET-2082D	101

Lista de Siglas

ABP	Activation by Personalization
ADC	Analog to Digital Converter
ADR	Adaptive Data Rate
AES	Advanced Encryption Standard
API	Application Programming Interface
ANATEL	Agência Nacional de Telecomunicações
BER	Bit Error Rate
BW	Bandwidth
CI	Circuito Integrado
CoAP	Constrained Application Protocol
CSS	Chirp Spread Spectrum
CSV	Comma Separated Value
DC	Duty Cycle
DER	Data Extraction Rate
DES	Data Encryption Standard
DR	Data Rate
EIRP	Effective Isotropic Radiated Power
ETSI	European Telecommunications Standards Institute
FAP	Fair Access Policy
FCC	Federal Communications Commission
FSK	Frequency Shifting Keying
GLEON	Global Lake Ecological Observatory Network
GND	Ground

GPIO	General Purpose Input/Output
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I ² C	Inter-integrated Circuit
I ² S	inter-IC Sound
IAB	Internet Architecture Board
IdC	Internet das Coisas
IEEE	Institute of Electrical and Electronics Engineering
IoT	Internet of Things
IP	Internet Protocol
IPSO	Internet Protocol for Smart Objects
ISM	International Scientific and Medical
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JSON	JavaScript Object Notation
JST	Japan Solderless Terminal
LiPo	Lithium Polymer
LoRa	Long Range
LPP	Low Power Packet
LPWAN	Low Power Wide Area Network
M2M	Machine to Machine
MD5	Message-Digest Algorithm 5
MIMO	Multiple Input Multiple Output
MQTT	Message Queuing Telemetry Transport
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology

OSI	Open System Interconnection
OTAA	Over-the-air Activation
OW	OneWire
pH	Potencial Hidrogeniônico
PIC	Programmable Intelligent Computer
QoS	Quality of Service
RAM	Random Access Memory
REPL	Read-Eval-Print Loop
RF	Rádio Frequênciâa
RFC	Request For Comment
RFID	Radio-Frequency IDentification
ROM	Read Only Memory
RTC	Real Time Clock
RTC	Real Time Counter
SDK	Software Development Kit
SF	Spreading Factor
SHA	Secure Hash Algorithm
SPI	Serial Peripheral Interface
SSD	Sistemas de Suporte à Decisão
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTN	The Things Network
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
UFES	Universidade Federal do Espírito Santo
UFRJ	Universidade Federal do Rio de Janeiro

ULP	Ultra Low Power
USB	Universal Serial Bus
VCC	Voltage Common Collector
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity

Sumário

1	INTRODUÇÃO	20
1.1	Proposta	21
1.2	Objetivos	22
1.2.1	Objetivo Geral	22
1.2.2	Objetivos Específicos	23
1.3	Estrutura	23
2	TRABALHOS RELACIONADOS	25
3	CONCEITOS DE BASE	27
3.0.1	Modelos de Comunicação	27
3.0.1.1	Comunicação Dispositivo para Dispositivo	27
3.0.1.2	Comunicação Dispositivo para Nuvem	28
3.0.1.3	Comunicação por compartilhamento de dados no <i>Back-End</i>	28
3.0.1.4	Comunicação Dispositivo para <i>Gateway</i>	29
3.1	Low Power Wide Area Network	29
3.1.1	Tecnologias LPWAN	32
3.2	LoRa	33
3.2.1	Modulação LoRa	34
3.2.1.1	<i>Chirp Spread Spectrum</i>	34
3.2.1.2	<i>Spreading Factor, Code Ratio e Bandwidth</i>	35
3.2.2	Frequência	39
3.2.3	LoRaWAN	42
3.2.3.1	Segurança	43
3.2.3.2	Classes dos Dispositivos	44
3.2.4	Discussão	46
3.3	Microcontroladores	47
3.3.1	Comparativo	48
3.4	MicroPython	49
4	TECNOLOGIAS UTILIZADAS	51
4.1	LoPy4	51
4.2	Placa de Expansão	52
4.3	Sensores	53
4.3.1	MAX44009	53
4.3.2	BME280	55

4.3.3	DS18B20	56
4.4	<i>The Things Network</i>	58
4.4.1	<i>LoRaWAN Network Stack</i>	61
4.4.2	Alternativas	62
4.5	<i>Cayenne</i>	62
4.6	Baterias	63
5	ANÁLISES LORA E COLISÃO DE DADOS	65
5.1	LoRa	65
5.2	Colisão de Dados	72
5.2.1	LoRaSIM	73
6	ARQUITETURA DE HARDWARE E SOFTWARE	77
6.1	<i>Hardware</i>	78
6.2	<i>Software</i>	79
7	PROTÓTIPO	81
7.1	Software	86
8	AVALIAÇÃO E RESULTADOS	93
8.1	Interface de Monitoramento via Nuvem	93
8.2	Avaliação no Iago Terra Alta	95
8.3	Autonomia de Bateria	100
8.4	Avaliação dos Sensores usados na EstAcqua	103
8.5	Custo	104
9	CONCLUSÃO	108
9.1	Trabalhos Futuros	109
	APÊNDICE A – <i>LOW POWER PACKET</i>	110
	APÊNDICE B – <i>NANO GATEWAY</i>	112
	APÊNDICE C – NÓ	123
	REFERÊNCIAS	129

1 Introdução

Segundo o relatório da *National Aeronautics and Space Administration* (NASA)¹ a temperatura global aumentou 1,0 °C desde o ano de 1880, sendo que os últimos 18 anos mais quentes ocorreram depois do ano 2000. Além do aumento da temperatura, tem-se também o aumento do nível dos mares, a diminuição das camadas polares e o aumento da emissão de dióxido de carbono. Diante disso, monitorar o meio ambiente é de extrema importância.

Dentre os ecossistemas aquáticos, os lagos têm sido considerados como efetivos sentinelas de mudanças ambientais ao integrar em seus sistemas físicos, químicos e biológicos respostas da variabilidade dos fluxos de energia e matéria, inclusive possibilitando o registro de longo prazo de mudanças climáticas (1, 2). A necessidade de avaliação das tendências dos processos de estratificação e desestratificação térmica, hidrodinâmica, distribuição de nutrientes e gases dissolvidos, como oxigênio, além de produtividade primária e secundária, demandam o monitoramento contínuo e de longo prazo de variáveis chave destes processos. A partir da análise de dados e da geração de informações subsidiam-se Sistemas de Suporte à Decisão (SSD) para gestão ambiental lacustre quanto aos problemas de eutrofização, assoreamento, contaminação e a consequente perda de bens e serviços ambientais proporcionados pelos ecossistemas aquáticos (3). Nessa linha, o Programa *Global Lake Ecological Observatory Network - GLEON* (4) tem como foco o desenvolvimento de projetos científicos inovadores para coleta, compartilhamento e interpretação de dados em alta resolução de sensores para compreensão, predição e comunicação do papel e resposta dos lagos em um ambiente global em mudança.

Atualmente, profissionais e pesquisadores fazem monitoramento de áreas urbanas e rurais a partir da obtenção de variáveis ambientais (e.g. climáticas, limnológicas, oceanográficas), visando a criação de um banco de dados com a evolução das condições ambientais. Para isso utilizam-se estações meteorológicas, através das quais são obtidos dados climáticos como temperatura do ar, precipitação da chuva, velocidade do vento, umidade e iluminância, assim como sensores hidrológicos para a obtenção de variáveis como temperatura da água, concentração de oxigênio dissolvido, pH, turbidez, entre outras.

Entretanto, os equipamentos mais comercializados para a medição dessas variáveis apresentam um grande problema: são equipamentos proprietários, de elevado custo², com código fonte fechado e sem interoperabilidade com equipamentos de outros fabricantes, como por exemplo o TidBit V2 da empresa Onset, que se trata de um *data logger*, com bateria não substituível, custando U\$ 133,00. Dessa forma, o usuário fica restrito à aquisição

¹ <https://climate.nasa.gov>

² Normalmente são utilizados diversos sensores em aplicações

de produtos do mesmo fabricante que já utiliza. Ou seja, caso outro fabricante lance um produto de melhor qualidade, com mais funcionalidades ou de menor custo, ele não será compatível com os outros produtos já adquiridos.

Dentre os procedimentos mais utilizados, os sensores são instalados no local de medição e fazem a coleta dos dados através de um *data logger* e, após um período, normalmente de um mês³, deve-se voltar até o sensor para coletar os dados registrados. Nesse ínterim, caso o sensor seja furtado ou danificado, perdem-se todos os dados coletados. Além disso, os sistemas de monitoramento em tempo contínuo convencionais não dispõem de avisos prévios no caso de uma variável ultrapassar um valor limite, como por exemplo, se a concentração de oxigênio dissolvido for inferior a 2,0 mg/L, implica em estresse fisiológico para os peixes. Caso o local de instalação dos equipamentos seja de difícil acesso, a tarefa de instalar o equipamento e coletar os dados se torna ainda mais exaustiva.

1.1 Proposta

Estamos vivendo uma época de proliferação de objetos inteligentes com capacidade de sensoriamento, processamento e comunicação (5). Nos últimos anos ocorreu uma miniaturização e redução dos preços dos sensores, tornando mais fácil a tarefa de monitorar. Com a facilidade de obtenção de sensores de baixo custo e consumo, diversas soluções de monitoramento utilizando conceitos de Internet das Coisas (do inglês *Internet of Things*) estão sendo desenvolvidas. Atualmente temos 11 bilhões de dispositivos *IoT* e, segundo a Forbes⁴, o ano de 2018 será de grandes avanços nessa área.

Nesse contexto, esta dissertação apresenta a EstAcqua, uma estação para monitoramento ambiental e oceanográfico, com sensores de superfície que coletam dados de pressão atmosférica, umidade, temperatura e iluminância, além de sensores submersos para coletar temperaturas em diferentes profundidades. Para isso, é utilizado um microcontrolador com baixo consumo de energia para efetuar as leituras dos dados dos sensores e transmiti-los via LoRaWAN para um *gateway* remoto, conectado à Internet, que repassa os dados recebidos para um servidor na nuvem.

Essa estação, além de possuir uma arquitetura aberta de implementação, com uma solução integrada de *software* e *hardware* de baixo custo, tem como diferenciais: (i) possibilidade de instalação em locais remotos e sem acesso à Internet ou sinal de celular; (ii) longo alcance de transmissão (podendo chegar a dezenas de quilômetros); (iii) acesso aos dados pela Internet; (iv) criação de gráficos mostrando a variação dos dados, com possibilidade de criação de alarmes para notificar situações que necessitam de atenção, tornando possível tomar ações preventivas; (v) fácil instalação e utilização; e (vi) longa

³ <https://www.onsetcomp.com/products/data-loggers-sensors/temperature>

⁴ <https://www.forbes.com/sites/bernardmarr/2018/01/04/the-internet-of-things-iot-will-be-massive-in-2018-here-are-the-4-predictions-from-ibm>

autonomia de bateria, dispensando a necessidade de bancos de bateria para atingir essa longevidade.

Para validação do protótipo da estação ambiental, foram realizadas medições na lagoa situada dentro da Universidade Federal do Espírito Santo (UFES) e transmitidos os dados para o Departamento de Oceanografia e Ecologia, situado a 130 m do local das medições. Também foram efetuados testes no Lago Terra Alta, no município de Linhares, no norte do estado do Espírito Santo, onde o protótipo foi colocado em uma balsa instalada no meio do lago e os dados coletados foram transmitidos para a margem do lago situada a até 3,2 km de distância, como mostra a Figura 1.

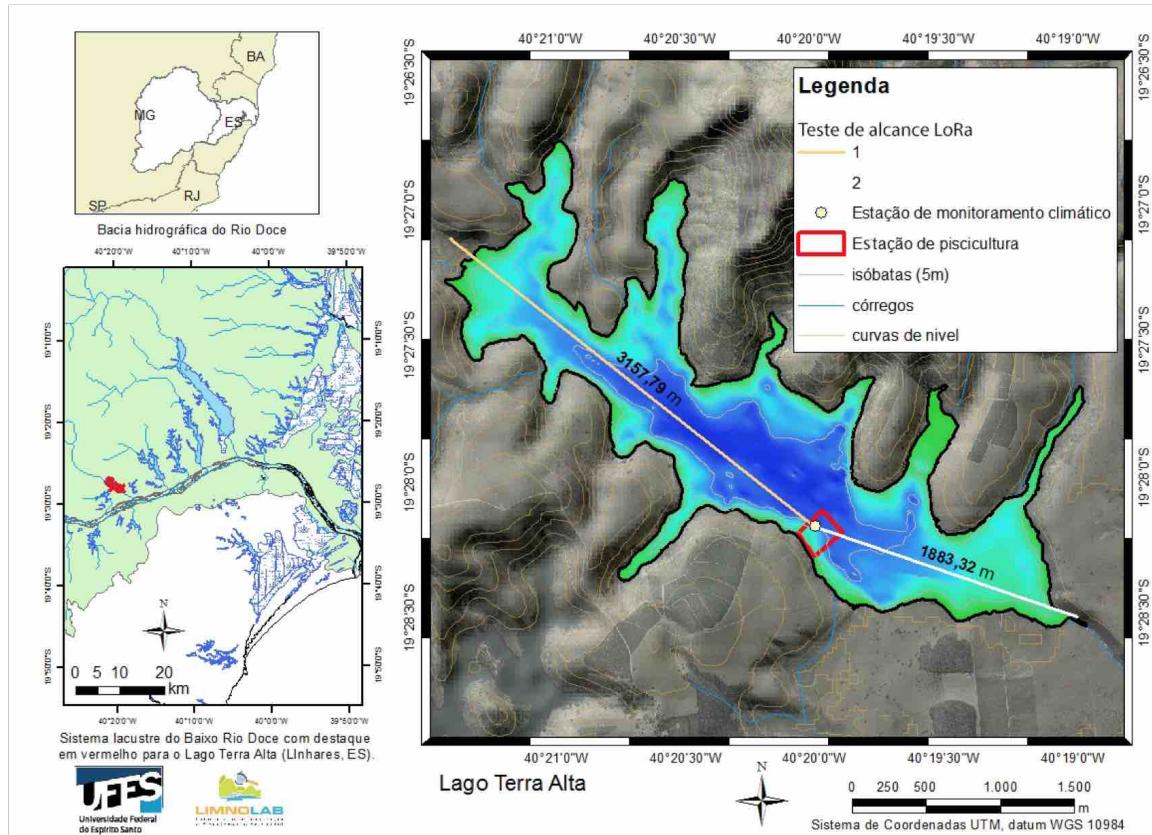


Figura 1 – Localização da EstAcqua no lago Terra Alta (Linhares, ES) e experimento de alcance de dados via LoRaWAN

1.2 Objetivos

1.2.1 Objetivo Geral

Implementar e avaliar a EstAcqua: uma solução de baixo custo integrada de *hardware* e *software* para monitoramento ambiental, utilizando sensores e microcontroladores já existentes.

1.2.2 Objetivos Específicos

- Construir um protótipo de uma estação ambiental e oceanográfica de baixo custo, com *hardware* de baixo custo, utilizando *software* aberto com baixo consumo de energia e longo alcance de transmissão.
- Utilizar tecnologias de baixo custo, tais como sensores e microcontroladores utilizados em soluções de Internet das Coisas.
- Facilitar a obtenção e visualização dos dados coletados pela estação.
- Comparar os dados coletados pela estação com dados obtidos através de equipamentos comerciais atualmente utilizados por profissionais da oceanografia.
- Disponibilizar acesso aos dados através de uma interface *web* e *mobile* para que possam ser acessados remotamente.
- Mostrar, através dos resultados, que a solução proposta é viável tanto para fins acadêmicos para pesquisadores de oceanografia como para fins comerciais.

1.3 Estrutura

Esta dissertação está estruturada da seguinte forma:

- No Capítulo 2, serão apresentados e discutidos trabalhos relacionados, destacando os diferenciais deste trabalho.
- No Capítulo 3, serão explicados os conceitos de Internet das Coisas, MicroPython e LoRa, necessários para um melhor entendimento deste trabalho.
- No Capítulo 4, serão detalhados o *hardware* utilizado para a construção do protótipo e o *software* utilizado no controle, obtenção, transmissão e visualização dos dados coletados pelos sensores.
- No Capítulo 5, serão apresentados as análises LoRa, para determinação dos parâmetros utilizados. Também serão demonstrados os procedimentos para calibração e escolha dos sensores de temperatura subaquática.
- No Capítulo 6, serão apresentadas as arquitetura de *software* e *hardware*. Serão detalhadas as características e funcionalidades do protótipo desenvolvido, bem como os desafios encontrados na implementação do mesmo.
- No Capítulo 7, será mostrada a evolução da EstAcqua, desde seu ínicio em um *protoboard* até a sua versão final. Serão demonstrados os pré-testes realizados antes de levá-la para seu local de instalação.

- No Capítulo 8, será detalhada a avaliação do protótipo, através dos dados coletados, a fim de destacar seus principais diferenciais.
- O Capítulo 9 traz conclusões gerais e sugestões de trabalhos futuros.

Os resultados preliminares deste trabalho foram publicados no Seminário Integrado de *Software e Hardware* (SEMISH) no Congresso da Sociedade Brasileira de Computação (CSBC) em julho de 2018 (6).

2 Trabalhos Relacionados

A popularização de microcontroladores de fácil programação e interface amigável proporcionou a criação de diversas soluções integradas de *hardware & software*, dentre elas para a área de monitoramento ambiental. Com a facilidade de integração com os microcontroladores, removeu-se a barreira que outrora existia: a dificuldade eletrônica para que seja possível integrar o microcontrolador com outros dispositivos.

No trabalho de [Tose\(7\)](#) foi desenvolvido um protótipo para monitoramento de estações de esgoto com Arduino conectado a sensores de temperatura, umidade e gases, utilizando uma tecnologia de comunicação sem fio, *ZigBee*, para transmissão dos dados. Entretanto, essa é uma abordagem de alto custo financeiro, com baixo alcance (devido à frequência de operação), muito sensível a interferências de outros dispositivos, com baixa autonomia de bateria e dependente de um computador localizado próximo ao local de instalação do protótipo. A EstAcqua se diferencia por ter um alcance de transmissão significativamente maior, funcionar com bateria e a possibilidade de funcionamento sem a necessidade de um computador próximo.

Em seu trabalho, [Oliveira\(8\)](#) mostra a criação de uma estação geomagnética utilizando conceitos de Internet das Coisas com uma interface *web* para visualização dos dados *online*. O magnetômetro conta com sensores de temperatura para ajudar nas medidas realizadas e armazena em um banco de dados em um cartão micro SD. O presente trabalho se diferencia por usar LoRaWAN para transmissão dos dados, ter baixo consumo de energia e ser focado em monitoramento ambiental e oceanográfico.

O trabalho de [Mois, Folea e Sanislav\(9\)](#) fez um estudo comparativo de três soluções diferentes de sensores inteligentes para o monitoramento de dados ambientais utilizando transmissão sem fio *Wi-Fi* e *Bluetooth*. Por se tratar de tecnologias de baixo alcance, o emprego dessas soluções é voltado para ambientes fechados, como escritórios. O diferencial em relação a esse trabalho é a possibilidade de uso interno ou externo com transmissão de longo alcance.

Já no trabalho de [Tzortzakis, Papafotis e Sotiriadis\(10\)](#), foi utilizado microcontroladores Atmega328 com placa de transmissão sem fio LoRa para coletar dados ambientais em cidades e transmiti-los via LoRa para um *gateway* que por sua vez encaminhava os dados para um servidor IoT utilizando GPRS. Os dispositivos que coletavam os dados eram energizados através de bateria e placa solar. O diferencial desse trabalho é utilização de nós expostos ao tempo, transmissão de dados próximos ao chão e utilização de LoRaWAN no envio dos dados coletados.

Para entender melhor sobre os parâmetros LoRa e como eles se relacionam, foi uti-

lizado como base o trabalho de [Augustin et al.\(11\)](#) onde é explicado detalhadamente o que são os parâmetros **Spreading Factor**, **TX Power**, **bandwidth** e **Code Ratio**, necessários para realizar uma transmissão de dados sem fio LoRaWAN, como eles se relacionam e as fórmulas usadas para definir seus valores.

Na grande maioria dos trabalhos encontrados na literatura, as soluções são para curto alcance, dependentes de um computador próximo para se obter os dados dos sensores ou de uma fonte de energia externa para energizar o equipamento. Um exemplo é o trabalho proposto por [Bharathkumar et al.\(12\)](#), em que se utiliza um microcontrolador PIC¹ conectado a um sensor de temperatura e umidade com a possibilidade de envio de mensagens de texto para um celular. A solução proposta neste trabalho utiliza um *hardware* mais robusto, com mais sensores e *software* aberto, possibilitando o envio de notificações de acordo com a necessidade do usuário.

Dessa forma, um diferencial da arquitetura proposta neste trabalho é a sua aderência ao modelo de Internet das Coisas, onde os dados dos sensores podem ser acessados através de uma interface *Web* ou de um aparelho celular, sem a necessidade de infraestrutura existente no local de instalação da EstAcqua. Além disso, possui bateria recarregável com 1-2 anos de autonomia ou bateria descartável de até 10 anos de vida útil, utilizando *hardware* de baixo custo, *software* de código aberto e serviços *online* gratuitos para a visualização dos dados coletados e, caso haja alguma indisponibilidade de acesso à Internet, os dados coletados são salvos localmente, evitando a perda de dados.

¹ <https://www.microchip.com/design-centers/microcontrollers>

3 Conceitos de base

Neste capítulo serão apresentados os conceitos chave utilizados na elaboração deste trabalho, cujo conhecimento foi essencial para a criação do protótipo da EstAcqua.

3.0.1 Modelos de Comunicação

Em março de 2015, a *Internet Architecture Board* (IAB) lançou a *Request For Comments* (RFC) 7452 ([13](#), [14](#)), mostrando quatro modelos de comunicação utilizados por dispositivos da Internet das Coisas. A seguir serão apresentados esses modelos e suas respectivas características.

3.0.1.1 Comunicação Dispositivo para Dispositivo

Representa a comunicação de dois ou mais dispositivos entre si sem a necessidade de um *gateway* ou um servidor. Os dispositivos podem se conectar uns aos outros através de diversas redes, utilizando tecnologias como *Bluetooth*, *Wi-Fi*, *Zigbee* ou até mesmo a Internet.

É muito utilizado em sistemas de automação residencial onde dispositivos necessitam de poucos pacotes para se comunicarem entre si. Por exemplo, lâmpadas, interruptores, termostatos e travas de portas. Um exemplo de dois dispositivos de Internet das Coisas se comunicando diretamente entre si pode ser visto na Figura 2.

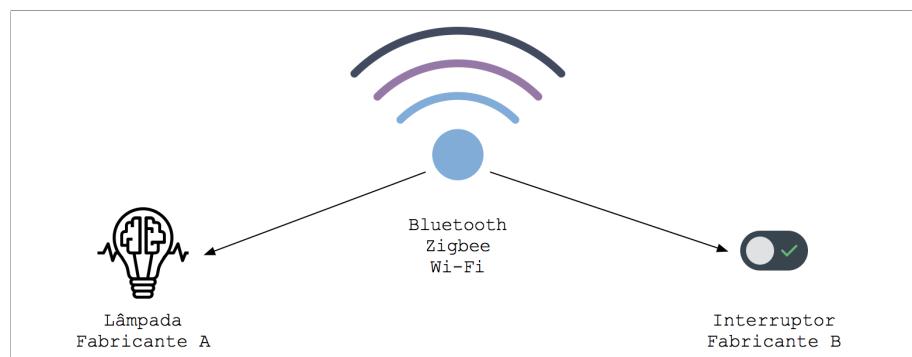


Figura 2 – Exemplo de dois dispositivos da Internet das Coisas usando o modelo de comunicação Dispositivo para Dispositivo

Fonte: Adaptado de [Tschofenig et al.\(13\)](#)

Nela pode-se perceber que a interoperabilidade entre dois dispositivos depende de ambos usarem a mesma tecnologia de transmissão sem fio ([14](#)).

3.0.1.2 Comunicação Dispositivo para Nuvem

Na Figura 3 é mostrado o modelo de comunicação dispositivo para nuvem, onde um dispositivo transmite de forma direta para um serviço de Internet das Coisas situado na nuvem. Normalmente utiliza-se mecanismos de comunicação tradicionais como a rede Ethernet ou o *Wi-Fi*.

É muito utilizado por dispositivos domésticos ou *wearables* (14), como por exemplo sistemas de controle de temperatura, televisões *smart*, *smart watches* etc. Geralmente o serviço na nuvem é do mesmo fabricante do dispositivo. Se o dispositivo utiliza um protocolo proprietário, esse dispositivo irá se comunicar única e exclusivamente com o serviço provido pelo fabricante do mesmo.

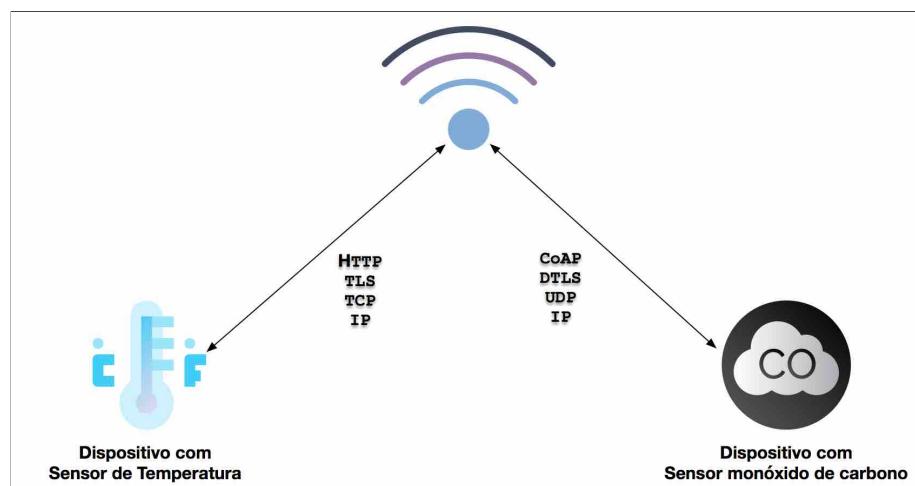


Figura 3 – Exemplo de dois dispositivos da Internet das Coisas usando o modelo de comunicação Dispositivo para Nuvem

Fonte: Adaptado de [Tschofenig et al.\(13\)](#)

3.0.1.3 Comunicação por compartilhamento de dados no *Back-End*

Trata-se de uma arquitetura de comunicação que possibilita ao usuário exportar e analisar dados de dispositivos da Internet das Coisas e combinar com dados de outros dispositivos (14, 13). Esse modelo é demonstrado na Figura 4, onde os dados de diferentes dispositivos enviados para a nuvem podem ser acessados para análise. Por exemplo, em uma empresa que possui diferentes sensores de temperatura conectados utilizando esse tipo de comunicação, possibilitando analisar todas as temperaturas independente do fabricante do sensor. Difere do modelo de comunicação dispositivo para nuvem, pois o dado transmitido não fica somente no serviço na nuvem do fabricante do produto, sendo possível compartilhar o dado coletado com outras aplicações.

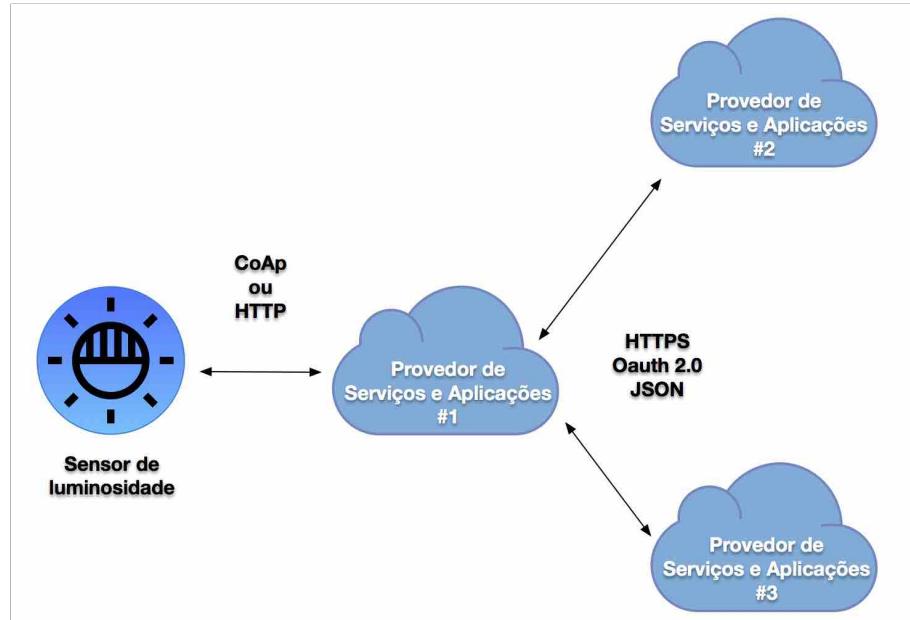


Figura 4 – Exemplo do modelo de comunicação por compartilhamento de dados no *Back-End*

Fonte: Adaptado de [Tschofenig et al.\(13\)](#)

3.0.1.4 Comunicação Dispositivo para *Gateway*

Nesse modelo de comunicação, o dispositivo da Internet das Coisas se conecta a um *gateway* através de comunicação sem fio, como por exemplo *Bluetooth*, *Wi-Fi* ou LoRa, e o *gateway* se comunica com o provedor de serviços e aplicações através da Internet. A Figura 5 mostra a representação desse modelo, onde os dispositivos precisam do *gateway* pois não possuem habilidade nativa de se conectar a um serviço na nuvem (14, 13).

Neste trabalho é utilizado esse modelo de comunicação, onde os dispositivos com os sensores transmitem seus dados via LoRa para um *gateway* e o *gateway* encaminha os dados para um serviço situado na nuvem.

3.1 Low Power Wide Area Network

O termo *Low Power Wide Area Network* (LPWAN) designa redes de comunicação sem fio, projetadas para longo alcance com uma baixa taxa de transmissão, para sensores e microcontroladores alimentados com bateria. Ou seja, são redes de transmissão sem fio para longas distâncias, com baixo consumo de energia, baixa largura de banda e baixo custo de implementação, apagando as lacunas deixadas pelas redes tradicionais, que não foram desenvolvidas para o cenário atual de Internet das Coisas.

Existem duas áreas principais onde as tecnologias LPWAN são mais adequadas (15):

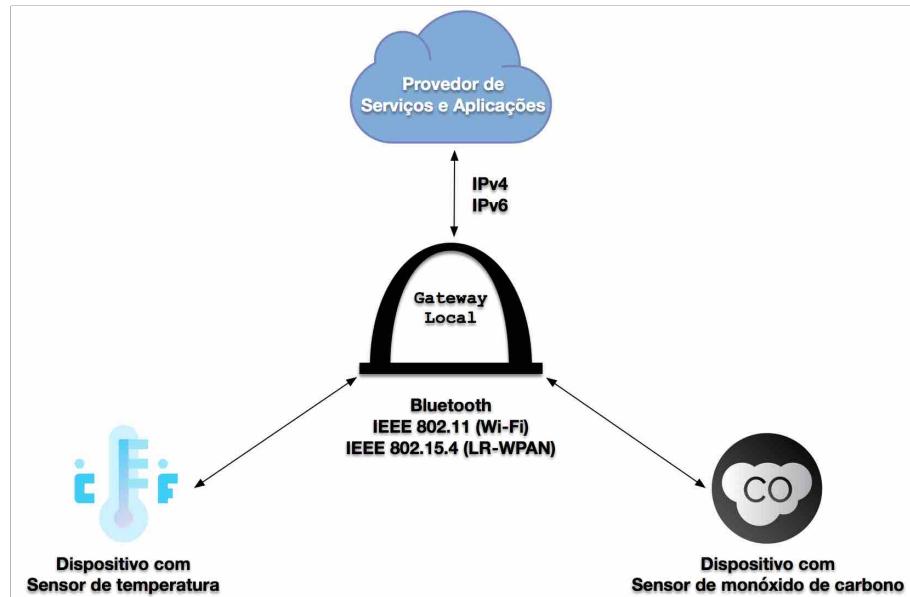


Figura 5 – Exemplo do modelo de comunicação Dispositivo para Gateway

Fonte: Adaptado de [Tschofenig et al.\(13\)](#)

- Lugares com número de conexões fixas de média ou alta densidade, como por exemplo em cidades e edifícios.
- Aplicações alimentadas por bateria com alta autonomia, como por exemplo medição de água em grandes áreas, detectores de gás, agricultura inteligente, fechaduras de portas alimentadas por bateria e pontos de controle de acesso.

Diferentes tecnologias de transmissão de dados sem fio atendem às necessidades específicas de cada aplicação. Aplicações com requisitos de longo alcance e de baixa largura de banda, típicos para aplicações IoT (15), não são bem suportados por essas tecnologias existentes. A Figura 6 mostra a relação entre alcance e largura de banda em LPWAN comparando com outras tecnologias de transmissão sem fio.

Para a escolha de qual tecnologia utilizar na EstAcqua, procuramos uma que atenda aos seguintes requisitos:

- Longo alcance
- Proporcione alta autonomia de bateria
- Baixa largura de banda (poucos bytes são transmitidos)
- Baixo custo de instalação e operação

Como um dos principais aspectos da EstAcqua é a possível instalação em lugares remotos, precisamos de uma tecnologia que consiga longo alcance de transmissão. Dessa

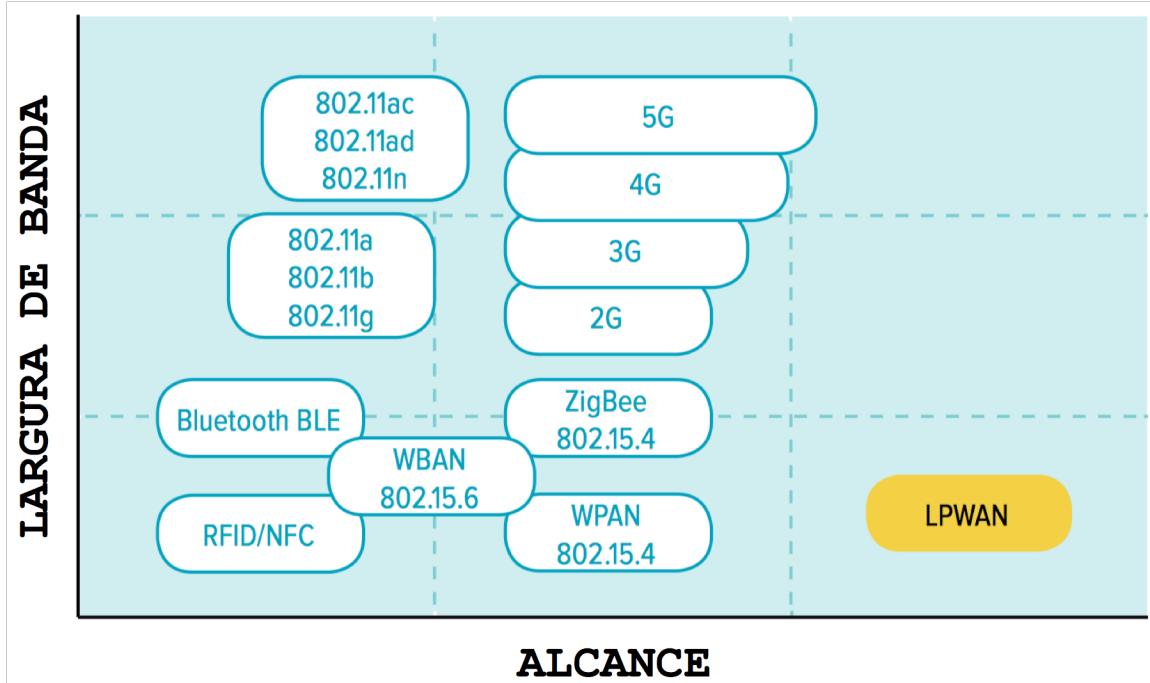


Figura 6 – Comparação entre a largura de banda e o alcance das principais tecnologias de transmissão sem fio

Fonte: Adaptado de [Link-Labs\(15\)](#)

forma, a utilização de *Radio-Frequency IDentification* (RFID) e *Bluetooth* já são descartadas devido ao curíssimo alcance.

A utilização de *Wi-Fi* para transmissão de dados do nó para o *gateway* também deve ser descartada, pois:

- Possui uma alta largura de banda, não necessária para a aplicação
- Possui um alcance maior que o do *Bluetooth* mas ainda assim é bem baixo comparado com quilômetros de distância que procuramos obter
- Opera na banda 2.4GHz, ficando suscetível a interferências devido ao fato dessa banda ser bem populosa.

As redes *Mesh* são uma alternativa de protocolo ao padrão 802.11 para diretrizes de tráfego de dados e voz além das redes a cabo ou sem fio (16). Existem vários tipos de redes *Mesh*, como por exemplo: ZigBee, Z-Wave, IEEE 802.15.4, 6LowPAN etc. Entretanto, todas elas compartilham o mesmo *modus operandi*: usam redes sem fio de alta taxa de dados de curto alcance para construir redes que dependem da retransmissão de dados entre nós (17). Dessa forma, para conseguir longo alcance de transmissão seria necessário a instalação de diversos dispositivos na rede *Mesh*, aumentando o custo, aumentando a complexidade da rede e necessitando de pessoas com conhecimento específico em redes

Mesh para efetuar a instalação. No caso de uma rede *Mesh* com baixa densidade de nós, o alcance será pequeno e a performance da rede será impactada (baixa vazão e não pode fornecer conectividade confiável) (17).

O uso de redes de celular como 3G, 4G e 5G apresentam um custo recorrente, uma vez que deve-se contratar o plano de alguma operadora de celular. Adicionalmente, dispositivos que utilizam essa tecnologia possuem uma baixa autonomia de bateria (17). Outro fator crucial que impede a utilização dessa tecnologia na EstAcqua, é a necessidade de haver cobertura de celular no local onde a EstAcqua será instalada. Em lugares remotos e áreas rurais geralmente não há cobertura de celular, como é o caso do lago Terra Alta (local de instalação da EstAcqua).

Por fim, as redes LPWAN possuem as características que buscamos: alto alcance (podendo chegar a dezenas de quilômetros), *data rate* baixo (transmitindo pacotes de até 250 bytes podendo chegar até a 5 kbps), e, consequentemente, baixo consumo de bateria possibilitando que aplicações possam chegar a até 10 anos de autonomia de bateria.

3.1.1 Tecnologias LPWAN

Uma vez decidido utilizar LPWAN, deve-se escolher qual tecnologia LPWAN utilizar. Para isso foram analisadas as 6 principais tecnologias LPWAN disponíveis no mercado (15): Symphony Link, NWave, Ingenu, Sigfox, Weightless e LoRaWAN.

Symphony Link é a solução LPWAN da Link Labs¹. É uma solução fechada que utiliza a camada física de LoRa mas não utiliza a arquitetura LoRaWAN. Conforme será demonstrado na subseção 3.2.4, LoRaWAN não é uma pilha completa no modelo OSI, com isso o Symphony Link fez sua própria programação da camada MAC e trata desconexões, qualidade de serviço, retransmissão de pacotes etc. Com isso, eles vendem seu próprio *hardware* contendo um chip LoRa e a sua própria camada MAC. Além de obter um hardware fechado, mais caro do adquirir um chip LoRa, deve-se ainda adquirir o software da Symphony Link para funcionar, aumentando o gasto com *hardware* e *software*.

NWave² opera em redes sub-gigahertz e em topologia estrela. Se trata de um protocolo proprietário onde deve-se adquirir o *hardware* da própria empresa para funcionar. Com o fato de ser proprietário, pouco se sabe sobre essa tecnologia além do que o fabricante afirma.

Ingenu³ foge completamente da especificação da EstAcqua. Ele opera em banda 2.4 GHz que, além de ser congestionada, possui um baixo alcance de transmissão de dados. A tecnologia utilizada por Ingenu ainda consome muita energia e não é recomendado para uso em dispositivos energizados através de bateria (15).

¹ <https://www.link-labs.com>

² <https://www.nwave.io>

³ <http://www.ingenu.com>

Sigfox⁴ instala antenas em torres de celulares e vendem o serviço de utilização. Opera em banda sub-gigahertz e cada mensagem possui o tamanho máximo de 12 *bytes*, com taxa de transmissão variando entre 100 e 600 bps. Dessa forma o tempo de ar do pacote é alto e, devido a limitação de 1% da ETSI, resulta em uma capacidade máxima de 6 mensagens de 12 *bytes* por hora (18). Essa tecnologia é adequada para qualquer aplicação que precise enviar poucos e infrequentes dados. Não é recomendado para aplicações que utilizam bateria (15) e o valor a ser pago depende do número e tamanho das mensagens que serão transmitidas por dia.

Weightless⁵ é um padrão aberto criado por uma organização sem fins lucrativos com objetivo de desenvolver padrões abertos e testar tecnologias futuras. Opera em banda sub-gigahertz e, apesar de ser padrão aberto, é necessário adquirir o *hardware* para os nós e o *gateway*. Um kit inicial com dois nós e um *gateway* pode ser adquirido por U\$ 1.500,00 (19).

Por fim, LoRaWAN tem um ecossistema aberto, o que significa que há mais softwares e fornecedores disponíveis. Apesar de necessitar uma nuvem *IoT* para utilizar LoRaWAN, é possível encontrar diversas com serviço gratuito e de qualidade, além de ser mais barato a aquisição de dispositivos LoRa, por exemplo, três microcontroladores LoPy4 (dois operando como nó e um como *nanogateway*) podem ser adquiridos por U\$ 120,00 (20).

Diante disso foi escolhido utilizar LoRa para a transmissão dos dados, por possuir um custo de aquisição baixo, não precisar pagar por serviços de utilização, sejam eles nuvem *IoT* ou por mensagens, e por já possuir implementado uma camada MAC (LoRaWAN), deixando o programador livre para implementar demais funções caso necessário.

3.2 LoRa

LoRa é a camada física de um protocolo *IoT* de longo alcance e baixo consumo de energia (21, 22). A tecnologia foi inicialmente desenvolvida pela empresa francesa Cycleo⁶ e adquirida em 2012 pela Semtech (23). Com o intenso crescimento de redes *Low Power Wide Area Network* (LPWAN) se fez necessário criar um padrão para LoRa que fosse utilizado por todos. Então, em março de 2015, foi criada a *LoRa Alliance* (24). A *LoRa Alliance* é responsável por definir as especificações e padrões da tecnologia LoRa, bem como todo o processo de conformidade e certificação para garantir a interoperabilidade entre os padrões LoRa e os dispositivos.

LoRa é a contração de *Long Range*. Opera em um espectro não licenciado da

⁴ <https://www.sigfox.com>

⁵ <https://http://www.weightless.org>

⁶ <https://www.cykleo.fr/en/>

International Telecommunications Union, nas frequências 433 MHz, 868 MHz e 915 MHz, pertencentes à faixa de frequência para *Industrial Scientific and Medical* (ISM) (25). Por não precisar pagar para usar a frequência, o custo já é reduzido. Além disso, por se tratar de uma tecnologia de longo alcance, são necessários poucos *gateways* para cobrir uma grande área. Como curiosidade, a Bélgica possui uma área de 35,000 km² e com apenas 7 *gateways* é possível oferecer cobertura LoRa em toda a extensão do país (26).

LoRa representa a camada física de uma rede LoRaWAN, responsável por gerenciar a modulação, potência, recepção, transmissão e condicionamento do sinal. Já LoRaWAN define o protocolo de comunicação e a arquitetura do sistema para a rede. A Figura 7 mostra a pilha do protocolo LoRa, onde pode-se ver as atribuições das camadas física e MAC.

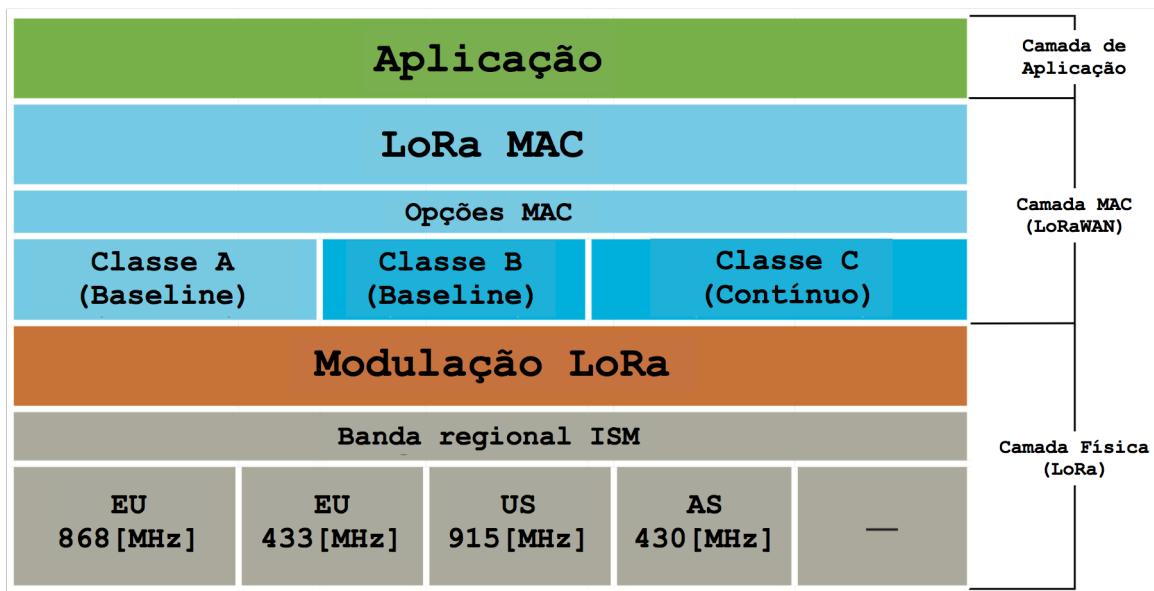


Figura 7 – Pilha do protocolo LoRa

Fonte: Adaptado de [Alliance\(21\)](#)

3.2.1 Modulação LoRa

Diferente de diversos dispositivos de comunicação sem fio que utilizam a modulação *Frequency Shifting Keying* (FSK) para conseguir uma transmissão eficiente e de baixa potência, LoRa utiliza a modulação *Chirp Spread Spectrum* (CSS). CSS, além de necessitar de uma potência baixa para transmissão, ainda aumenta significativamente o alcance da comunicação.

3.2.1.1 Chirp Spread Spectrum

Uma das vantagens do *Chirp Spread Spectrum* se deve ao fato dos deslocamentos de tempo e frequência entre o transmissor e receptor serem equivalentes (27), reduzindo

a complexidade no processo de modulação e demodulação. O processo de modulação, utilizado em LoRa, funciona movendo um tom de Rádio Frequência através do tempo de uma forma linear. Dessa forma a frequência instantânea $f(t)$ varia linearmente com o tempo (27):

3.2.1.2 Spreading Factor, Code Ratio e Bandwidth

Antes de aprofundar nos cálculos de taxa de transmissão e tempo para transmitir um símbolo, é necessário entender alguns conceitos:

- *Spreading Factor (SF)*: ou fator de espalhamento, é o número de *chirps* utilizado por símbolo de dados (28). Quanto menor o SF, maior a taxa de transmissão. Por outro lado, um SF maior garante uma maior robustez na transmissão com maior alcance, entretanto ocorre um aumento na possibilidade de colisão devido ao aumento de tempo de ar⁷ do pacote (28, 29).
- *Code Rate (CR)*: a proporção de dados não redundantes, ou seja, de informação útil (30). Um *Code Rate* de $\frac{k}{n}$ significa que para cada k bits de informação útil, são gerados n bits de dados, dos quais $n - k$ são redundantes.
- *Bandwidth (BW)*: a quantidade máxima de dados que podem ser transferidos através de uma rede em um determinado período.

LoRa possui 6 diferentes possibilidades de escolha para o *Spreading Factor*: $SF = \{7, 8, 9, 10, 11, 12\}$, quatro possibilidades para o *Code Rate*: $CR = \{\frac{4}{5}, \frac{4}{6}, \frac{4}{7}, \frac{4}{8}\}$ e três diferentes *Bandwidth*: $BW = \{125 \text{ kHz}, 250 \text{ kHz} \text{ e } 500 \text{ kHz}\}$. Com isso podemos calcular a duração da transmissão de um símbolo (29) usando a Fórmula 3.1:

$$T_s = \frac{2^{SF}}{BW} \quad (3.1)$$

Analisando a Fórmula 3.1, pode-se perceber:

- Para uma largura de banda fixa, aumentar o *Spreading Factor* em uma unidade resulta em um tempo de transmissão duas vezes maior. Da mesma forma, reduzir em uma unidade o *Spreading Factor* resulta em um tempo de transmissão duas vezes menor. A Figura 8 demonstra essa situação, onde ao aumentar o SF em uma unidade dobrou-se o tempo de transmissão.
- Para obter o menor valor de T_s , deve-se utilizar o menor *Spreading Factor* ($SF = 7$) e o maior *Bandwidth* ($BW = 500 \text{ kHz}$). Analogamente para se obter o maior

⁷ Tempo necessário para que a antena transmita o pacote

valor de T_s deve-se utilizar o maior *Spreading Factor* ($SF = 12$) e o menor valor de *Bandwidth* ($BW = 125$ kHz).

A Figura 8 esclarece o significado da Fórmula 3.1: o tempo necessário para varrer da frequência f_1 à frequência f_2 . Esse é o tempo de símbolo (21) calculado em T_s .

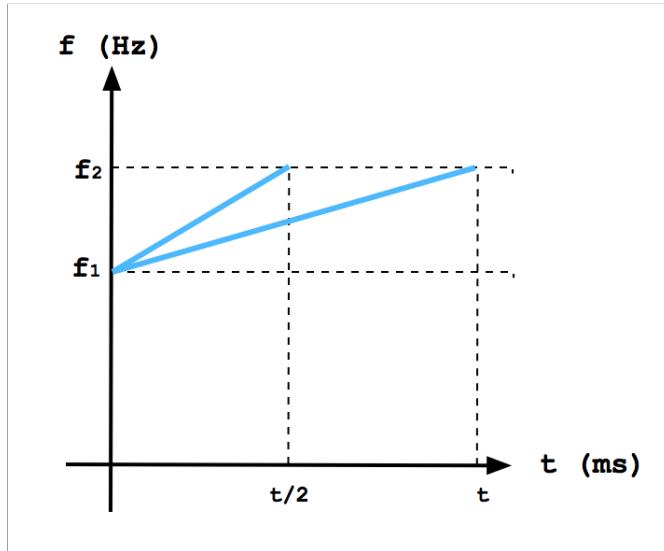


Figura 8 – Tempo necessário ao incrementar em uma unidade o SF, mantendo o BW fixo

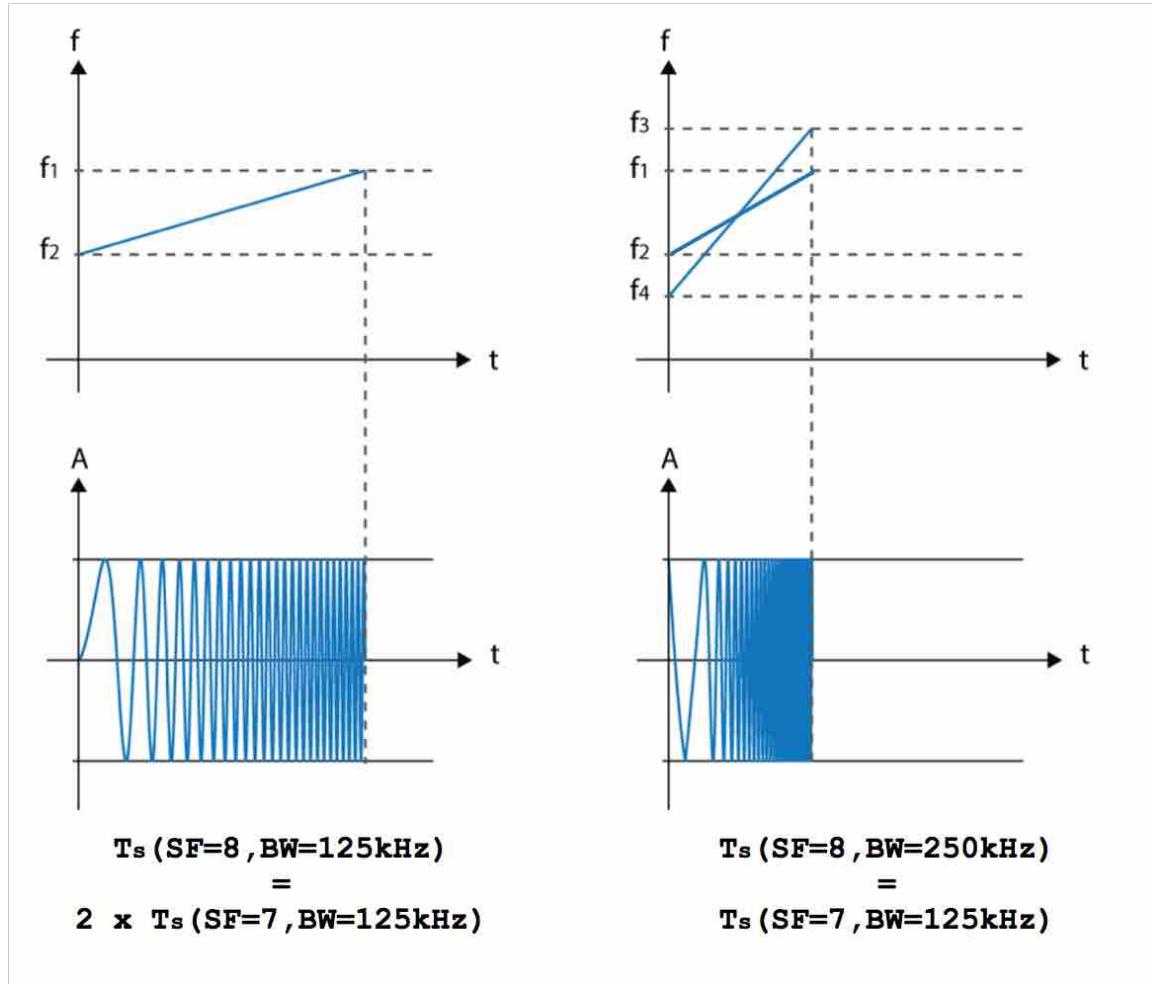
Outro fator importante sobre o T_s é a possibilidade de conseguir o mesmo tempo utilizando SF e BW diferentes. A Figura 9 mostra, à esquerda, que aumentando em uma unidade o SF, o tempo necessário para a varrer da frequência f_1 para f_2 é duas vezes maior. E também mostra uma situação muito importante, onde o T_s em duas situações com SF e BW diferentes, possui o mesmo valor.

Essa é uma característica bem marcante do LoRa. Os quadros (31) são ortogonais, o que possibilita o envio de vários quadros ao mesmo tempo (26). Isso possibilita que o receptor, desde que o *Spreading Factor* seja diferente, detecte os pacotes recebidos, mesmo que possuam a mesma frequência e largura de banda (29). Isso permite que redes utilizando LoRa aumentem sua vazão (32).

A taxa de *bits* transmitidos por segundo, R_b , é calculada em função do SF e BW selecionados (26):

$$R_b = SF \times \frac{BW}{2^{SF}} \quad [bps] \quad (3.2)$$

Pela Fórmula 3.2 percebe-se que a menor taxa de *bits* transmitidos ocorre quando se utiliza o maior SF com a menor BW. Analogamente, a maior taxa é alcançada com o menor SF e maior BW. A Tabela 1 mostra a taxa de *bits*, em *bits* por segundo (bps), para cada combinação de SF e BW.

Figura 9 – Relação entre *Spreading Factor*, *Bandwidth* e T_s

Fonte: Adaptado de Ruano(22)

Tabela 1 – Taxa de *bits* transmitidos por segundo em função de SF e BW

SF	125 kHz	250 kHz	500 kHz
7	6835	13671	27343
8	3906	7812	15625
9	2197	4396	8793
10	1220	2441	4882
11	671	1342	2685
12	366	732	1464

Fonte: Semtech(23)

Se estiver utilizando algum *Code Rate*, deve-se modificar a Fórmula 3.2 para incluir o parâmetro:

$$R_b = SF \times \frac{BW}{2^{SF}} \times CR \quad [bps], \quad CR \in \left\{ \frac{4}{5}, \frac{4}{6}, \frac{4}{7}, \frac{4}{8} \right\} \quad (3.3)$$

Pela Fórmula 3.3 pode-se perceber a relação de compromisso na escolha do CR. Um CR maior aumenta a robustez da comunicação, mas em contra partida reduz a taxa de *bits* transmitidos. Para $CR = \frac{4}{8}$, a taxa de transmissão de *bits* por segundo é reduzida pela metade.

- Tempo de ar do pacote

Para o cálculo do tempo de ar (*air time*, em inglês) do pacote, t_{pacote} , utiliza-se a seguinte fórmula (33):

$$t_{pacote} = t_{preambulo} + t_{payload}, \quad (3.4)$$

onde $t_{preambulo}$ é o tempo necessário para enviar o preâmbulo do pacote e $t_{payload}$ o tempo necessário para transmitir o dado.

Esses dois tempos podem ser calculados conforme as fórmulas abaixo:

$$t_{preambulo} = (n_{preambulo} + 4.25) \times T_s \quad (3.5)$$

onde $n_{preambulo}$ é o número de símbolos de preâmbulo programados e T_s é o tempo de símbolo demonstrado na Fórmula 3.1. Para o cálculo de $t_{payload}$:

$$t_{payload} = n_{payload} \times T_s \quad (3.6)$$

sendo $n_{payload}$ o número de símbolos que compõem o *payload* e o cabeçalho do pacote (33), podendo ser calculado pela Fórmula 3.7:

$$n_{payload} = 8 + \max\left(\frac{8 \times PL - 4 \times SF + 44 - 20 \times H}{4 \times SF - 8 \times DE} \times (CR + 4), 0\right), \quad (3.7)$$

onde:

- **PL:** número de *Bytes* do *payload*
- **SF:** *Spreading Factor* utilizado
- **H:** utilizado para indicar a presença de cabeçalho, sendo $H = 0$ a ausência de cabeçalho e, $H = 1$ quando o cabeçalho está presente
- **DE:** indica quando a otimização de baixa taxa de dados está ativada, $DE = 1$, ou desativada, $DE = 0$

- **CR:** *Code Rate* utilizado, sendo 1, 2, 3 ou 4

O número máximo de *Bytes* que podem ser enviados por pacote depende do *Data Rate* utilizado (34). O *Data Rate* (DR) é definido de acordo com o SF e BW escolhidos e depende da faixa de frequência utilizada. A Tabela 2 mostra o *Data Rate* e o tamanho máximo do *payload* para a faixa de 902 a 928 MHz. Essa faixa pertence a banda US915, utilizada e regulada pelos Estados Unidos da América, e foi a homologada pela ANATEL para uso no Brasil (35).

Tabela 2 – Tamanho máximo do *payload*, em *Bytes*, por *Data Rate*

DR	SF	BW	Payload
0	10	125 kHz	11
1	9	125 kHz	53
2	8	125 kHz	129
3	7	125 kHz	242
4	8	500 kHz	242
8	12	500 kHz	33
9	11	500 kHz	109
10	10	500 kHz	222
11	9	500 kHz	222
12	8	500 kHz	222
13	7	500 kHz	222

Fonte: [Semtech\(23\)](#)

Os *Data Rates* 5, 6, 7 e 14 são reservados para uso futuro. Como o presente trabalho utiliza a banda US915 (902-928 MHz), apenas os dados referentes a essa banda serão detalhados. As demais bandas (EU863-870, CN779-787, EU433, AU915-928, CN470-510, AS923, KR920-923 e INDIA865-867) podem ser consultadas na documentação dos parâmetros regionais LoRaWAN (34).

As redes LoRaWAN possuem um *Adaptive Data Rate* (ADR), o que permite que o *Data Rate* de um dispositivo transmitindo LoRa seja dinamicamente modificado. Os dispositivos que estão próximos ao receptor podem ser configurados para uma taxa de dados mais alta devido à qualidade do sinal. Se o receptor estiver recebendo um sinal de alta qualidade, ele pode solicitar que o transmissor aumente seu *Data Rate*. A mesma situação é possível no caso de um sinal de baixa qualidade, sendo necessário diminuir o *Data Rate*.

3.2.2 Frequência

Até o momento da escrita deste trabalho, apenas as bandas EU868 e US915 possuem um ato regularizador (21) pelas agências técnicas responsáveis, sendo a *European*

*Telecommunications Standards Institute*⁸ (ETSI) responsável pela EU868 e a *Federal Communications Commission*⁹ (FCC) pela US915.

A Tabela 3 mostra um resumo das principais características das bandas EU868 e US915, utilizadas na Europa e Estados Unidos da América (EUA), respectivamente (21).

Tabela 3 – Comparativo das limitações LoRa na Europa e Estados Unidos da América

	Europa	EUA
Frequência	867-869 MHz	902-928 MHz
Canais	10	64 + 8 + 8
BW Uplink	125 ou 250 kHz	125 ou 500 kHz
BW Downlink	125 kHz	500 kHz
Potência TX Uplink	+14 dBm	+20 dBm
Potência TX Downlink	+14 dBm	+27 dBm
Spreading Factor	7-12	7-12
Taxa de bits/s	250 bps a 50 kbps	980 bps a 21,9 kbps
Link Budget	155 dB	154 dB

Fonte: Semtech(23)

Um fator importante é o *Duty Cycle*, que será apresentado a seguir.

- ***Duty Cycle***

É a fração de um período no qual um sinal ou sistema está ativo sobre o tempo total. O *Duty Cycle* é expresso em porcentagem ou uma razão (28). Pode ser calculado da seguinte maneira:

$$DC = \frac{\tau}{T} \times 100 \quad \{ \%\} \quad (3.8)$$

$$DC = \frac{\tau}{T}, \quad DC \leq 1.0 \quad (3.9)$$

sendo T o tempo de permanência do sinal e τ o tempo ativo do sinal.

- **EU868**

São definidos dez canais, dos quais oito utilizam a modulação CSS, podendo obter taxa de transferência de *bits* de 250 bps a 5,5 kbps ($CR = \frac{4}{5}$, SF = 7, BW = 125 kHz). E possui dois canais especiais, sendo um deles de alta taxa de dados com taxa máxima de transmissão de *bits* de 11 kbps e um único canal especial, que utiliza a modulação *Frequency-Shift Keying* (36, 21) com taxa máxima de 50 kbps.

⁸ <https://www.etsi.org>

⁹ <https://www.fcc.gov>

A potência máxima de saída (EIRP) permitida pelo órgão regulamentador ETSI, para os dispositivos que operam nessa faixa de frequência na Europa, é +14 dBm. Apesar de não existirem limites definidos para a taxa de transmissão máxima ou de tempo de permanência no canal, a ETSI restringe o *Duty Cycle* entre 0,1%, 1,0% e 10% (37).

- **US915**

A banda ISM para a América do Norte, aprovada pela FCC, é de 902-928 MHz (38), muito maior que a banda 867-869 MHz definida para a Europa. Consequentemente são definidos mais canais, sendo 64 canais de *uplink* de 125 kHz de 902,3 MHz a 914,9 MHz em com incrementos de 200 kHz. Além desses, existem 8 canais de *uplink* com BW de 500 kHz e incrementos de 1,6 MHz das frequências 903 MHz à 914,9 MHz. Para *downlink* são reservados 8 canais de 500 kHz de largura de banda nas frequências de 923,3 MHz até 927,5 MHz, conforme demonstrado na Figura 10.

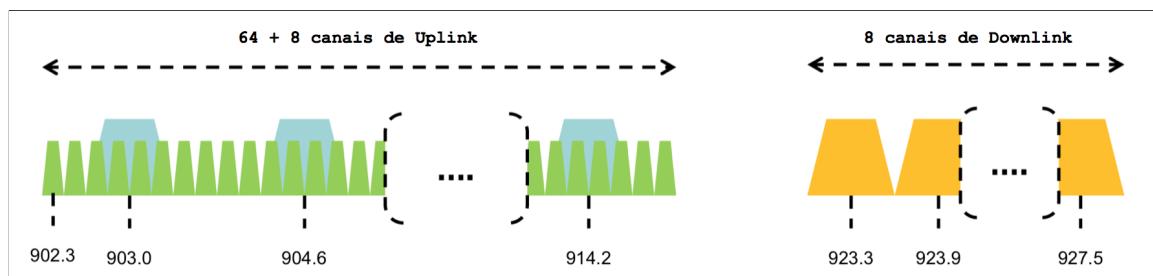


Figura 10 – Faixa de frequência 902-915 MHz demonstrando os canais de *uplink* e *downlink* juntamente com as larguras de banda de 200 kHz e 1,6 MHz

Fonte: Alliance(21)

A FCC não impõe limitações no *Duty Cycle* mas limita a potência máxima em +20 dBm (suficiente para a larga maioria dos dispositivos), com a possibilidade de aumentar até +30 dBm caso seja realmente necessário, além de limitar o tempo máximo de permanência no canal em 400 ms.

Apesar da ANATEL ter regulamentado a faixa 902-928 MHz para uso no Brasil, ela não funciona da mesma maneira que a faixa 902-928 MHz regulamentada pela FCC. Isso se deve ao fato da FCC disponibilizar toda a faixa entre 902 e 928MHz para uso (38) e essa mesma faixa não está completamente disponível no Brasil, pois ela é utilizada para GSM900¹⁰ (39). A Resolução 454 da ANATEL (40) limita a 400 ms de *air time* a cada 10-20 s, e trata a faixa de 902 a 907,5 MHz como uma faixa ISM de radiação restrita, de 907,5 a 910 MHz como subfaixa de extensão, de 910 MHz a 912,5 MHz como subfaixa D e 912,5 MHz a 915 MHz como subfaixa E (41). Como há essa restrição na faixa de

¹⁰ https://en.wikipedia.org/wiki/GSM_frequency_bands

907,5 MHz até 915 MHz e para essa faixa de frequência a largura de banda é de 200 kHz, podemos calcular a quantidade de canais, C, que não podem ser utilizados:

$$C = \frac{(915 - 907,5) \times 10^6}{200 \times 10^3} = \frac{7,5 \times 10^6}{200 \times 10^3} = \frac{7500}{200} = 37,5 \quad (3.10)$$

Ou seja, dos 64 canais existentes, definidos pela FCC, na faixa 902-928 MHz, apenas 26 podem ser utilizados no Brasil (40). Essa situação representa um grande problema na universalização dos padrões, pois um dispositivo projetado para funcionar nessa faixa de frequência (902-928 MHz) causará uma grande dificuldade aos fabricantes que precisam adequar os seus produtos para atender este requisito no Brasil. Como essa restrição da faixa foi baseada na regulamentação da *FCC Part 15* (42) que posteriormente mudou para liberar toda a faixa 902-915 MHz, já foi realizada uma consulta externa com propostas de adequação técnica para essa faixa no Brasil (43). Outros países passam por problemas semelhantes com a faixa de 902-928 MHz. A Austrália, por exemplo, utiliza essa mesma faixa, mas de 902-915 MHz é reservada para telefonia móvel (44).

3.2.3 LoRaWAN

LoRaWAN define o protocolo de comunicação e a arquitetura do sistema para uma rede LoRa. Como o protocolo e a arquitetura de rede têm mais influência na determinação na vida útil da bateria de um dispositivo, na capacidade da rede, na qualidade do serviço e na segurança, LoRaWAN desempenha um papel fundamental.

Muitas redes implantadas utilizam uma arquitetura de rede em malha¹¹ em que os nós (45) encaminham as informações de outros nós para aumentar o alcance da rede. Em contrapartida também se aumenta a complexidade e reduz o tempo de vida da bateria, a medida que os nós recebem e encaminham informações de outros nós que provavelmente são irrelevantes para eles. A arquitetura utilizada em LoRaWAN é estrela (46), suportando também a topologia estrela de estrela (26), com um longo alcance e preservando a vida útil da bateria uma vez que os nós não precisam, necessariamente, repassar informações de outros nós.

Conforme demonstrado na Figura 11, os nós não estão associados a um *gateway* específico. Em vez disso, os dados transmitidos por um nó geralmente são recebidos por vários *gateways*. Os *gateways* então encaminham o pacote recebido para o servidor na nuvem utilizando 4G, *Wi-Fi* etc. O servidor na nuvem fica responsável pela inteligência da rede. Se um nó for móvel ou estiver em movimento, não é necessário *handover*¹² entre *gateways*, o que é um recurso essencial para aplicações de rastreamento.

¹¹ https://en.wikipedia.org/wiki/Mesh_networking

¹² <https://en.wikipedia.org/wiki/Handover>

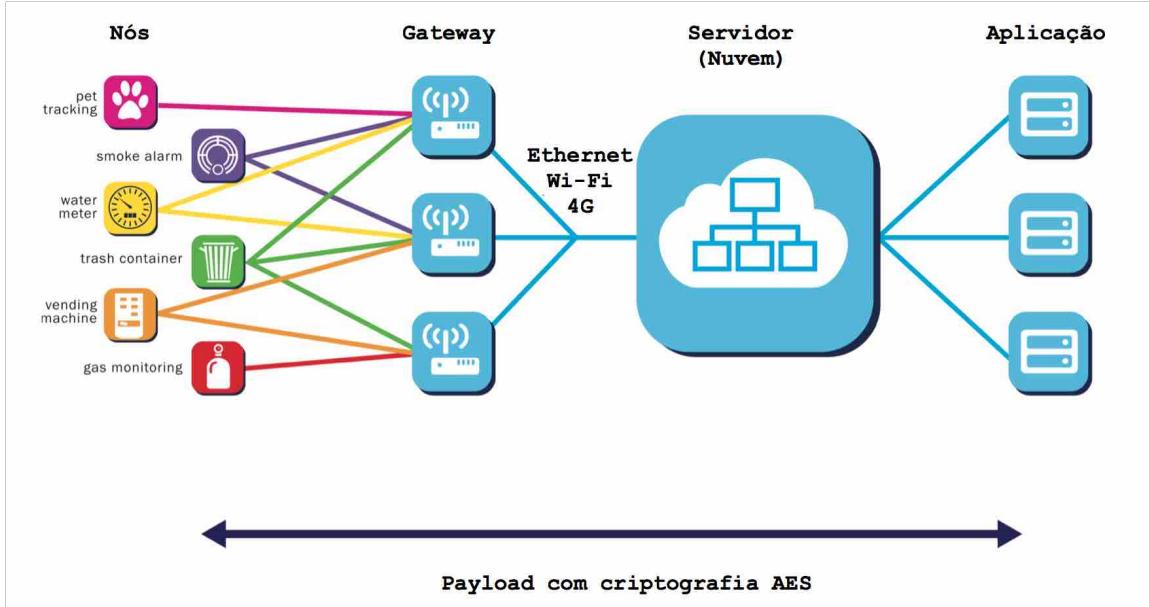


Figura 11 – Típica topologia de rede LoRaWAN

Fonte: Adaptado de [Alliance\(21\)](#)

Outro fato importante mostrado na Figura 11 é o fato de que os dados do usuário são transportados do nó para o *gateway* através do protocolo LoRaWAN. O *gateway* por sua vez encaminhará o pacote para um serviço de rede LoRaWAN dedicado na nuvem. Isso é uma característica única do LoRaWAN, uma vez que a maioria das outras arquiteturas de *Wide Area Network* (WAN) não tem controle algum do dado transmitido para a Internet uma vez que ele sai da rede local.

Os nós em uma rede LoRaWAN se comportam como em uma rede ALOHA (47), de forma assíncrona e transmitindo sempre quando há dados para enviar. Isso implica que, caso haja um mal gerenciamento dos nós, a taxa de colisão seja elevada (22, 21). Por outro lado, a não necessidade de sincronizar com os outros dispositivos da rede proporciona um menor consumo de bateria - um dos principais requisitos da Internet das Coisas(26).

3.2.3.1 Segurança

Para toda *Low Power Wide Area Network* (LPWAN) é extremamente importante incorporar segurança. O LoRaWAN utiliza duas camadas de segurança: uma para a rede e outra para a aplicação (21). A segurança de rede garante a autenticidade de um nó na rede, enquanto a segurança da aplicação garante que a o servidor na nuvem não tenha acesso aos dados do usuário. Conforme demonstrado na Figura 11, desde o início da transmissão de um dado na rede é utilizada criptografia *Advanced Encryption Standard* (AES), passando por toda a rede LoRa, chegando até a aplicação, onde ele é descriptografado (23, 21).

A segurança de rede usa uma chave (NwkSKey) para autenticar um dispositivo

na rede e uma chave separada (AppSKey) para os dados transmitidos para a aplicação. Para ingressar em uma rede LoRa, o dispositivo envia uma solicitação *JOIN*¹³ e o *gateway* responderá com um endereço de dispositivo e um *token* de autenticação. Nesse procedimento, o nó envia as chaves NwkSKey e AppSKey. Esse método é chamado de *Over-The-Air Activation* (OTAA). Outro método que também pode ser utilizado é o *Activation By Personalization* (ABP), onde a chave de autenticação é registrada no servidor na nuvem e depois é programada no dispositivo que irá se conectar à rede.

Não existe um método ideal de autenticação e a escolha entre ABP ou OTAA depende do dispositivo que está sendo utilizado, método de operação, etc. A Tabela 4 faz um comparativo dos prós e contras entre ABP e OTAA.

Tabela 4 – Comparativo entre os métodos de autenticação OTAA e ABP

	OTAA	ABP
Prós	Chaves de sessão são geradas quando necessárias, impossibilitando que uma chave seja comprometida antes da ativação.	O dispositivo não precisa ser capaz de realizar um procedimento de <i>JOIN</i> .
	Se o dispositivo mudar de rede, ele pode se autenticar na nova rede gerando novas chaves, sem a necessidade de ser reprogramado.	O dispositivo não precisa decidir a hora de solicitar um <i>JOIN</i> , pois nunca é necessário.
Contras	É necessário pré-programar cada dispositivo com uma chave de identificação do dispositivo (DevEUI), da aplicação (AppKey), além do AppEUI correto.	O esquema para gerar o NwkSKey e o AppSKey deve garantir que eles sejam exclusivos, para evitar que a rede possa ser violado caso um único dispositivo seja comprometido. Se o dispositivo for comprometido a qualquer momento, mesmo antes da ativação, a chave pode ser descoberta.
	O dispositivo deve suportar a função <i>JOIN</i> e armazenar chaves geradas dinamicamente. Nem todos os dispositivos possuem essa capacidade, em especial dispositivos mais simples com menor preço.	As configurações de rede não podem ser especificadas no momento em que o dispositivo se une à rede. E caso o dispositivo seja movido para outra rede, ou a chave utilizada por ele seja revogada, deve-se reprogramar o dispositivo com a nova chave.

3.2.3.2 Classes dos Dispositivos

Como os nós em uma rede LoRa servem para diversas aplicações e possuem requisitos diferentes, LoRaWAN os categoriza em 3 classes para otimizar cada nó em sua

¹³ Solicitação para que possa se conectar a rede

respectiva função. A Figura 12 mostra as 3 classes; A, B, C. Nela é possível perceber que a classe do dispositivo é uma relação de compromisso entre o consumo de bateria e a latência para receber um *downlink*.

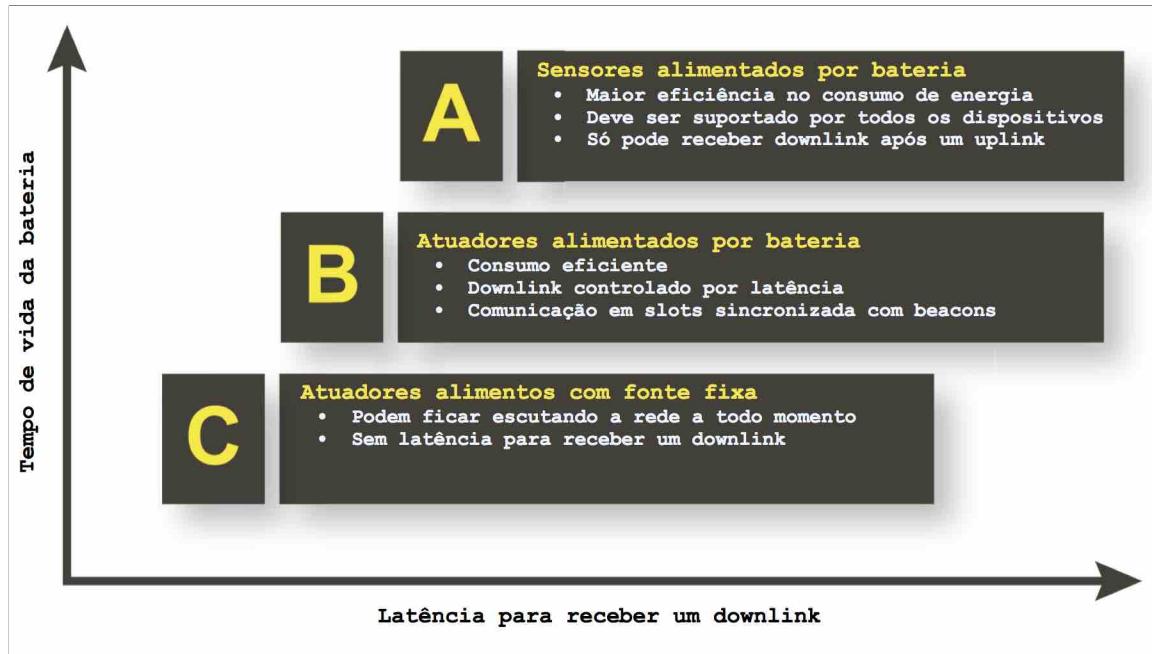


Figura 12 – Classes de dispositivos que operam em uma rede LoRaWAN

Fonte: Adaptado de [Alliance\(21\)](#)

- **Classe A**

Dispositivos pertencentes à classe A são sensores e terminais alimentados por bateria. Todos os dispositivos que se autenticam em uma rede LoRaWAN são primeiro associados como Classe A, com a possibilidade de mudar para outra classe durante a operação. Dispositivos operando na Classe A, após realizarem um *uplink*, abrem duas janelas de *downlink* com 1 s cada. Dessa forma possuem o menor consumo de energia, porque o nó está sempre dormindo e quando acordar para transmitir um dado, serão abertas as janelas para recepção de *downlink*. O servidor pode responder na primeira janela de *downlink* ou na segunda janela de *downlink*, mas não deve usar as duas janelas. Se o servidor não responder em nenhuma dessas janelas de *downlink*, a próxima oportunidade será após o próximo *uplink* do nó.

- **Classe B**

Além de possuir janelas de recebimento de *downlink* iguais aos dispositivos da Classe A, os dispositivos de Classe B abrem janelas de recebimento extra em horários programados. Para que o dispositivo final abra sua janela de recebimento no horário

programado, ele recebe um sinal sincronizado pelo *gateway*. Os dispositivos da Classe B equilibram o consumo da bateria e a latência de recebimento de *downlink*.

- **Classe C**

Os dispositivos dessa classe usam o máximo de energia, mas têm a menor latência. São abertas duas janelas de recepção de *downlink* após efetuar um *uplink*, igual na Classe A, mas mantém uma janela de recepção contínua. Ou seja, dispositivos da Classe C mantêm quase que continuamente as janelas de *downlink* abertas, sendo fechadas somente durante a transmissão.

3.2.4 Discussão

Apesar da fácil escalabilidade e capacidade de uma rede LoRaWAN, ela possui lacunas em sua arquitetura e protocolos. Algumas são pequenos detalhes mas existem lacunas que requerem uma atenção especial:

- LoRaWAN não é uma pilha de rede completa no modelo *Open System Interconnection* (OSI). Por não ter algumas camadas, faltam recursos comuns da camada de rede, camada de sessão e camada de transporte. Dessa forma fica a cargo do desenvolvedor adicionar serviços como compactações, mecanismos de repetição e *Quality of Service* (QoS), se necessário.
- LoRaWAN tem seu funcionamento baseado em um servidor na nuvem. Apesar de existirem diversos servidores gratuitos, pode ser que em algum momento seja necessário assinar um plano para utilizar esses servidores. Alternativamente, pode-se implementar um servidor LoRaWAN próprio, local ou remoto.
- Existem poucos fornecedor de *chips* LoRa além da Semtech. Foi anunciada uma parceria com a empresa ST Microelectronics para que ela também possa fabricar os *chips* (48), além de outros fabricantes como Microchip, NXP, Murata. Por se tratarem de poucos fornecedores, eles podem dominar o mercado aumentando o preço, ou simplesmente descontinuando a fabricação.
- O LoRaWAN é baseado no protocolo ALOHA, que não é bom para vários dispositivos transmitindo no mesmo canal. O índice de colisão pode ser alto caso os nós LoRaWAN não tenham configuração adequada, cabendo ao programador tentar mitigar ao máximo as colisões.
- Recebimento de *downlink* limitado comparado a *uplink*.

- O LoRaWAN possui alta latência e, consequentemente, não possui capacidade para aplicações em tempo real. Por exemplo, não se deve deixar o controle de uma válvula de alívio de pressão para ser controlada via LoRa.
- Limitações em fazer atualizações *Over The Air* (OTA) (49), ou seja, para atualizar *firmware* de dispositivos LoRa deve-se retirá-lo do local, parando suas atividades, e conectá-lo a um computador para que seja realizada a atualização do *firmware*.
- Nós em movimento em alta velocidade ainda são um desafio para LoRaWAN. Dependendo da situação uma mensagem de 50 *Bytes* pode demorar quase 2 s para ser transmitida. Com isso, aplicações em veículos em alta velocidade, por exemplo, são bem problemáticas para se tratar com LoRaWAN.

3.3 Microcontroladores

Um microcontrolador é um computador pequeno e independente que está alojado em um único circuito integrado. São comumente utilizados em projetos de Internet das Coisas (50) devido a:

- **Simplicidade:** microcontroladores não precisam de sistemas operacionais para funcionar e são fáceis de interagir com dispositivos externos, como sensores e atuadores. Sua falta de dependências externas também os torna fáceis de configurar. Basta ligá-los, carregar o *firmware* e pronto. Além disso, a codificação necessária para programar um microcontrolador é mínima.
- **Segurança:** Em virtude de sua relativa simplicidade, os microcontroladores oferecem menos meios de serem atacados. Como regra, cada porta aberta e protocolo disponível também é uma vulnerabilidade potencial.
- **Custo:** Na maioria dos aplicativos de IoT atuais, um microcontrolador pode fornecer todo o poder de processamento e funcionalidade necessário. Como resultado, os microcontroladores são, na maioria das vezes, a melhor e mais econômica opção de hardware para aplicativos *IoT*. No geral, eles oferecem funcionalidade simples e segura por um pequeno custo.

Também pode ser utilizado *System On Chip* (SOC) (51) em projetos de *IoT*, como por exemplo o Raspberry Pi¹⁴, DragonBoard¹⁵ ou Beagle Bone¹⁶. Entretanto, para o projeto da EstAcqua, esse tipo de equipamento foi descartado pelos seguintes motivos:

¹⁴ <https://www.raspberrypi.org>

¹⁵ <https://developer.qualcomm.com/hardware/dragonboard-410c>

¹⁶ <https://beagleboard.org/>

- Alto consumo de energia. Por se tratar de um computador com sistema operacional, não são projetados para funcionar com bateria
- Alto custo. Com preços entre U\$35,00 e U\$65,00 apenas pelo SOC, além da necessidade de aquisição de *shield* LoRa
- Como o nó funciona em modo *deep sleep*, não seria possível conseguir economia de energia ligando e desligando o equipamento repetidas vezes.

3.3.1 Comparativo

Pode-se caracterizar dispositivos *IoT* em termos dos seguintes recursos (52):

- **Aquisição e controle de dados:** é o processo de medir condições do mundo real e converter essas medidas em leituras digitais em intervalos de tempo fixo (taxa de amostragem de dados).
- **Processamento e armazenamento de dados:** os dispositivos *IoT* exigem recursos de processamento e armazenamento de dados para tratar os dados coletados, podendo processar localmente ou transmitir esses dados para outros dispositivos ou aplicações.
- **Conectividade:** a conectividade de rede é uma das características que definem qualquer dispositivo *IoT*.
- **Gerenciamento de energia:** o gerenciamento de energia é uma preocupação especial para dispositivos de *IoT* portáteis.

Serão analisados os dispositivos abordados no trabalho de Gerber(52) com o objetivo de verificar qual dispositivo melhor atende os requisitos da EstAcqua. O Quadro 1 mostra o comparativo entre Arduino Uno, Pyboard¹⁷ e LoPy4.

Apesar do Arduino Uno e Pyboard não possuírem interface de rede LoRa, é possível adquirir *shield* ou um *breakout circuit*¹⁸. Por exemplo, o Dragino LoRa *Shield* (56) para Arduino pode ser adquirido por U\$ 21,00 (57). Para o Pyboard encontra-se disponível no mercado o Adafruit RFM95W por U\$ 20,00 (58). Combinando o valor do Arduino Uno com o *shield* LoRa, tem um custo de U\$ 43,00, e o Pyboard com o RFM95W em U\$ 56,00.

A escolha em utilizar o microcontrolador LoPy4 pelos seguintes motivos:

¹⁷ <https://store.micropython.org/product/PYBv1.1>

¹⁸ São placas que contém um componente único que devido às suas dimensões muito pequenas não podem ser ligados diretamente a uma placa de microcontrolador pelas suas saídas e por isso eles são "quebrados", ou expandidos, de modo a se ter acesso aos seus terminais.

Quadro 1 – Comparativo entre microcontroladores

	Arduino Uno	Pyboard	LoPy4
Processador	ATMega328PU	Cortex M4	32-bit Xtensa LX6
Frequência	16 KHz	168 MHz	240 MHz
Armazenamento	32 KB	1 MB	8 MB
RAM	2 KB	192 KB	4 MB
Rede	-	-	<i>Bluetooth, Wi-Fi, Sigfox e LoRa</i>
Ativo	13,9 mAh	16 mAh	35,4 mAh
Dormindo	6,2 μ Ah	6 μ Ah	18,5 μ Ah
Linguagem	C ou C++	MicroPython	U\$ MicroPython
Preço	U\$ 22,00	U\$ 35,00	U\$ 40,00

Fonte: Adaptado de [Arduino](#); [Pyboard](#); [Pycom](#)([53](#), [54](#), [55](#))

- **Baixo custo:** comparado com os demais que necessitam de um *shield* para transmitir via LoRa.
- **Memória:** possui uma memória de armazenamento interna de 8 MB e 4 MB de memória RAM.
- **LoRa:** o fato de possuir chip LoRa já integrado não necessitando de equipamentos externos.
- **Linguagem:** possui MicroPython embarcado. Conforme visto anteriormente, MicroPython é uma linguagem desenvolvida especificamente para microcontroladores com desempenho melhor que C.

3.4 *MicroPython*

O *MicroPython* foi criado por Damien George, um programador australiano. Trata-se de uma implementação da linguagem de programação *Python 3*¹⁹, escrita na linguagem C²⁰, otimizada para microcontroladores. Possui um *Read-Eval-Print Loop* (REPL)²¹ para execução imediata de comandos e é possível utilizar *MicroPython* em dezenas de placas microcontroladoras, dentre elas Pyboard, ESP8266²², ESP32²³, BBC Micro Bit²⁴, WiPy, LoPy, OpenMV Cam M7, placas Adafruit com CircuitPython (*fork* de *MicroPython*), etc. Sua última versão estável no momento é a 1.9.4²⁵.

¹⁹ <https://www.python.org>

²⁰ [https://en.wikipedia.org/wiki/C_\(programming_language\)](https://en.wikipedia.org/wiki/C_(programming_language))

²¹ https://en.wikipedia.org/wiki/Read-eval-print_loop

²² <https://www.espressif.com/en/products/hardware/esp8266ex/overview>

²³ <https://www.espressif.com/en/products/hardware/esp32/overview>

²⁴ <http://microbit.org>

²⁵ <https://github.com/micropython/micropython>

Na realização deste trabalho foi utilizado microcontrolador LoPy/LoPy4 com ESP32, suporte nativo a MicroPython e comunicação sem fio LoRa. No Capítulo 4 será apresentado o microcontrolador e seus sensores.

4 Tecnologias Utilizadas

Conforme representado na Figura 11, uma típica rede LoRaWAN é composta por sensores que transmitem seus dados via LoRa para um *gateway*. O *gateway* encaminha o dado através de *Wi-Fi*, *Ethernet*, 4G etc, para um servidor de Internet das Coisas na nuvem (*IoT Cloud* em inglês), até chegar à aplicação. A seguir serão apresentados os *hardwares* e *softwares* utilizados neste trabalho. Maiores detalhes sobre a arquitetura e como eles interagem entre si encontram-se disponíveis no Capítulo 6.

4.1 LoPy4

O LoPy4 é uma placa de desenvolvimento, com *MicroPython* nativo e quatro opções de conexão sem fio: LoRa, Sigfox, *Wi-Fi* e *Bluetooth*, produzido pela empresa Pycom. A Figura 13 mostra uma imagem do LoPy4 destacando suas principais características.

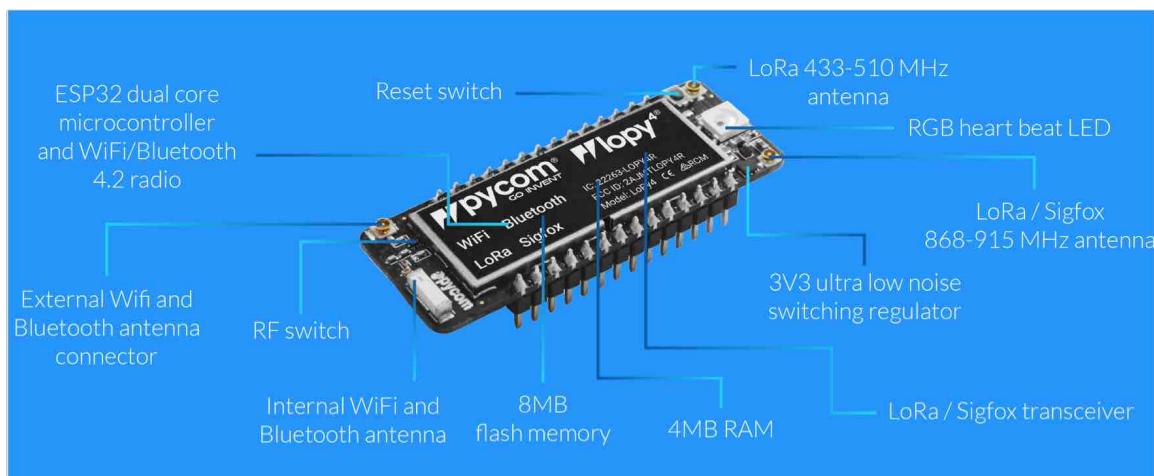


Figura 13 – LoPy4 da Pycom

Fonte: [Pycom\(59\)](#)

Além de ser pequeno, medindo 55 mm x 20 mm, o LoPy4 tem baixo consumo de energia⁽⁵⁹⁾. Possui um *chipset* ESP32 da *Espressif* (60), dois processadores: um processador de rede responsável pelo conectividade do *Wi-Fi* e pilha IPv6, e um processador principal completamente livre para rodar a aplicação programada nele. Ainda possui um coprocessador *Ultra Low Power* (ULP) para monitorar as entradas de *General Purpose Input/Output*¹ (GPIO), os canais *Analog to Digital Converter*² (ADC) e os periféricos internos enquanto o LoPy4 se encontra no modo *deep sleep*.

¹ https://en.wikipedia.org/wiki/General-purpose_input/output

² https://en.wikipedia.org/wiki/Analog-to-digital_converter

Quando o LoPy4 se encontra no modo *deep sleep*, as CPUs, a maior parte da RAM e todos os periféricos digitais são desligados. As únicas partes do *chip* que ainda podem ser ligadas são: controlador *Real-Time Counter* (RTC), periféricos RTC (como por exemplo o coprocessador ULP) e memórias RTC (lentas e rápidas)(61).

É capaz de utilizar LoRa nas bandas: 868 MHz (Europa), 915 MHz (Américas, Austrália e Nova Zelândia), 433 MHz(Europa) e 470-510 MHz (China). Possui 4 MB de *Random Access Memory* com capacidade *multi-threading*, *hardware* de aceleração de ponto flutuante e memória interna *flash* de 8 MB.

Para interfacear com o LoPy4, estão disponíveis duas *Universal Asynchronous Receiver-Transmitter* (UART), um *Serial Peripheral Interface* (SPI), dois *Inter-Integrated Circuit* (I^2C), um *Inter-IC Sound* (I^2S) e oito ADCs de 12 bits. Como segurança tem implementado: *Secure Hash Algorithms* (SHA), *MD5 Algorithm* (MD5), *Data Encryption Standard* (DES) e *Advanced Encryption Standard* (AES).

4.2 Placa de Expansão

A placa de expansão da Pycom, representada na Figura 14, possui 65 mm de largura e 50 mm de comprimento e serve para facilitar o interfaceamento com o LoPy4 (dentre outros microcontroladores da Pycom: WiPy, FiPy e SiPy). Não é necessária uma placa de expansão para programar e utilizar o LoPy4, mas sua utilização torna facilita bastante o processo.

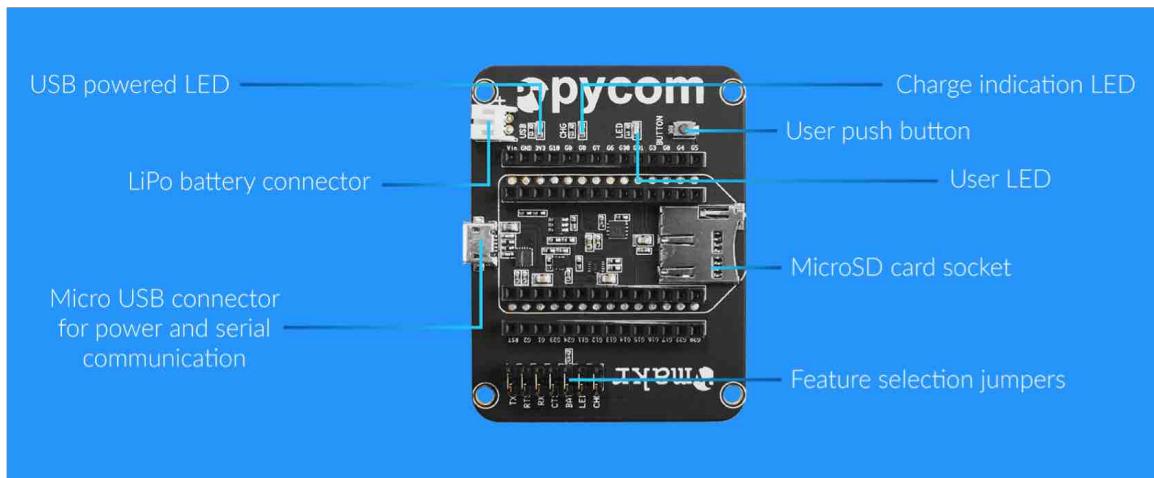


Figura 14 – Placa de Expansão da Pycom

Fonte: [Pycom\(59\)](#)

Possui uma entrada micro *Universal Serial Bus* (USB) e um conector para bateria de polímero de lítio (LiPo), ambas podendo energizar o LoPy4. Devido à presença de

um conversor USB para Serial FT234XD³, a entrada micro USB pode ser utilizada para transferir o programa para a memória interna do LoPy4. Possui também um BQ24040⁴, que se trata de um carregador para bateria LiPo com início de carregamento automático, com possibilidade de escolha entre as correntes de 100 mAh e 450 mAh para carregar a bateria conectada na placa de expansão.

O conector da bateria é do estilo *Japan Solderless Terminal* (JST) PH 2,0 mm 2P, evitando que a bateria se solte com facilidade e, consequentemente, desenergizando o LoPy4. Como mais uma forma para diminuir os riscos de danificar o equipamento, a placa de expansão conta com um *Auto Switching Power MUX* TPS2115A⁵, que protege o circuito caso a bateria seja conectada com a polaridade invertida.

Possui 7 *jumpers* para desabilitar ou habilitar recursos da placa de expansão, além de barra de pinos fêmea para facilitar a conexão de sensores, ou qualquer outro dispositivo externo, ao LoPy4. Por último, possui uma entrada para cartão micro SD de até 32 GB, podendo assim salvar e ler arquivos no cartão micro SD.

4.3 Sensores

A EstAcqua possui sensores de iluminância, pressão atmosférica, umidade, temperatura externa e subaquática. Para isso são utilizados três sensores diferentes:

- **MAX44009:** um sensor de iluminância
- **BME280:** um circuito integrado com sensores de temperatura, umidade e pressão atmosférica
- **DS18B20:** sensor de temperatura com encapsulamento para mensurar temperatura subaquática

A seguir serão detalhadas as especificações desses sensores.

4.3.1 MAX44009

O sensor de luz ambiente MAX44009 possui o menor consumo de energia dentre os sensores de luz disponíveis no mercado⁽⁶²⁾, com menos de 1 μ A de corrente de operação, e possui uma larga faixa de 22 bits de 0,045 lux a 188.000 lux. Demonstrado na Figura 15, possui as dimensões 2 mm x 2 mm x 0,6 mm, com tensão de operação entre 1,7 V e 3,6 V, além de uma corrente típica de operação de 0,65 μ A.

³ http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT234XD.pdf

⁴ <http://www.ti.com/product/BQ24040>

⁵ <http://www.ti.com/lit/ds/symlink/tps2115a.pdf>

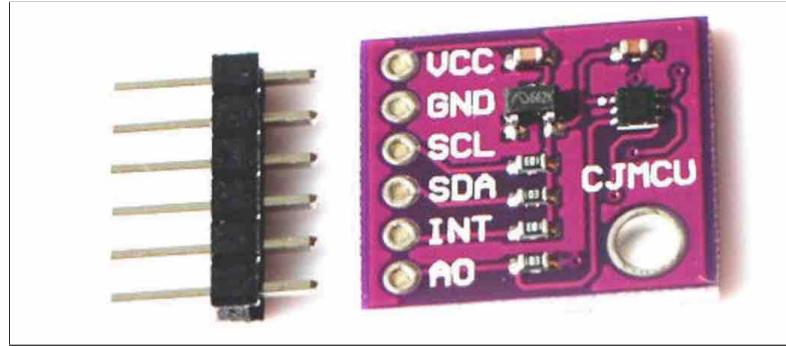


Figura 15 – Sensor de luz ambiente MAX44009, produzido pela *MAXIM Integrated*

Fonte: [eBay\(63\)](#)

O MAX44009 é um Circuito Integrado (CI) com dois fotodiodos e ADC integrado, utilizando interface I^2C para comunicação. Um diferencial desse CI é a existência de dois fotodiodos para fazer a conversão da luz incidente em bits. Um filtro óptico impede que a luz ultravioleta e o infravermelho cheguem no fotodiodo. O segundo fotodiodo, sensível principalmente à luz infravermelha, é então usado para coincidir com a resposta de luz fluorescente e incandescente. Os sinais, após passarem pelo ADC, passam por um processamento de sinal digital e sua resposta é projetada para corresponder à resposta espectral do olho humano. A Figura 16 mostra o diagrama de blocos do MAX44009, onde é possível ver os dois fotodiodos, os ADC e a central de processamento de sinais digitais.

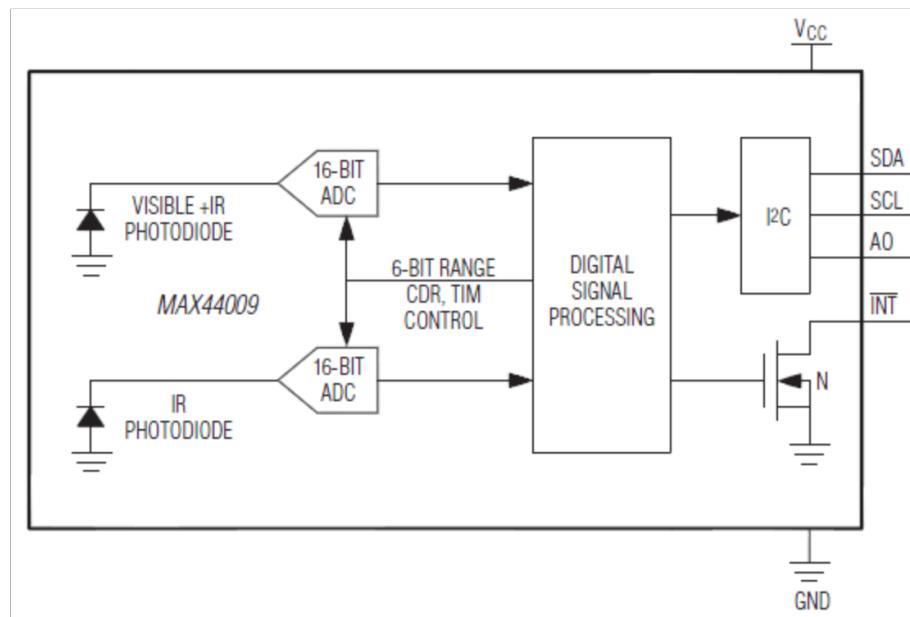


Figura 16 – Diagrama de blocos do MAX44009

Fonte: [MAXIM Integrated\(62\)](#)

4.3.2 BME280

O BME280, mostrado na Figura 17, é um sensor ambiental integrado desenvolvido especificamente para aplicações móveis onde o tamanho e o baixo consumo de energia são as principais restrições de projeto. Combina alta linearidade individual com sensores de alta precisão para pressão, umidade e temperatura, e mede apenas 2,5 mm x 2,5 mm x 0,93 mm.

O BME280 foi projetado para baixo consumo de corrente (3,6 μ A a 1 Hz). O sensor de umidade apresenta um tempo de resposta extremamente rápido. Já o sensor de pressão é um sensor de pressão barométrica absoluta com precisão e resolução excepcionalmente altas e ruído muito baixo. Adicionalmente, o sensor de temperatura integrado foi otimizado para um ruído muito baixo e alta resolução. Apesar de ser usado principalmente para compensação de temperatura dos sensores de pressão e umidade, também pode ser usado para estimar a temperatura ambiente(64).

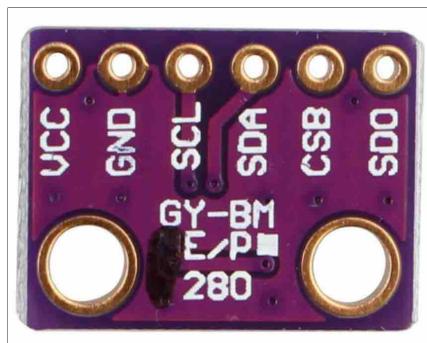


Figura 17 – Sensor de umidade, pressão atmosférica e temperatura BME280, produzido pela Bosch

Fonte: [DX\(65\)](#)

Pode ser alimentado com tensão entre 1,7 V e 3,6 V e está disponível com as interfaces de comunicação I^2C e SPI. O sensor pode ser operado em três modos de energia: modo de suspensão, modo normal e modo forçado. No modo normal, o sensor alterna automaticamente entre uma medição e retorna ao modo de suspensão (onde não faz medição). No modo forçado, o sensor realiza uma única medição sob solicitação e retorna ao modo de suspensão posteriormente. A Figura 18 mostra o diagrama de blocos do BME280, onde é possível ver os sensores de pressão, umidade e temperatura. Os sensores de pressão e temperatura possuem uma resolução de no máximo 20 bits, enquanto que o sensor de umidade possui uma resolução fixa de 16 bits. As leituras dos sensores são enviadas para um ADC e posteriormente para uma unidade lógica que trata os dados e armazena-os nos registradores para que possam ser acessados através da interface de comunicação.

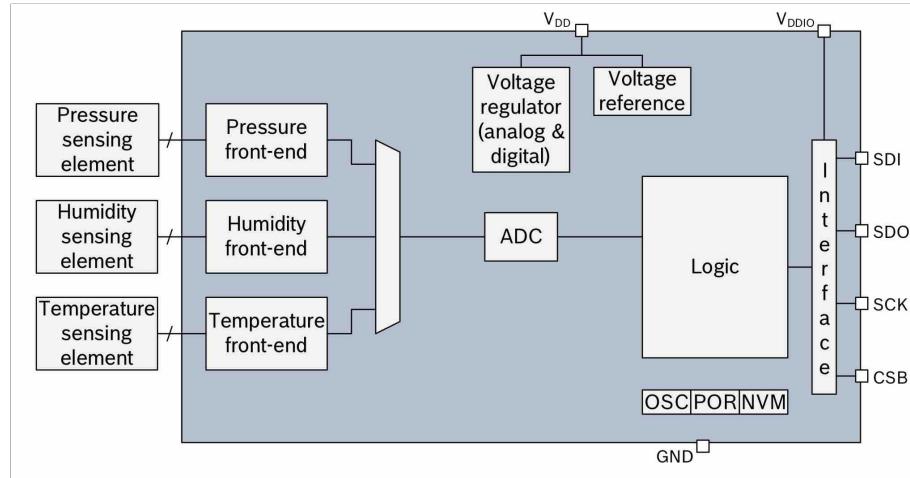


Figura 18 – Diagrama de blocos do BME280

Fonte: [Bosch\(64\)](#)

4.3.3 DS18B20

O termômetro digital DS18B20, mostrado na Figura 19, faz medições de temperatura em graus Celsius, podendo escolher resolução de 9 bits a 12 bits. Possui uma memória não volátil onde o programador pode definir temperatura mínima e máxima para que um alarme seja disparado caso essas temperaturas sejam atingidas. Diferente dos outros sensores listados acima, o DS18B20 se comunica através de um barramento chamado *1-Wire*⁶ (OW), que requer apenas um fio (além dos fios de alimentação VCC e GND) para comunicação com um microprocessador central. Alternativamente, o DS18B20 pode ser configurado para operar no modo "parasita", onde a energia para funcionamento é derivada do fio de comunicação, eliminando a necessidade de uma fonte de alimentação externa.

Figura 19 – Sensor de temperatura DS18B20 com encapsulamento para medições subaquáticas de temperatura, produzido pela *MAXIM Integrated*Fonte: [Amazon\(66\)](#)

⁶ 1-Wire é uma marca registrada da *Maxim Integrated*

Cada DS18B20 possui um código serial exclusivo de 64 bits, que permite que múltiplos DS18B20 funcionem no mesmo barramento OW. Além de poder utilizar vários DS18B20 no mesmo barramento, devido ao código serial único de cada DS18B20 é possível identificar cada um deles no sistema, sabendo sua localização e podendo aplicar fórmulas corretivas ou de compensação para cada DS18B20 individualmente. A Figura 20 mostra o diagrama de blocos do DS18B20, onde é possível ver o circuito para operação em modo "parasita", a *Read Only Memory* (ROM) de 64 bits que contém o serial do DS18B20, o sensor de temperatura e os alarmes para temperatura alta ou baixa.

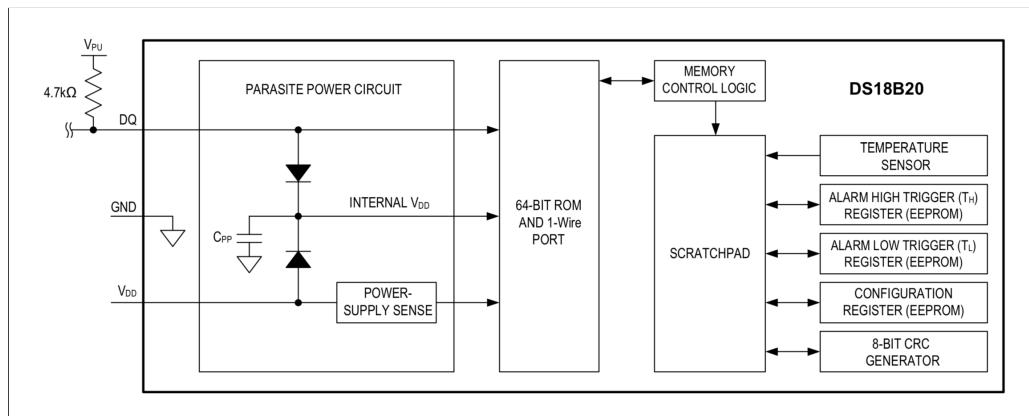


Figura 20 – Diagrama de blocos do DS18B20

Fonte: [MAXIM Integrated\(67\)](#)

Projetado para funcionar com tensão de entrada de entre 3,0 V e 5,5 V, com corrente típica de operação de 1 mA, e 750 nA no modo *stand-by*(67), consegue medir temperaturas entre -55 °C e +125 °C. A Tabela 6 mostra o tempo necessário para que o DS18B20 faça a conversão da temperatura dependendo da resolução usada. Como pode ser observado, mesmo na maior resolução, o tempo de conversão é menor que 1 s. Caso a aplicação não requeira uma grande precisão, pode-se usar a resolução de 9 bits e o tempo de conversão máximo necessário passa a ser menor que 100 ms. Com 9 bits tem-se o menor tempo de conversão porém com resolução de 0,5 °C, já com 12 bits tem-se uma resolução de 0,0625 °C, porém com o maior tempo.

Tabela 6 – Tempo de conversão por resolução do DS18B20

Resolução	Tempo (ms)
9-bits	93,75
10-bits	187,5
11-bits	375
12-bits	750

Outra característica marcante do DS18B20 é a sua acurácia. Para uma longa faixa de temperatura, dos -10 °C até +85 °C, possui uma acurácia de $\pm 0,5$ °C. A Figura 21

mostra a curva de erro típica do DS18B20. Podemos observar que na faixa entre $+20\text{ }^{\circ}\text{C}$ e $+30\text{ }^{\circ}\text{C}$, o erro médio, P , é menor que $0,2\text{ }^{\circ}\text{C}$. Ainda são mostradas as faixas de $+3\sigma$ e -3σ , onde σ é o erro padrão. Na estatística, quando o resultado de uma série de medições é indicado como $(P \pm 3\sigma)$ podemos interpretar da seguinte maneira: o valor mais provável da medida é P , ou, a probabilidade de que o valor real esteja no intervalo $(P - 3S, P + 3S)$ é de 99,7%. Ou seja, é bem provável que a temperatura medida tenha um erro que $0,2\text{ }^{\circ}\text{C}$ e, no pior caso, tem-se 99,7% de certeza que o erro não é maior que $0,4\text{ }^{\circ}\text{C}$.

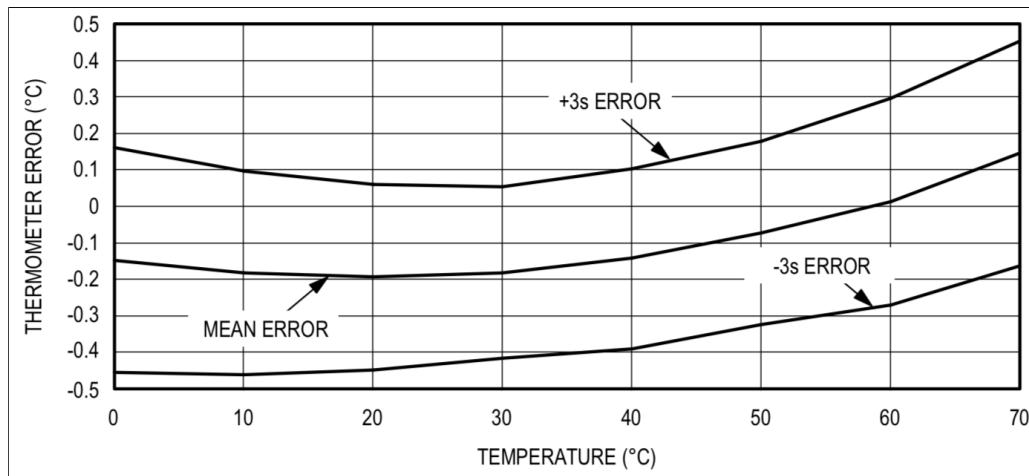


Figura 21 – Curva de erro típica do DS18B20

Fonte: [MAXIM Integrated\(67\)](#)

4.4 The Things Network

A *The Things Network* (TTN) é o servidor *IoT* na nuvem utilizado nesse projeto. É um dos servidores gratuitos para LoRaWAN mais utilizados, com mais de 44 mil desenvolvedores, mais de 4 mil *gateways* de usuários conectados à rede ao redor do mundo e mais de 26 mil aplicações em funcionamento. A Figura 22 mostra a distribuição dos *gateways* ao redor do mundo, em que se percebe que mais da metade dos *gateways* se encontram no oeste europeu.

A TTN comercializa nós e *gateways* LoRa e provê treinamento individual e coletivo para empresas e desenvolvedores que querem entrar no mundo LoRa. Possui uma comunidade bem ativa nos fóruns, sempre colaborando e ajudando a resolver problemas, além de prover diversos meios de integrar a TTN com a aplicação que se deseja usar. Já possui integração nativa com diversas aplicações como: *Cayenne* (utilizado nesse trabalho), *Hypertext Transfer Protocol* (HTTP), permitindo ao usuário realizar *uplink* para um *gateway* e receber *downlink* por HTTP, *OpenSensors*⁷ e *EVRYTHNG*⁸. Caso queira criar

⁷ www.opensensors.io

⁸ www.evrythng.com



Figura 22 – *Gateways LoRaWAN conectados a The Things Network*

Fonte: [The Things Network\(68\)](#)

sua própria aplicação, disponibiliza *Application Programming Interface* (API) para uso com *Message Queuing Telemetry Transport* (MQTT) e diversos *Software Developer Kits* (SDK) para uso com *Python*, *Java*⁹, *Node.Js*¹⁰, *NODE-RED*¹¹ e *Go*¹².

A *The Things Network* é engajada em testar os limites da tecnologia LoRa e encoraja seus usuários a testarem também. Em agosto de 2017, colocaram um pequeno dispositivo LoRa, conectado à TTN, em um balão atmosférico. O balão demorou 3 h para chegar a altitude de 38,8 km. Quando chegou nessa altitude, o balão sobrevoava a cidade de Osterwald (oeste da Alemanha, próximo à divisa da Alemanha com a Holanda). Nesse momento, o nó que estava acoplado ao balão transmitiu um sinal que foi recebido por 148 diferentes *gateways* conectados à TTN. O *gateway* mais longe se encontrava na Breslávia (cidade da Polônia), a uma distância de 702,00 km utilizando apenas 25 mW na transmissão⁽⁶⁹⁾! A Figura 23 mostra o trajeto e a distância percorrida.

Em julho de 2018, outro usuário da TTN testou a transmissão com o nó em terra (ao invés de um balão atmosférico). O sinal transmitido pelo seu nó, situado na cidade de Lowestoft (cidade no leste da Inglaterra), foi recebido por um *gateway* localizado na cidade Holandesa de Utrecht. Ou seja, o sinal viajou por 235 km, atravessando o Mar do Norte, até ser recebido. A Figura 24 mostra o trajeto percorrido pelo sinal.

Também em julho de 2018, a TTN anunciou uma parceria com a *Packetworx*¹³, uma empresa líder em IoT situada nas Filipinas, para que sejam implantados 2.500 *gateways*

⁹ <https://www.oracle.com/java/>

¹⁰ <https://nodejs.org/>

¹¹ <https://nodered.org/>

¹² <https://golang.org/>

¹³ <https://packetworx.com>

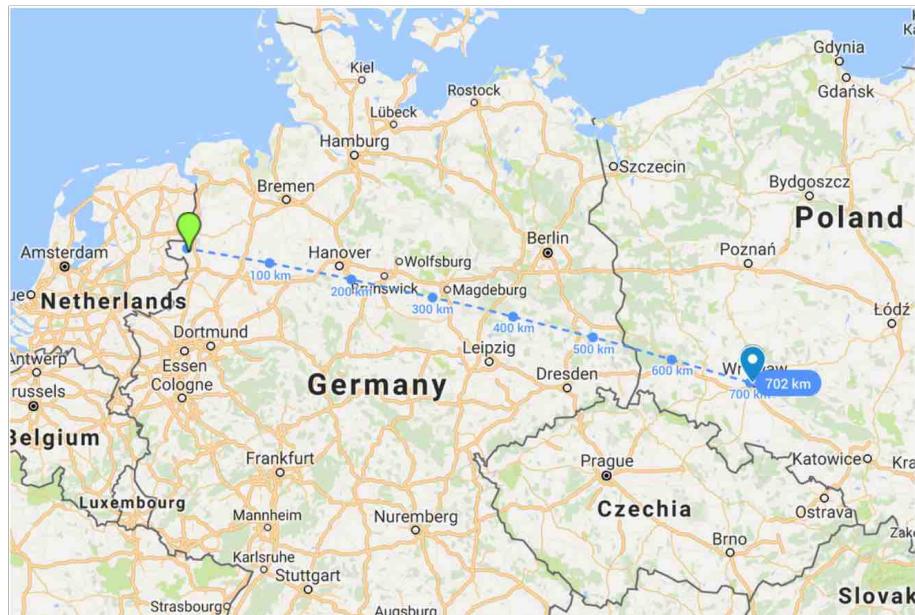


Figura 23 – Mapa com a distância percorrida pelo sinal transmitido até a chegada no *gateway*

Fonte: The Things Network(68)

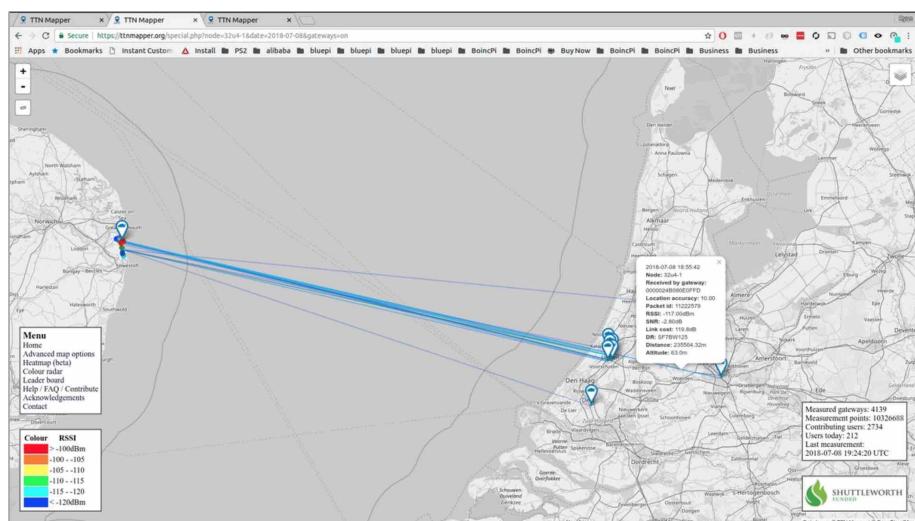


Figura 24 – Sinal transmitido por um nó localizado no leste da Inglaterra é recebido na Holanda

Fonte: The Things Network(68)

conectados a TTN (70). Dessa forma os desenvolvedores das Filipinas e do sudoeste da Ásia poderão lançar suas aplicações na rede da TTN se juntando aos mais de 4 mil *gateways* já existentes.

4.4.1 LoRaWAN Network Stack

Se trata de um servidor de rede LoRaWAN completo com gerenciamento para gateways, aplicativos, dispositivos e usuários, possibilitando a utilização de APIs abertas para integrar a solução implantada na nuvem ou localmente. Possui recursos aprimorados de segurança de ponta a ponta, suporte a balanceamento de carga de acordo com as certificações de segurança ISO.

Se trata de uma pilha de servidores LoRaWAN completa e de código aberto, em sua terceira versão, projetada para vários cenários de implantação, suportando todas as versões existentes do LoRaWAN (incluindo sua última versão, 1.1), modos de operação A, B e C e todos os parâmetros regionais. A Figura 25 detalha os componentes da pilha.

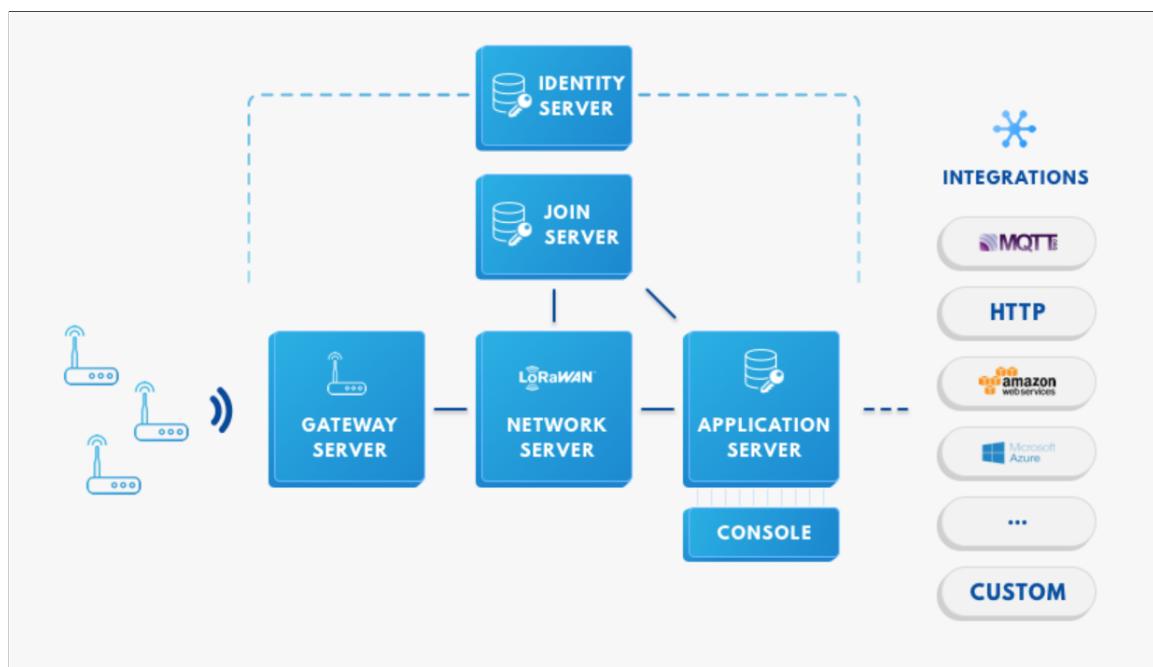


Figura 25 – Visão geral da *Things Network Stack* V3

Fonte: TTN(71)

Utiliza criptografia, integridade e autenticação no nível da rede e do aplicativo. Em sua terceira versão, utiliza *Join Servers* que armazenam com segurança as chaves LoRaWAN e emite as chaves de sessão para o *Network Server* e *Application Server*. Isso separa o armazenamento seguro do roteamento de pacotes, permitindo que os usuários hospedem servidores de associação no local e usando módulos seguros de *hardware* para manter controle total sobre chaves de segurança.

Foi projetado para ser implantado como uma rede privada com gateway único, com escalabilidade para ser utilizado comercialmente e globalmente distribuída. O *Things Network Stack* V3 não é apenas sobre serviços em nuvem: os *gateways* são fundamentais em cada rede LoRaWAN, portanto, além do suporte a qualquer gateway com encaminhador

de pacote padrão, permite configuração remota, conectividade segura e um modo de baixa largura de banda para reduzir custos.

4.4.2 Alternativas

O LoRIOT (72) é uma nuvem *IoT* paga, completa (com servidor de rede e aplicação) que opera no Brasil. Apesar de ser voltada para empresas, é possível adquirir planos para todo tipo de demanda. O valor a ser pago depende da quantidade de nós, gateways e mensagens transmitidas por dia.

Também é possível instalar o próprio servidor de rede LoRa e aplicação. Existem projetos de software aberto como por exemplo o LoRaServer (73). Se trata de um servidor de rede LoRaWAN, com suporte a dispositivos Classe A, B e C, suporte a autenticação via OTAA e ABP, suporte a todas regiões. A instalação requer um servidor com o sistema operacional Debian ou Ubuntu, banco de dados PostgreSQL e um servidor MQTT.

Outra opção é o LoRaWAN-server (74), com suporte a diversos *gateways*, com verificação de criptografia e integridade dos dados, suporte a autenticação via OTAA e ABP, suporte a *Cayenne LPP*, suporte aos protocolos HTTP 1.1 e HTTP 2.0 (REST API), suporte a todos parâmetros regionais LoRa. Se diferencia do LoRaServer por ser programado usando Erlang^{14,15} e por funcionar em todos os principais sistemas operacionais, incluindo Windows, Linux, OS X e Solaris e até mesmo em sistemas embarcados, como por exemplo o Raspbian.

4.5 Cayenne

O Cayenne é o primeiro construtor de projetos de IoT de arrastar e soltar do mundo (75), permitindo que desenvolvedores, *designers* e engenheiros criem rapidamente protótipos e compartilhem seus projetos de dispositivos conectados. O Cayenne foi projetado para ajudar os usuários a criarem protótipos da Internet das Coisas e depois trazê-los para a produção.

É uma aplicação que permite adicionar atuadores, extensões e sensores com facilidade. Possibilita a criação de eventos como alertas de limites e atividades dos sensores, aplicando um mecanismo de regras e acionando quando ocorrem ações ou situações específicas.

Duas principais características do Cayenne são a existência um aplicativo para celulares, possibilitando que o usuário monitore e controle remotamente seus projetos de IoT a partir do Android ou iOS, e um *dashboard online* em que é possível utilizar *widgets*

¹⁴ <https://www.erlang.org/>

¹⁵ Desenvolvida pela Ericsson para suportar aplicações distribuídas e tolerantes a falhas a serem executadas em um ambiente de tempo real e ininterrupto

personalizáveis para visualizar dados, configurar regras e agendar eventos, tornando mais fácil a visualização dos dados.

Se trata de um serviço gratuito, não exclusivo para LoRa, podendo adicionar uma gama de sensores e microcontroladores.

4.6 Baterias

A seguir serão apresentadas diferentes tipos de tecnologia de bateria, sendo elas: *Li-Po*, *Li-Ion*, *LiFePO₄*, *LiSOCl₂*.

- *Li-Po*

Uma bateria de polímero de lítio ou, mais corretamente, uma bateria de polímero de íon de lítio, é uma bateria recarregável de íon de lítio usando um eletrólito polimérico em vez de um eletrólito aquoso. Essas baterias fornecem energia específica mais alta do que outros tipos de baterias de lítio e são usadas em aplicações em que o peso é uma característica essencial, como dispositivos móveis e aeronaves controladas por rádio.

A voltagem de uma célula *Li-Po* depende da sua química e varia de cerca de 2,7-3,0 V (descarregada) a cerca de 4,2 V (totalmente carregada). As baterias *Li-Po* oferecem aos fabricantes vantagens competitivas. Eles podem facilmente produzir baterias de praticamente qualquer formato desejado. Por exemplo, os requisitos de espaço e peso de dispositivos móveis e notebooks podem ser completamente satisfeitos. Além disso, eles têm taxa de baixa auto-descarga, que é de cerca de 5% ao mês (76).

- *Li-Ion*

O íon de lítio é uma bateria de baixa manutenção, uma vantagem que a maioria das outras químicas não pode reivindicar. Não há memória e não é necessário efetuar um ciclo programado para prolongar a vida da bateria. Além disso, a auto-descarga é inferior a metade em comparação com o NiCd, tornando o íon de lítio adequado para aplicações modernas. As células de íons de lítio causam pouco dano quando descartadas.

Entretanto, requer circuito de proteção para manter a tensão e a corrente dentro dos limites de segurança. Ainda está sujeita ao envelhecimento, mesmo que não esteja em uso. Existem restrições de transporte, onde o envio de quantidades maiores pode estar sujeito a controle regulatório. Esta restrição não se aplica a baterias pessoais de mão. O custo de fabricação chega a ser até 40% maior que o custo de uma bateria de NiCd.

- *LiFePO₄*

As baterias *LiFePO₄* são mais conhecidas por seu forte perfil de segurança, resultado de uma química extremamente estável. Quando sujeitas a eventos perigosos, como colisão ou curto-círcuito, eles não explodem ou pegam fogo, reduzindo significativamente qualquer chance de dano.

A vida útil geralmente é de cinco a dez anos ou mais. O tempo de carregamento da bateria também é consideravelmente reduzido, outro benefício de desempenho conveniente. Também vale a pena mencionar as características de espaço eficiente do *LiFePO₄*. Com um terço do peso da maioria das baterias de chumbo-ácido e quase a metade do peso do popular óxido de manganês, a *LiFePO₄* oferece uma maneira eficaz de usar espaço e peso.

- *LiSOCl₂*

Possui uma tensão de circuito aberto de 3,67 V e uma tensão de operação de 3,60 V, que são consideravelmente maiores do que em qualquer outra bateria primária comercialmente disponível (77). A descarga automática dessa bateria é extremamente baixa, menos de 1%, o que pode suportar longos períodos de armazenamento e atingir uma vida útil de 10 a 20 anos.

É indicada para uma gama de aplicações: medição de energia elétrica (medidor de eletricidade, medidor de gás, medidor de água e medidor de calor), alarmes e dispositivos sem fio de segurança, GPS, eletrônica profissional, exploração de óleo e até mesmo telemetria automotiva.

5 Análises LoRa e Colisão de Dados

Nesse capítulo serão apresentadas os testes de transmissão efetuados para determinação dos melhores parâmetros LoRa para utilização na EstAcqua e a simulação de uma rede LoRaWAN com centenas de nós para estimar a taxa de colisão de dados.

5.1 LoRa

Para prover uma comunicação eficiente e confiável, existe uma larga variedade de parâmetros. O conjunto de parâmetros, S , para cada dispositivo LoRa é formado conforme demonstrado em 5.1:

$$S = \{SF, BW, CR, TP, F\} \quad (5.1)$$

Considerando os parâmetros válidos para a banda US915, temos:

- ***SF***: o fator de espalhamento, sendo possíveis os valores 7, 8, 9 ,10, 11 e 12.
- ***BW***: a largura de banda, sendo possível 125 kHz ou 500 kHz
- ***CR***: o *Code Rate*, cujos valores podem ser 4/5, 4/6, 4/7 ou 4/8.
- ***TP***: *Transmission Power*, a potência de transmissão, podendo ser entre 0 e 20 dBm.
- ***F***: A frequência utilizada, lembrando que existem 80 canais definidos na banda US915.

A frequência escolhida não possui um impacto direto na qualidade da transmissão, sendo mais relevante em locais com diversos dispositivos LoRa operando, onde a escolha de uma frequência menos congestionada torna-se importante. Nem no local de testes e nem onde protótipo foi instalado haviam dispositivos LoRa operando, fato que facilitou a escolha da frequência. A TTN possui uma lista de definições de planos de frequência para os dispositivos que operam em sua rede (78). Esses planos de frequência são baseados no que é especificado no documento de parâmetros regionais LoRaWAN (34). Como o que muda de plano para plano é a frequência de transmissão e no local de instalação da EstAcqua não existe outro dispositivo transmitindo com frequência sub-GHz, foi escolhido o primeiro plano, com frequência de 903,9 MHz podendo utilizar DR0, DR1, DR2 ou DR3 (Tabela 2).

Uma vez definida a frequência, ainda falta decidir qual SF, BW, CR e TP serão utilizados. São centenas de combinações possíveis, tornando impraticável testar cada uma. É preciso determinar a configuração que minimiza o custo de energia de transmissão e atenda ao desempenho e especificações de comunicação necessários. Para isso é necessário entender o impacto da seleção de parâmetros da transmissão.

A potência de transmissão (TP) tem um impacto direto no consumo de energia, quanto maior a potência maior será o consumo de energia. Outro fator que determina o consumo de energia é o tempo necessário para transmitir (*Air Time*) um pacote, que depende da combinação de CR, BW e SF selecionados. A influência desses parâmetros no tempo de ar do pacote acontece da seguinte forma (79):

- **SF:** a diminuição do SF resulta em um tempo de ar menor, enquanto o aumento do SF resulta em um tempo de ar maior.
- **BW:** a diminuição da largura de banda resulta em um tempo de ar maior, enquanto que o aumento da largura de banda resulta em um tempo de ar menor.
- **CR:** um CR igual a 4/5 possui um menor tempo de ar, já um CR de 4/8 possui um tempo de ar maior.

Como o local de instalação do protótipo é no Terra Alta, conforme mostrado na Figura 1, foram realizados testes de transmissão, variando esses parâmetros, em um ambiente mais similar ao local real de instalação do protótipo. Por isso foi escolhido a praia de Camburi, situado na cidade de Vitória no Espírito Santo. Nesta primeira avaliação, *gateway* ficou posicionado em um píer e foram realizados os testes de transmissão com o nó posicionado em duas distâncias: primeiro a 500 m de distância com o nó na areia da praia, e o outro a 2,23 km de distância com o nó posicionado em outro píer bem próximo do nível do mar. A Figura 26 mostra a localização do *gateway* e as posições do nó.

Para avaliar os efeitos à medida que se aproxima da água do mar, o *gateway* foi posicionado no píer em três alturas diferentes em relação ao nível do mar, sendo:

- **Posição 1:** $(5,5 \pm 0,4)$ m
- **Posição 2:** $(2,9 \pm 0,3)$ m
- **Posição 3:** $(1,1 \pm 0,1)$ m

Para o teste de transmissão foram selecionadas 26 configurações diferentes, variando CR, SF, BW e TP, transmitindo pacotes de 10 *Bytes*. Posteriormente foram comparados os dados recebidos pelo *gateway* com os dados transmitidos para análise de taxa de perdas, relação sinal/ruído, vazão e *air time* para que fosse escolhida uma configuração



Figura 26 – Local da realização do teste e as respectivas distâncias entre os nós e o *gateway*

que atenda aos requisitos da EstAcqua, garantindo uma boa qualidade do sinal, uma vazão compatível com as necessidades da EstAcqua e minimizando o *air time* e energia gasta para transmissão. A Tabela 7 detalha os parâmetros de cada configuração (C), utilizada.

Além das restrições de transmissão estabelecidas pelos órgãos reguladores, ainda existem restrições de transmissão do servidor *IoT* escolhido. No caso da TTN, existe uma política de acesso justa (80) (*Fair Access Policy*), onde cada *gateway* registrado na rede pode enviar 30 s de mensagens *uplink* (do *gatyeway* para a TTN) por dia, e apenas 10 mensagens de *downlink* (da TTN para o *gateway*) por dia. Ou seja, quanto menor for o tempo de ar do pacote, mais mensagens poderão ser enviadas por dia.

Para o cálculo do tempo de ar do pacote, leva-se em conta o SF, BW, CR e o tamanho do pacote. Como o tamanho do pacote utilizado no teste é fixo em todas as configurações, 10 Bytes, apenas as configurações 1-13, 14 e 16 representam configurações distintas dos parâmetros supracitados. Usando o auxílio de uma calculadora de parâmetros LoRa (81), foi calculado o tempo de ar do pacote nessas 15 configurações distintas, conforme mostrado na Figura 27.

Sendo:

- **BW:** 0 para 125 kHz, e 2 para 500 kHz
- **CR:** 1 para 4/5, e 4 para 4/8

Anteriormente foi explicado que configurações diferentes podem resultar em um tempo de símbolo igual. Mas é importante ressaltar que o tempo de símbolo ser igual para duas configurações distintas não implica em um tempo de ar do pacote igual. Por exemplo,

Tabela 7 – Parâmetros selecionados para cada configuração do teste

C	SF	BW	CR	TP
1	7	0	1	20
2	7	2	1	20
3	8	0	1	20
4	8	2	1	20
5	9	0	1	20
6	9	2	1	20
7	10	0	1	20
8	10	2	1	20
9	11	0	1	20
10	11	2	1	20
11	12	0	1	20
12	12	2	1	20
13	7	2	1	5
14	7	2	4	5
15	12	2	1	5
16	12	2	4	5
17	7	2	1	10
18	7	2	4	10
19	12	2	1	10
20	12	2	4	10
21	7	2	1	15
22	7	2	4	15
23	12	2	1	15
24	12	2	4	15
25	7	2	4	20
26	12	2	4	20

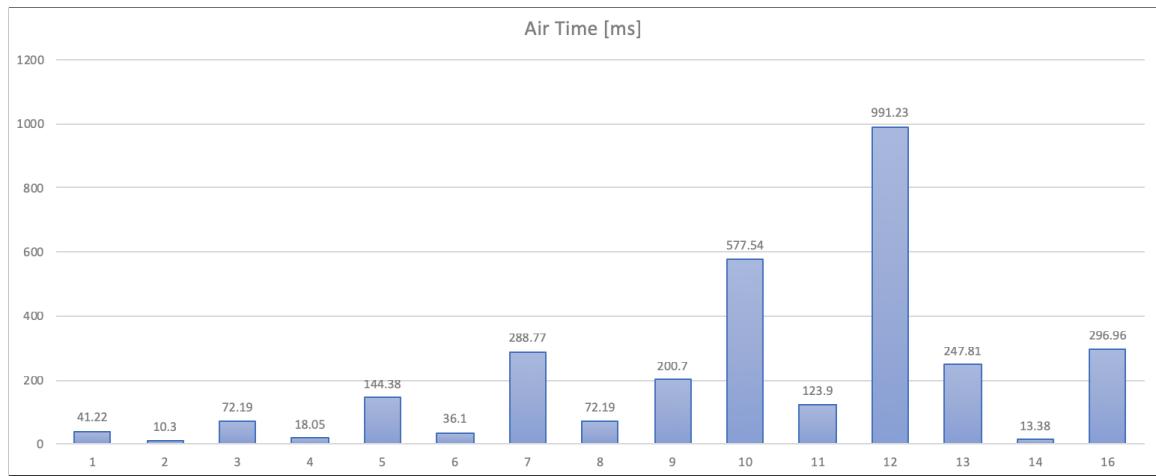


Figura 27 – Tempo de ar do pacote enviado por configuração

pela Fórmula 3.1 temos que o tempo de símbolo para a Configuração 1 ($SF = 7$ e $BW = 125$ kHz) será igual ao da Configuração 6 ($SF = 9$ e $BW = 500$ kHz). Apesar do tempo de símbolo ser igual, pode-se perceber na Figura 27 que o tempo de ar da Configuração 1 é

diferente da Configuração 6. É importante ficar atento a esse detalhe na escolha de quais parâmetros utilizar para evitar que haja um desperdício de energia e aumento do tempo de ar do pacote.

Outro fator importante para a decisão é a taxa de perda dos pacotes transmitidos. Há uma relação de compromisso entre as configurações: maior alcance ou maior resiliência a interferências. A Figura 28 mostra a taxa de perda, em porcentagem, dos pacotes transmitidos com o nó nas distâncias de 500 m e 2,23 km.

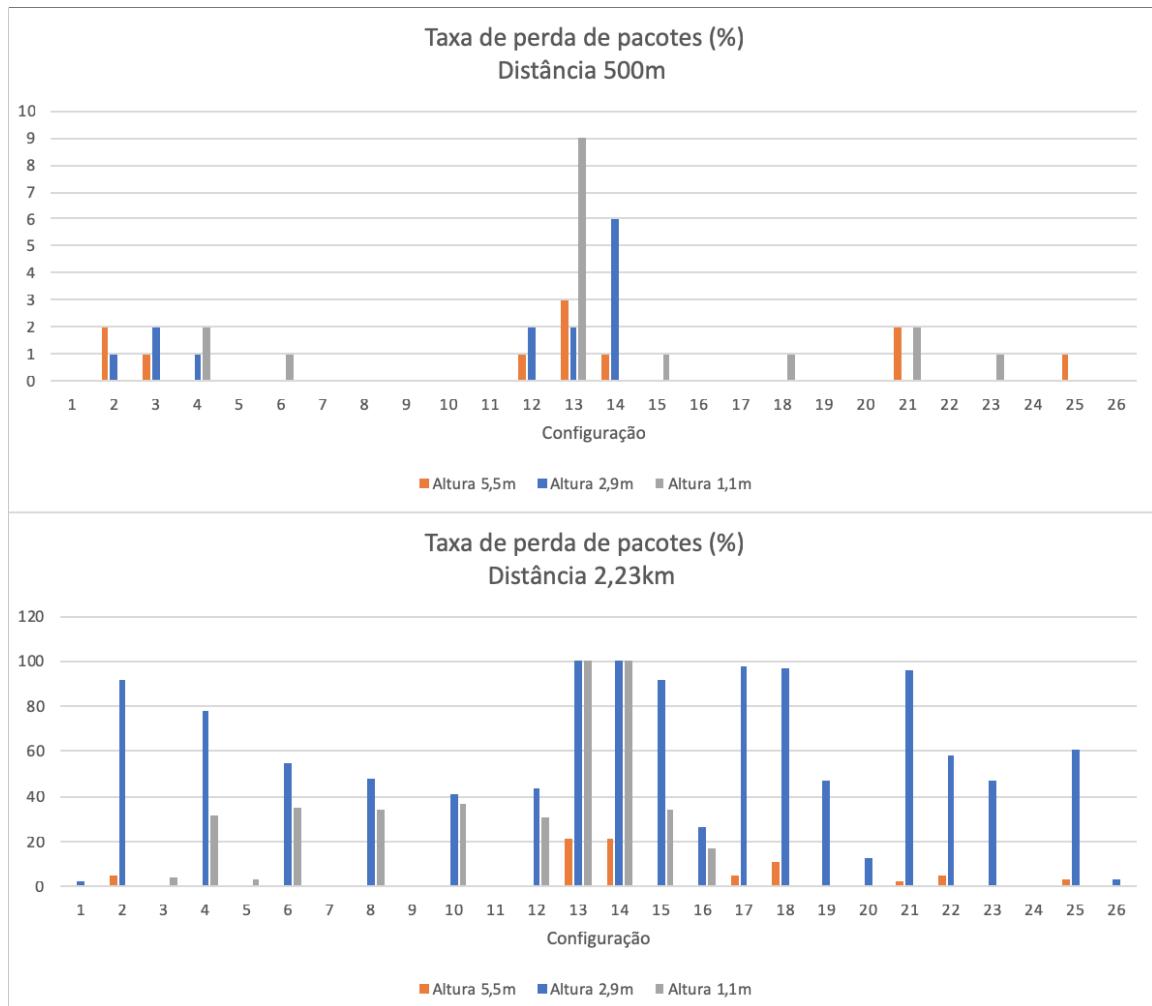


Figura 28 – Perda de pacotes por configuração nas distâncias de 500 m e 2,23 km

Nela podemos perceber o efeito da potência de transmissão. Pela Tabela 7, vemos que as Configurações 13 a 16 possuem TP = 5 dBm, ou seja, a menor potência de transmissão dos testes. Apesar das Configurações 13 e 14 terem apresentado 100% de perda com o nó situado a 2,23 km de distância do *gateway*, aumentando o fator de espalhamento para 12 e o *Code Rate* para 4/8, a taxa de perda de pacotes ficou por volta de 20% - como pode ser visto na Configuração 16, representada no gráfico.

Outra métrica importante na escolha de quais parâmetros utilizar é a vazão. A Figura 29 mostra a vazão, em bps, para cada configuração. Como é de se esperar, com o

aumento da distância, a vazão sofre uma queda devido às perdas de pacotes.

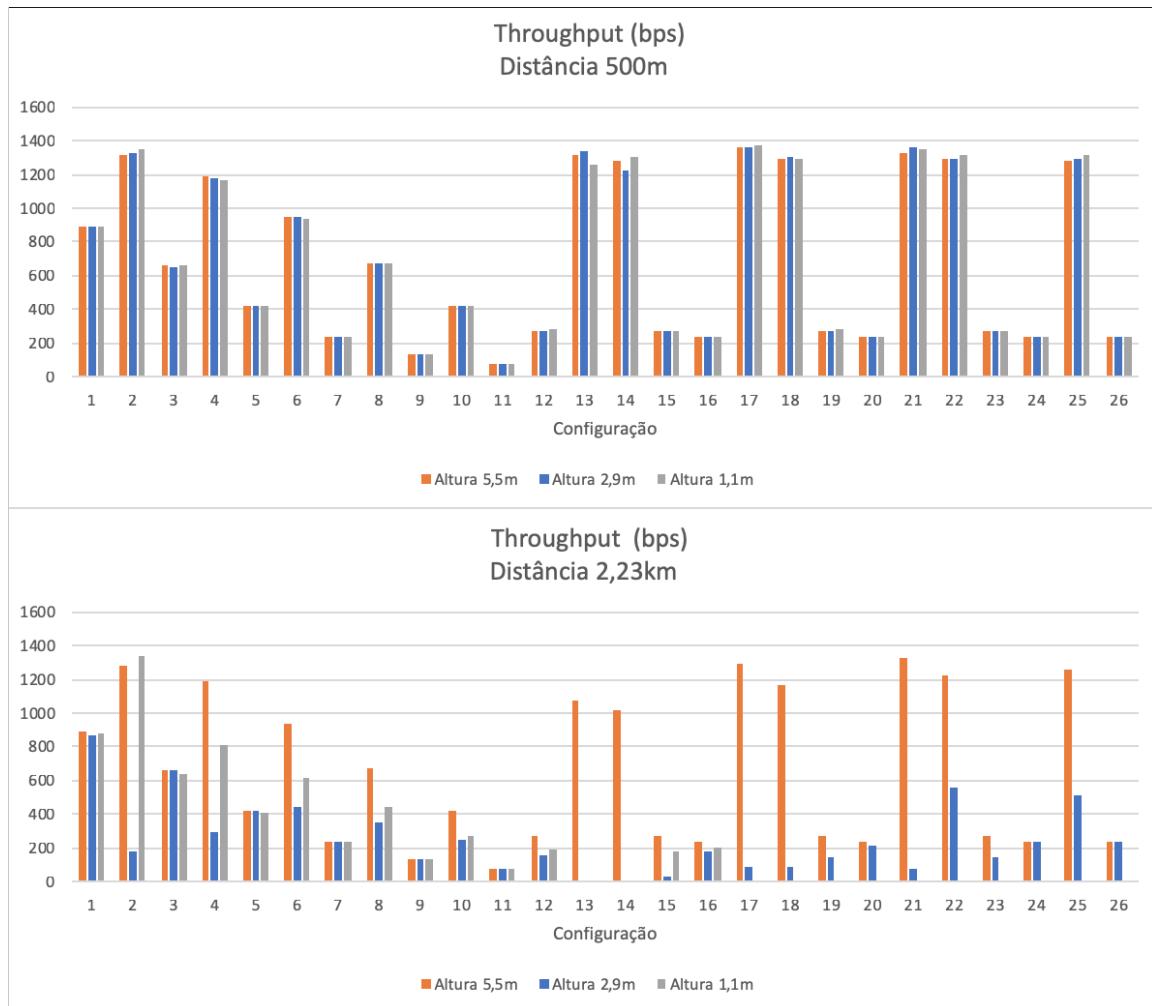


Figura 29 – Vazão, em *bits* por segundo, por configuração nas distâncias de 500 m e 2,23 km

Medir a vazão mostra o quanto diferente um sistema se comporta em relação aos cálculos teóricos. Por exemplo, utilizando a Fórmula 3.3 para calcular a taxa de transferência de bps, com os parâmetros utilizados em cada configuração, encontrariam-se valores bem maiores. Caso um sistema tivesse sido projetado com base nos valores teóricos, ao colocá-lo em funcionamento se teria uma grande surpresa. A Figura 30 mostra a porcentagem que a vazão representa em relação ao cálculo teórico de cada configuração. Como pode ser observado, nenhum chegou a 35% do valor teórico.

O tempo de ar do pacote representa um papel importante nas escolhas dos parâmetros a serem escolhidos. Um tempo de ar do pacote baixo está diretamente relacionado a um menor consumo da bateria. Conforme dito anteriormente, a TTN disponibiliza somente 30 s de mensagens diárias, ou seja quanto menor for o tempo de ar do pacote mais pacotes poderão ser enviados em um dia. Além das limitações da TTN e do consumo de energia, deve-se ater as limitações impostas pela ANATEL(40) que limita o tempo de permanência



Figura 30 – Porcentagem da vazão em relação ao cálculo de *bits* teórico para cada configuração

no canal em 400 ms. Para a escolha de qual configuração utilizar buscamos, idealmente, as seguintes características: menor tempo de ar do pacote, menor taxa de perdas, maior vazão. Analisando separadamente cada um desses aspectos e criando conjuntos que atendem a essas especificações, temos:

- **Tempo de ar (T_a):** {1, 2, 3, 4, 5, 6, 8, 11 e 14}
- **Taxa de perda (T_p):** {1, 3, 5, 7, 9, 11, 20, 24, 26}
- **Vazão (T):** {1, 3}

A interseção desses três conjuntos resultará em um conjunto de configurações que atendem às especificações supracitadas:

$$T_a \cap T_p \cap T = \{1, 3\} \quad (5.2)$$

Consultando a Tabela 7, vemos que as Configurações 1 e 3 representam os parâmetros $\{SF = 7, BW = 125 \text{ kHz}, CR = 4/5, TP = 20\}$ e $\{SF = 8, BW = 125 \text{ kHz}, CR = 4/5, TP = 20\}$, respectivamente.

Como boas práticas para transmissão LoRaWAN, deve-se priorizar a escolha que minimiza o tempo de ar do pacote (82), no caso $SF = 7$ e $BW = 125 \text{ kHz}$, e aumentar o *data rate* caso haja a necessidade de maior alcance; o *payload* máximo das duas configurações são 242 *Bytes* e 129 *Bytes* para SF7BW125 e SF8BW125, ou seja, ambas suportam o *payload* gerado pela EstAcqua (44 *Bytes* - 4 sensores de temperatura submersa, 1 - temperatura externa, 1 - pressão, 1 - umidade e 1 - iluminância). A lagoa onde será instalada o protótipo possui margens situadas a mais de 3 km de distância do *gateway* e, seguindo as normas da ANATEL e FCC para potência de transmissão, optamos em utilizar $SF = 8, BW = 125 \text{ kHz}, CR = 4/5$ e $TP = 20 \text{ dBm}$. Dessa forma, a EstAcqua funcionará seguindo a regulamentação, com um bom alcance e baixa taxa de perda.

5.2 Colisão de Dados

Uma forma utilizada para mitigar a colisão de dados, é através de regulamentações de uso de aplicações com baixo *Duty Cycle* (83). Quando dois pacotes chegam simultaneamente em um *gateway* podem ocorrer três situações: o *gateway* consegue decodificar os dois pacotes, um dos pacotes é descartado ou nenhum pacote é decodificado. Os fatores que definem o que irá ocorrer são o *Spreading Factor*, frequência, potência e tempo de chegada do pacote. Como LoRa é uma forma de modulação de frequência, pode ocorrer um fenômeno chamado Efeito de Captura (84). O Efeito de Captura ocorre quando dois sinais chegam ao *gateway* e o sinal mais fraco é suprimido pelo sinal mais forte.

Em países conectados com diversos *gateways*, como, por exemplo, Bélgica e Holanda (68), o dado transmitido por algum nó LoRa será recebido por vários *gateways*, conforme demonstrado na Figura 11. Haverá colisão de dados, mas eventualmente algum *gateway* receberá o pacote e o transmitirá para a nuvem *IoT*. Um *gateway* certificado LoRaWAN possui suporte a múltiplos canais de *uplink*, como por exemplo o *Cisco Wireless Gateway for LoRaWAN*, que possui 16 canais para *uplink* (85). Dessa forma, além de aumentar a vazão da rede, aumenta a probabilidade do dado transmitido ser recebido em algum *gateway*.

No presente trabalho foram utilizados dois microcontroladores LoPy4, sendo um para o nó e outro para o *gateway*. O microcontrolador LoPy4 está em conformidade com o padrão LoRaWAN desde que seja utilizado como um dispositivo Classe A, B ou C (55). Para o *gateway* é possível utilizar o *Semtech UDP Packet Forwarder*(86), que se trata de um encaminhador de pacotes utilizando o *Semtech UDP Protocol* (87) - primeiro protocolo de *gateway*, desenvolvido e mantido pela Semtech, para LoRaWAN. Dessa forma é possível

registrar um *gateway* na TTN, entretanto o mesmo não fica em conformidade com o padrão LoRaWAN por se tratar de um *gateway single-channel*. É um modo barato de começar a explorar as possibilidades LoRa, mas como apenas conseguem receber dados em uma única frequência e único SF, provê uma baixa cobertura, baixa vazão e não é recomendado que se utilize *gateways* dessa forma em aplicações (88), sejam elas comerciais ou caseiras.

Para as análises de colisão de dados foi utilizado o software LoRaSIM (89), desenvolvido pela Universidade de Lancaster, em um estudo sobre escalabilidade de redes LoRa (90).

5.2.1 LoRaSIM

O LoRaSIM é um simulador de eventos discretos baseado no SimPy¹ para simular colisões em redes LoRa e análise de escalabilidade. Existem quatro módulos disponíveis para a simulação de colisão de dados. A simulação utilizada foi a que retrata a situação de operação da EstAcqua: vários nós transmitindo para um único *gateway*. Todos os nós e o *gateway* são configurados para a mesma frequência, *Spreading Factor*, *Code Rate* e largura de banda, e é definido o intervalo médio de transmissão. Como esse simulador foi programado para ser utilizado nos experimentos descritos no trabalho de Bor et al.(90), foi necessário realizar pequenas modificações no código para que fosse possível fazer a simulação com os nós e *gateway* com os mesmos parâmetros utilizados na EstAcqua: {SF = 8, BW = 125 kHz, CR = 4/5, TP = 14 dBm, F = 903,9 MHz}.

A métrica utilizada para avaliar a colisão de dados é a *Data Extraction Rate* (DER), que representa a proporção de mensagens recebidas em relação às mensagens transmitidas durante um período de tempo. DER é um valor compreendido entre 0 e 1 e, quanto mais próximo o valor estiver de 1, menor a taxa de colisões. Em uma implantação perfeita, seria esperado DER = 1, ou seja, todos os dados transmitidos foram recebidos. Essa métrica não representa o desempenho individual de um nó, mas sim a implantação da rede como um todo.

Para iniciar a simulação deve-se utilizar o seguinte comando:

```
1 $ ./loraDir.py <NODES> <AVGSEND> <EXPERIMENT> <SIMTIME>
```

Onde:

- **NODES:** quantidade de nós que se deseja simular.
- **AVGSEND:** tempo médio entre a transmissão dos dados pelos nós, em ms.

¹ Simulador de eventos discretos para Python (<https://simpy.readthedocs.io>)

- **EXPERIMENT:** experimento que se deseja simular. A alteração realizada no código foi nessa parte para que fosse criado um experimento onde os nós estão configurados iguais ao nó da EstAcqua.
- **SIMTIME:** tempo que se deseja simular as transmissões, em ms.

Foram realizadas 500 simulações onde cada simulação incrementava em 1 unidade o número de nós. Para facilitar o registro das simulações, o código do simulador foi novamente alterado para gravar cada simulação no formato *Comma Separated Value* (CSV). As simulações foram realizadas considerando um tempo médio de transmissão de cada nó de 10 min e o tempo de simulação de 1 dia. O seguinte *bash script* foi desenvolvido para executar as 500 simulações, começando com 1 nó e incrementando em 1 unidade o número de nós até chegar em 500 nós.

```

1 #!/bin/bash
2 for value in {1..500}
3 do
4     echo Iniciando simulacao com $value nos
5     ./loraDir.py $value 600000 9 86400000
6 done
7 echo $value simulacoes foram efetuadas

```

Na Figura 31 o valor de DER após cada simulação, começou com 1 nó e chegando até 500 nós. Como era esperado, à medida que mais nós são adicionados a rede, o valor de DER vai reduzindo. Nela pode-se observar que por volta de 300 nós na rede resulta em $DER = 0,9$, significando que 10% dos pacotes transmitidos sofreram colisão. Caso o *gateway* utilizado estivesse em conformidade com os padrões LoRaWAN, seria possível utilizar diversos canais de *uplink* e seria possível utilizar a ortogonalidade dos *data rates*, aumentando assim a capacidade da rede e reduzindo o número de colisões. Um estudo realizado pela Semtech em parceria com a Comcast² nos Estados Unidos creditou o índice de 95% de sucesso sob condições de tráfego extremamente estressantes ao fato dos *data rates* serem ortogonais (91).

A Figura 32 mostra a quantidade de pacotes transmitidos e a quantidade de pacotes colididos para cada simulação. Com 500 nós tem-se aproximadamente 71000 pacotes transmitidos e 11000 pacotes colididos.

Apesar do foco deste trabalho não ser sobre a escalabilidade da tecnologia LoRa, é importante fazer o teste de colisão e correlacionar com outros estudos realizados. De acordo com o estudo de Bor et al.(90), a escalabilidade em redes com apenas um *gateway* transmitindo pacotes de 20 *Bytes* a cada 16,7 min atingiu uma taxa de colisão de 10% com apenas 64 nós na rede. Ou seja, uma baixa escalabilidade.

² <https://www.comcast.com>

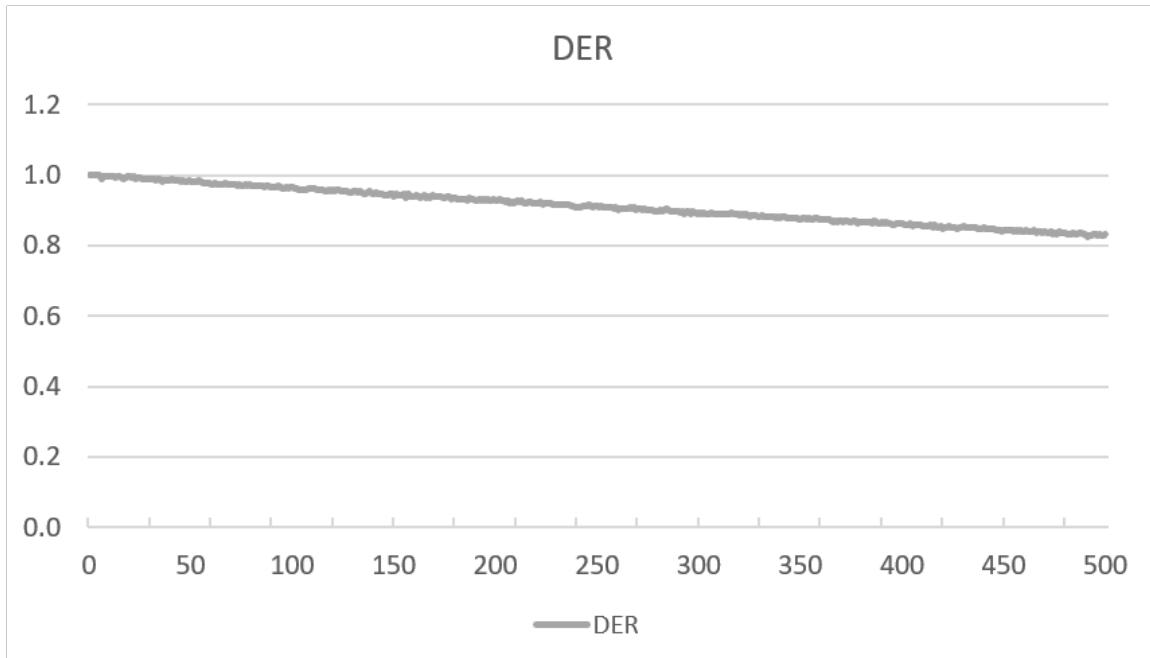
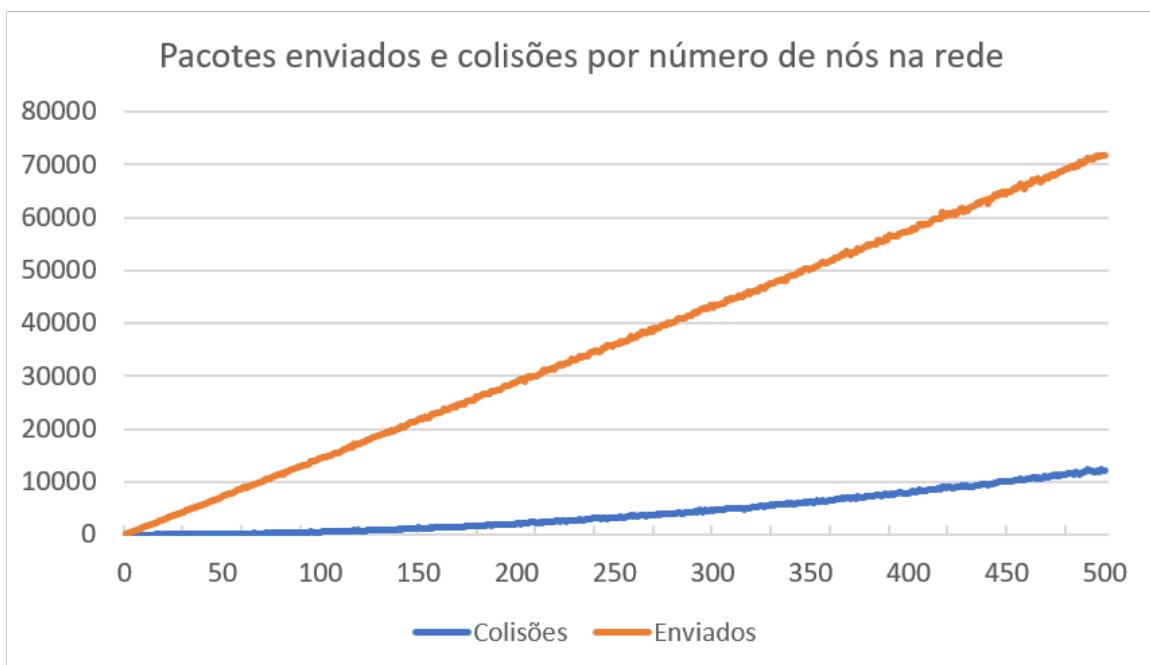
Figura 31 – *Data Extraction Rate* para cada simulação

Figura 32 – Pacotes transmitidos e pacotes que sofreram colisão para cada simulação

É importante ressaltar que LoRa deve ser utilizado em aplicações onde a perda de pacotes é aceitável (92), cujos dados dos sensores não requerem uma atualização frequente. Ou seja, é esperado perda de dados e isso não deve comprometer o funcionamento da rede. Estudos sobre colisão de dados e escalabilidade de rede LoRa (91, 84, 90) utilizam uma margem aceitável de taxa de perdas. No caso dos estudos citados foi utilizado $DER = 0,9$, o que resulta em 10% dos pacotes sendo perdidos. Utilizando o mesmo parâmetro na

EstAcqua, vemos pela Figura 31 que pode-se ter aproximadamente 270 nós na rede. Vale ressaltar que esse número leva em conta somente a taxa de colisões.

6 Arquitetura de *Hardware e Software*

A arquitetura da EstAcqua é dividida em quatro partes: um ou mais nós transmissores de dados, receptor de dados, nuvem de Internet das Coisas (*IoT Cloud*) e interação com usuário. A Figura 33 ilustra essas quatro partes.

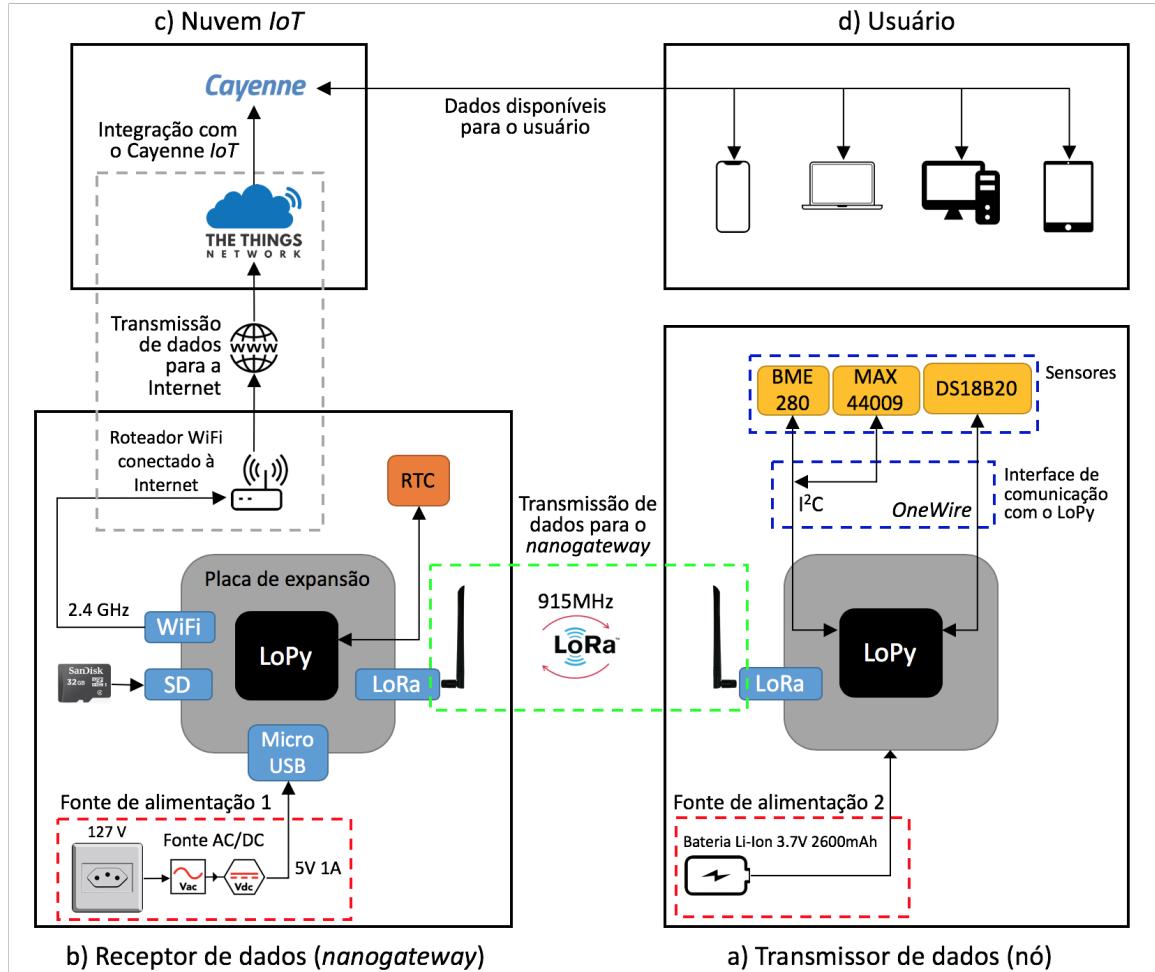


Figura 33 – Arquitetura da EstAcqua

Nela é demonstrado o nó conectado aos sensores BME280, MAX44009 e DS18B20. Os dados lidos dos sensores são empacotados e transmitidos para o *nanogateway*. O *nanogateway*, ao receber os dados enviados pelo nó, faz uma cópia de segurança em um cartão microSD conectado a ele. Posteriormente o dado é encaminhado para a nuvem *IoT*, nesse caso a TTN, e com a integração da TTN com o *Cayenne*, esse dado chega até o servidor de aplicação, onde é disponibilizado para o usuário.

6.1 Hardware

A arquitetura de *hardware* é composta pelos blocos Transmissor de dados (*nanogateway*) e Receptor de dados, representados na Figura 33:

- Receptor de dados (*nanogateway*): responsável por receber os pacotes enviados pelos Transmissores de dados. É alimentado por uma fonte de alimentação com saída DC de 5 V e 1 A e está conectado à Internet para encaminhar os dados recebidos para um servidor na nuvem *IoT*, conforme figura.
- Transmissor de dados (nó): responsável pela coleta e processamento dos dados obtidos dos sensores. Possui sensores de pressão, umidade, temperatura e iluminância situados na superfície, e sensores submersos de temperatura. É alimentado por uma bateria de 2.600 mAh com saída de 3,7 V.

Tanto o *nanogateway* como cada nó utilizam um microcontrolador LoPy4 pois já possui integrado *Wi-Fi*, LoRaWAN e *Bluetooth*. Além disso o LoPy4 possui um modo de economia de energia, *deep sleep*, em que um co-processador ULP (*Ultra Low Power*) é utilizado para monitorar as portas de entrada/saída do LoPy4, os canais do conversor analógico-digital e controlar a maioria dos periféricos internos com um consumo baixíssimo de energia. Adicionalmente é utilizada uma placa de expansão que possui entradas USB, cartão de memória e conector para a bateria, facilitando a interface com o LoPy4.

O nó é responsável pela coleta e processamento os dados dos sensores conectados a ele e pela transmissão via LoRaWAN para o *nanogateway*. Essa unidade está conectada a sensores ambientais e oceanográficos, sendo eles: (i) sensor BME280 para coletar pressão atmosférica, umidade e temperatura; (ii) sensor BH1750 (posteriormente substituído pelo MAX44009) para medir iluminância; (iii) sensores de temperatura DS18B20 para coletarem a temperatura da água ao longo de diferentes profundidades.

Os sensores BME280 e BH1750 utilizam interface de comunicação *I²C* e os sensores oceanográficos submersos, DS18B20, utilizam interface de comunicação *OneWire*. O protótipo do nó construído possui um BME280, um BH1750 e diversos DS18B20 para medir a temperatura em diferentes profundidades desde a superfície a até 15 m.

O *nanogateway* recebe os pacotes transmitidos pelos nós (Transmissores de dados) e envia para uma nuvem *IoT* para que os dados fiquem acessíveis de forma *online*. Como o *nanogateway* fica ativo o tempo todo, ele deve ser ligado à energia elétrica em um local próximo a uma rede sem fio com acesso à Internet. Ainda conta com um DS3231 via *I²C*, um *Real Time Clock* (RTC) com bateria externa, regulado por cristal de quartzo e com compensação de ano bissexto, para que o sistema mantenha seu horário sempre sincronizado, mesmo em caso de interrupções na alimentação. A nuvem *IoT* escolhida foi a TTN, por se tratar de uma comunidade colaborativa com mais de 35 mil membros e

quase 20 mil aplicações desenvolvidas, além de facilitar a integração dos dados recebidos com bancos de dados, HTTP e *Cayenne*.

Para diminuir as chances de perda de dados, o *nanogateway* possui um cartão micro SD de 32 GB de armazenamento onde são escritos todos os dados recebidos - até mesmo os dados enviados para a nuvem IoT.

Na Figura 34 são mostrados: o nó coletando a temperatura em diferentes profundidades do lago Terra Alta, bem como a temperatura, umidade e pressão do ambiente; o *nanogateway*; e microcontrolador LoPy(4) com Placa de Expansão utilizados.

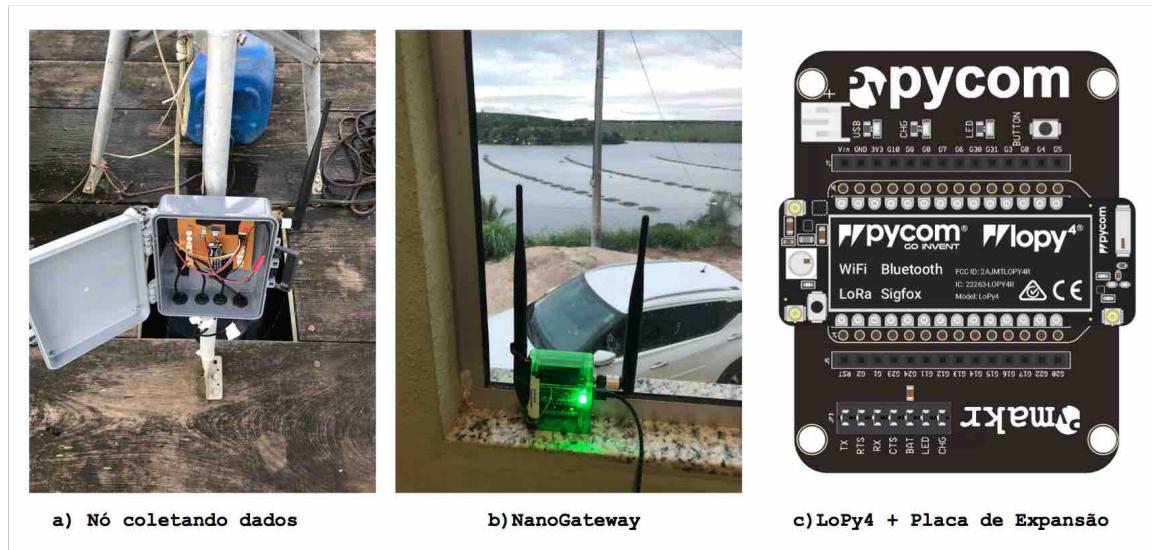


Figura 34 – *Hardware* da EstAcqua

6.2 Software

Com relação ao *software* embarcado na EstAcqua, ele é composto por dois códigos-fonte *MicroPython*¹ distintos, ambos criados pelos autores, sendo um específico para o *nanogateway*, com tratamento dos dados recebidos e encaminhamento para a TTN, e outro para o nó, para a leitura dos dados dos sensores e empacotamento para serem transmitidos para o *nanogateway*.

O nó possui um código adaptativo que reconhece automaticamente quando algum sensor é conectado ou desconectado. Dessa forma, é possível adicionar novos sensores (DS18B20, BME280 e MAX44009) e eles entram em funcionamento sem necessidade de alteração do código. O inverso também acontece, quando algum sensor é desconectado, o código se adapta e continua seu funcionamento. Para obter uma melhor leitura dos sensores, para cada sensor são efetuadas 15 leituras e então é obtida a média dessas leituras.

¹ <https://www.micropython.org>

Como cada nó foi projetado para ser instalado em um local de difícil acesso, sendo alimentado por uma bateria, foi necessário otimizar o código-fonte para que se tenha o menor consumo de energia, alta confiabilidade e seja capaz de funcionar por longos períodos de tempo com manutenção mínima. Para reduzir o tempo no modo ativo, foi configurado para realizar as conversões de temperatura dos sensores DS18B20 em paralelo, conseguindo um tempo total de aproximadamente 1,1 s para carregar os *drivers* dos sensores e realizar as leituras. O *driver* do sensor MAX44009 foi desenvolvido para uso na EstAcqua, primando ocupar pouco espaço e realizar a leitura da iluminância de forma confiável. Para isso foram realizados testes de leitura de iluminância com o *driver* desenvolvido.

Após a instalação e inicialização do nó, ele entra no modo ativo, que dura aproximadamente 3,5 s, onde (i) realiza as medições; (ii) compacta os dados para a criação de um pacote no formato *Cayenne LPP*²; (iii) envia o pacote para o *nanogateway*; (iv) e entra em modo *deep sleep*, onde permanece por 10 min. Após esse tempo ele acorda novamente e repete o processo.

Já o *nanogateway* possui um código para receber pacotes apenas dos nós que se autenticam através de *Authentication By Personalisation* (ABP). Após receber um pacote, os dados recebidos são gravados em um cartão micro SD e enviados para a TTN para visualização dos dados de forma *online*, tanto via *web* ou através do celular usando o aplicativo *Cayenne*, conforme ilustrado na Figura 33.

O código utilizado na EstAcqua, tanto o para o nó quanto para o *nanogateway*, os *drivers* desenvolvidos e/ou modificados dos sensores BME280, MAX44009 e DS18B20 estarão disponíveis no *GitHub* criado para o projeto da EstAcqua (93).

² <https://www.thethingsnetwork.org/docs/devices/arduino/api/cayennelpp.html>

7 Protótipo

Durante o processo de construção da EstAcqua, vários testes foram realizados para analisar possíveis melhorias e também testar se as modificações realizadas estavam corretas. Dessa forma, até chegar em seu formato final, a EstAcqua teve cinco versões.

A primeira versão foi construída para aprender sobre comunicação via LoRa. Foi então montada em uma placa de ensaio, *breadboard* em inglês, o microcontrolador LoPy4 junto com uma placa de expansão conectada a uma bateria de polímero de lítio de 500 mAh, e os sensores de iluminância, BH1750, e de temperatura, pressão e umidade, BME280. A Figura 35 mostra o protótipo montado.

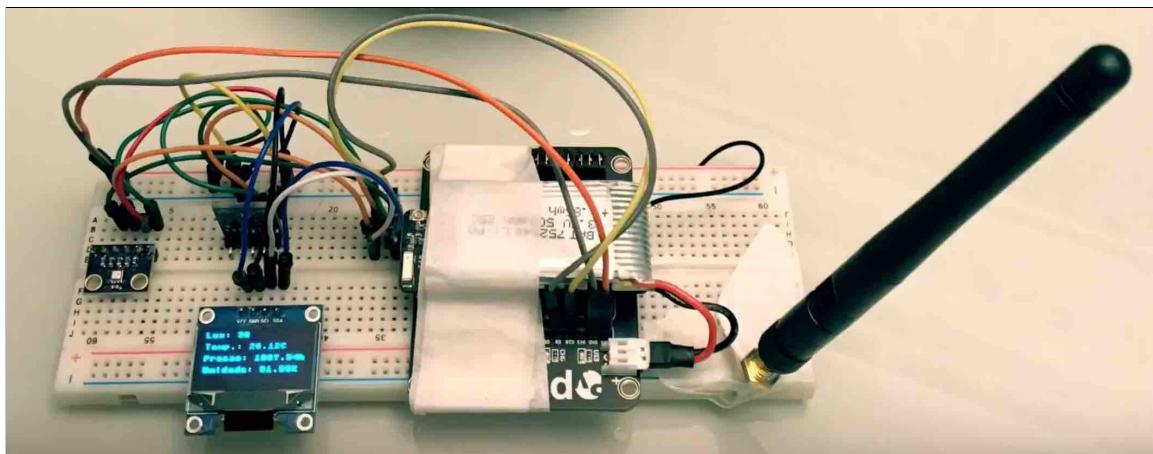


Figura 35 – Primeira versão da EstAcqua

O propósito dessa primeira versão era testar a transmissão entre dois dispositivos usando LoRa puro. Ou seja, sem utilizar LoRaWAN, nuvem *IoT* e servidor de Aplicação. Os dados dos sensores eram lidos e transmitidos para outro LoPy4. Como forma de *debug*, foi instalada tela OLED de 1,3" para que pudessem ser vistos os dados lidos pelos sensores. No outro LoPy4 (receptor dos dados), também havia uma tela OLED que mostrava os dados recebidos. Dessa forma era possível verificar se a transmissão havia ocorrido de forma correta.

Após ter dominado a comunicação via LoRa puro, o próximo desafio foi colocar a EstAcqua em uma rede usando LoRaWAN. Foi configurado o *gateway* para se autenticar na *The Things Network* e transmitir os dados para a aplicação *Cayenne*. Nesse estágio também foram adicionados os sensores de temperatura subaquática, DS18B20. Foi então realizado o primeiro teste da EstAcqua dentro da UFES. O *gateway* foi colocado no prédio do Departamento de Oceanografia e EstAcqua em uma lagoa situada a 128 m de distância, conforme mostrado na Figura 36.

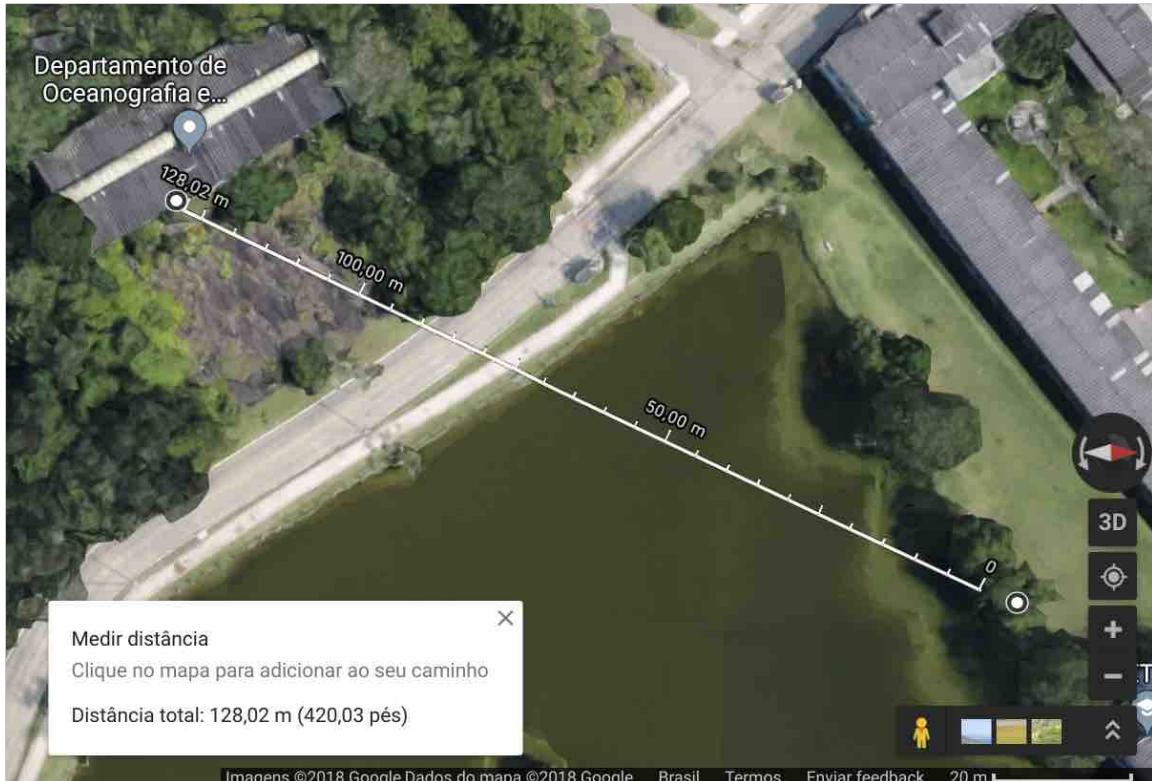


Figura 36 – Lagoa da UFES situada a 128 m de distância do Departamento de Oceanografia.

Para esse teste, a EstAcqua saiu do *breadboard* e foi colocada dentro de uma Caixa Patola PB-108, contendo botão liga/desliga, LED¹ de funcionamento, sensores DS18B20, BME280 e BH1750, e tela OLED. A Figura 37 mostra a segunda versão da EstAcqua coletando dados na lagoa da UFES.

Como o teste teve um resultado positivo, ou seja, os dados coletados foram transmitidos para o *gateway*, passando pela nuvem *IoT* e chegando na aplicação, foi realizado o primeiro teste com a EstAcqua no lago Terra Alta. A Figura 38 mostra a EstAcqua instalada na balsa situada no lago Terra Alta.

Neste primeiro teste no lago Terra Alta os dados enviados para o *gateway* não foram transmitidos para a nuvem *IoT*. Tanto a EstAcqua quanto o *gateway* gravaram os dados transmitidos e recebidos, respectivamente, em cartões micro SD e posteriormente foram analisados para verificar se a transmissão ocorreu sem problemas. Nessa primeira visita também foi realizado um *site survey* para verificar possíveis pontos de instalação do *gateway* e também verificar sobre a Internet disponível no local.

Após o primeiro teste no lago Terra Alta ter sido um sucesso, a EstAcqua foi novamente modificada. Em sua terceira versão foi removida a tela OLED e os sensores BME280 e BH1750 foram posicionados dentro da caixa estanque de uma câmera GoPro²,

¹ https://en.wikipedia.org/wiki/Light-emitting_diode

² <https://pt.shop.gopro.com/International/accessories-2/super-suit-über-protection-plus-dive-housing-for>



Figura 37 – EstAcqua V2 coletando dados na lagoa situada dentro da UFES

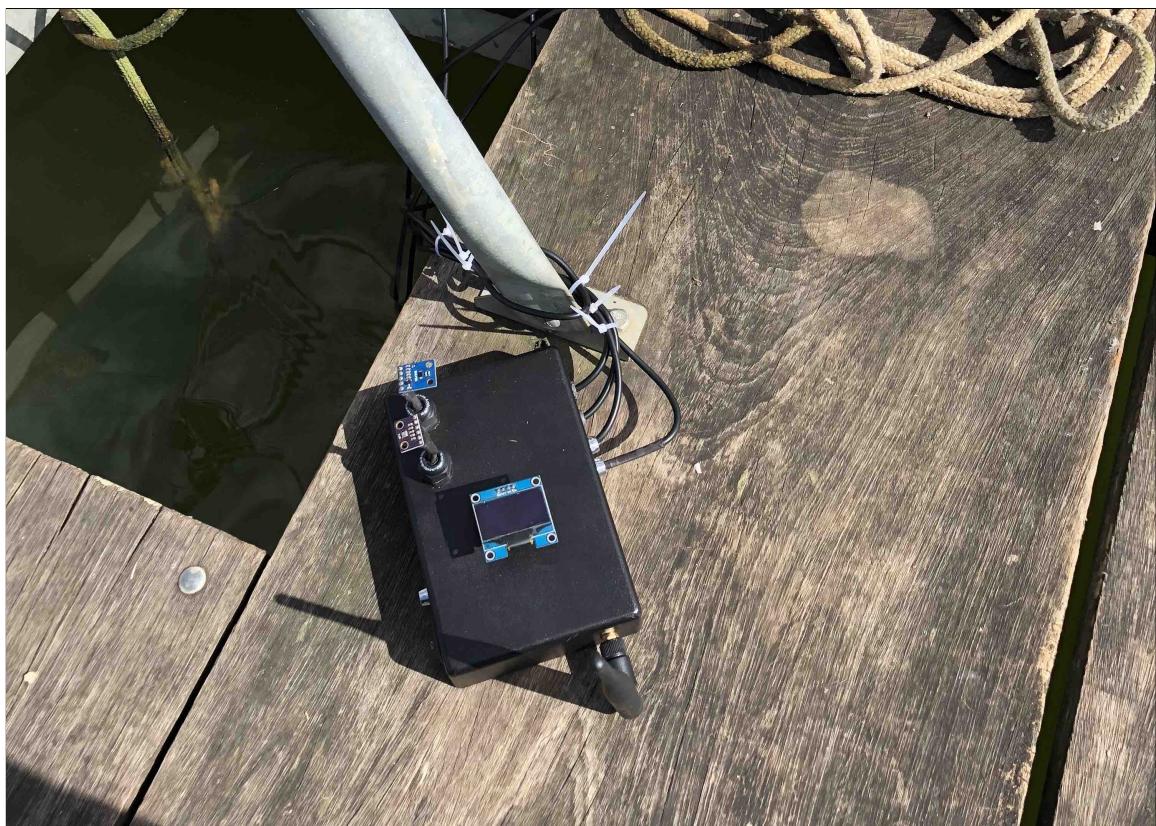


Figura 38 – EstAcqua V2 instalada na balsa no lago Terra Alta

conforme mostrado na Figura 39.



Figura 39 – Terceira versão da EstAcqua sendo testada na lagoa da UFES

O intuito desse teste era verificar o comportamento da aplicação *Cayenne* a longos períodos de funcionamento da EstAcqua. Para isso, a EstAcqua foi posicionada na lagoa da UFES e durante 6 h contínuas transmitiu os dados para o *gateway* situado no Departamento de Oceanografia. Todos os dados foram recebidos no *Cayenne*, possibilitando que gráficos fossem gerados para a análise dos dados coletados. A Figura 40 mostra o gráfico de umidade relativa do ar gerado durante esse teste.

Com o protótipo funcionando de forma satisfatória tanto no quesito *hardware* e *software*, foi realizado um segundo teste no lago Terra Alta. O protótipo foi separado em duas unidades: principal e secundária. Na unidade principal encontra-se o LoPy4, bateria, chave liga/desliga e antena. A unidade secundária é conectada à unidade principal através de um cabo, e nela estão os barramentos I^2C e *1-Wire*, de onde saem os sensores. A Figura 41 mostra a quarta versão do protótipo instalada na balsa no lago Terra Alta.

A caixa utilizada nas unidades principal e secundária possui uma borracha para vedação, tornando-a resistente à chuva. Nesse segundo teste o *gateway* estava conectado à Internet e os dados coletados pela EstAcqua chegavam até a aplicação final. Choveu durante o teste e a vedação das unidades principal e secundária resistiu à chuva.



Figura 40 – Gráfico de umidade gerado pelo *Cayenne*

Na quinta versão do protótipo da EstAcqua foi utilizada uma caixa certificada IP56 (resistente a chuva e poeira). Foram realizados ajustes finos para melhorar a autonomia da bateria. O sensor de iluminância, BH1750, foi substituído pelo sensor de iluminância MAX44009 por se tratar de um sensor com um fundo de escala maior que o BH1750 e consumir menos bateria no modo *deep sleep*. A placa de expansão utilizada no nó foi substituída por uma placa de circuito impresso fabricada pelo próprio autor. A Figura 42 mostra a placa construída para a EstAcqua.

A placa de circuito impresso possui dois barramentos, uma para I^2C e outro para *1-Wire*, chave liga/desliga, ponta de teste para consumo de corrente, circuito divisor de tensão para estimar a tensão da bateria e suporte para encaixe do LoPy4. Após essas melhorias, o consumo em modo ativo ficou em 50 mA e no modo *deep sleep* ficou somente em 94,8 μA , conforme mostrado na Figura 43.

A Figura 44 mostra a versão final da EstAcqua instalada no Lago Terra Alta. Ela foi posicionada de forma que a antena ficasse 50 cm acima da balsa. Como a superfície da balsa situa-se a 15 cm de distância da água do lago, a antena encontra-se a 65 cm de distância do nível da água do lago.

A caixa IP56 foi modificada para encaixar uma peça plástica que servirá de abrigo para os sensores de iluminância e de pressão, temperatura e umidade. Para isso foi realizado um corte circular, encaixada a peça plástica e depois vedado com silicone de forma a manter as características de resistência a poeira e chuva.

Após a instalação da EstAcqua na balsa, o *gateway* foi instalado dentro da casa, situada a aproximadamente 200 m de distância da balsa. A EstAcqua coletou dados do



Figura 41 – Quarta versão da EstAcqua instalada na balsa no lago Terra Alta

lago e do ambiente por 1 mês, com uma frequência de 10 min. A análise dos dados será apresentada no Capítulo 8.

7.1 Software

Os softwares utilizados no *NanoGateway* e no nó da EstAcqua encontram-se nos apêndices B e C, respectivamente. Sendo que o *software* utilizado no *NanoGateway* foi uma modificação realizada pelo autor da versão oficial disponibilizada pela Pycom³ e o *software* do nó foi completamente desenvolvido pelo autor.

³ <https://github.com/pycom/pycom-libraries/blob/master/examples/lorawan-nanogateway/nanogateway.py>

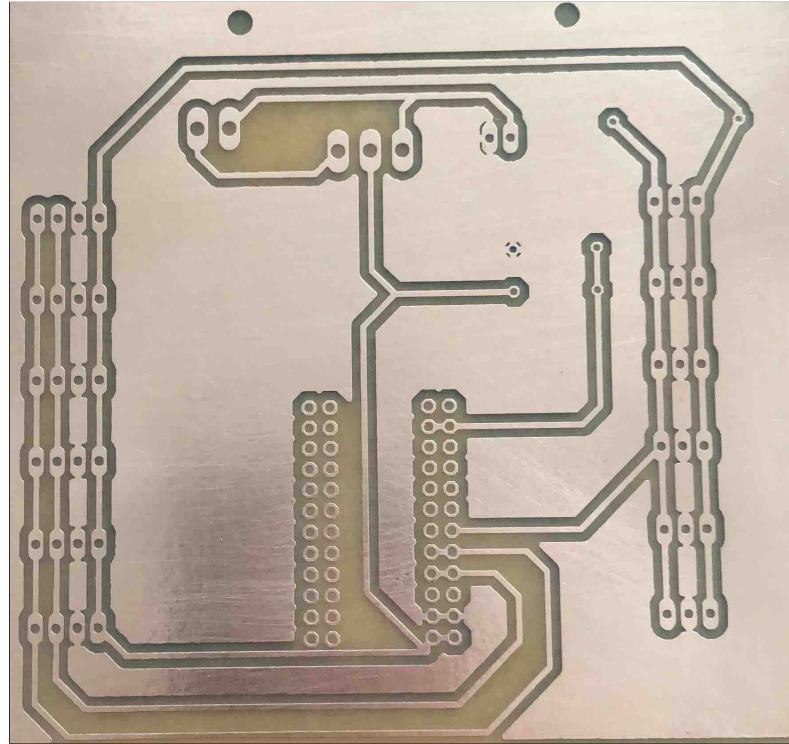


Figura 42 – Placa de circuito impresso desenvolvida para a EstAcqua



Figura 43 – Consumo de $94.8 \mu\text{A}$ no modo *deep sleep*

Durante todo o desenvolvimento da EstAcqua, o *software* utilizado no *NanoGateway* sofreu pequenas modificações, pois se tratava, basicamente, de um encaminhador de pacotes para a TTN. Uma de suas principais funções - criadas pelo autor - é a função de verificar se existe um cartão de memória inserido no *NanoGateway* e se já existe um arquivo de log para armazenar os dados recebidos. Essa função se encontra abaixo, onde pode-se perceber



Figura 44 – Quinta versão da EstAcqua instalada na balsa no Lago Terra Alta

que ela captura exceções e se adapta de acordo com a situação, não interrompendo a inicialização do *NanoGateway*.

```

1 try :
2     sd = SD()
3     os.mount(sd, '/sd')
4     print("microSD card mounted with free {:.3f} GB".format(os.getfree('/sd'
5     ')/(1024*1024)))')
6     flaguSDcard = True
7 except OSError as e:
8     pycom.rgbled(0xFF0000)
9     print("Caught exception while mounting microSD. {}".format(e))
10    flaguSDcard = False
11
12 try :
13     f = open('/sd/recv.txt', "r")
14     exists = True
15     f.close()

```

```

15 except OSError as e:
16     exists = False

```

Outra parte que merece destaque é a modificação realizada pelo autor que define como o *NanoGateway* deve se comportar ao receber um dado.

```

1 if events & LoRa.RX_PACKET_EVENT:
2     self.rxbn += 1
3     self.rxok += 1
4     rx_data = self.lora_sock.recv(256)
5     stats = lora.stats()
6     packet = self._make_node_packet(rx_data, self.rtc.now(), stats.
7         rx_timestamp, stats.sfrx, self.bw, stats.rssi, stats.snr)
8     # pacote recebido! pisca o led em azul para mostrar que o
9     # NanoGateway recebeu o dado
10    # mesmo que não esteja conectado na TTN ele irá salvar o dado
11    # no cartão de memória
12    # e esse dado depois pode ser recuperado
13    pycom.rgbled(0x0000ff)
14    utime.sleep_ms(100)
15    pycom.rgbled(0x00ff00)
16    # se o cartão estiver montado ele grava o pacote recebido
17    if flaguSDcard:
18        logfile = open('/sd/recv.txt', 'a')
19        logfile.write('{}\n'.format(packet))
20        logfile.close()
21        os.sync()
22        self._push_data(packet)
23        self._log('Received packet: {}', packet)
24        self.rxfw += 1

```

Como pode ser observado, ao receber um pacote, o *NanoGateway* pisca um led na cor azul para demonstrar que recebeu um pacote (um modo simples de verificar que um pacote foi recebido sem a necessidade de estar conectado via diretamente ao *NanoGateway*). Essa ação é efetuado mesmo que o *NanoGateway* não esteja conectado à TTN. Após receber o dado, caso o cartão microSD tenha sido corretamente montado, será efetuado o registro desse pacote recebido no cartão microSD.

Conforme dito anteriormente, o *software* do nó possui uma capacidade de auto adaptação em relação aos sensores conectados a ele. Ou seja, caso algum sensor se adicionado (desde que faça parte dos sensores conhecidos pela EstAcqua), o código irá iniciar e realizar a leitura do sensor adicionado. Da mesma forma, caso algum sensor seja removido, o código identificará que esse sensor não se encontra mais conectado a EstAcqua e não causará uma interrupção no funcionamento da mesma. O código abaixo mostra a parte responsável por essa auto configuração. Vale ressaltar que ele foi desenvolvido para funcionar com os sensores citados anteriormente, mas pode ser facilmente modificado para

adicionar novos sensores.

```

1 if len (connectedI2C) > 0:
2     connected_i2c = True
3
4 if connected_i2c:
5     for device in connectedI2C:
6         if device == 0x4A:    # MAX44009 - 74
7             light_sensor = max44009.MAX44009(i2c)
8             light_s = True
9         elif device == 0x76:   # BME280 - 118
10            bme = bme280.BME280(i2c=i2c, pressure_mode=bme280.OSAMPLE_8,
11                           iir=bme280.FILTER_8)
12            bme_s = True
13        else:
14            if flagDebug:
15                print("I2C nao reconhecido")    # Dispositivo nao cadastrado
16
17 if ow_s:
18     count = 0
19     for sensors in connectedOW:
20         temp.start_conversion(temp.roms[count])
21         count += 1
22
23 if light_s:
24     # Le iluminancia em lux do MAX44009
25     data = int(light_sensor.illuminance_lux)
26     illum.append(data)
27
28 if bme_s:
29     # Le valores BME280 com media para ter maior precisao :
30     numreadings = 15
31     samples_temperature = [0.0]*numreadings; mean_temperature = 0.0
32     samples_pressure = [0.0]*numreadings; mean_pressure = 0.0
33     samples_humidity = [0.0]*numreadings; mean_humidity = 0.0
34     for i in range (numreadings):
35         samples_temperature[i], samples_pressure[i], samples_humidity[i] =
36         bme.values
37         mean_temperature = sum(samples_temperature)/len(samples_temperature)
38         mean_pressure = sum(samples_pressure)/len(samples_pressure)
39         mean_humidity = sum(samples_humidity)/len(samples_humidity)
40         t = mean_temperature
41         p = mean_pressure/100    # Pa -> hectoPa
42         h = mean_humidity
43         bmet.append(t)
44         bmep.append(p)
45         bmeh.append(h)
46
47 if ow_s:
```

```

45     count = 0
46     for sensor in connectedOW:
47         tempOW = temp.read_temp_async(temp.roms[ count ])
48         owTemp.append(tempOW)
49         count += 1

```

Na versão final da EstAcqua haviam somente sensores conectados via barramento I^2C e 1-Wire. Dessa forma a primeira verificação é se existem sensores conectados ao barramento 1-Wire. Como somente um tipo de sensor utilizado possui esse barramento (DS18B20), basta apenas identificar a quantidade de sensores conectados. Já no caso do I^2C , caso existam sensores conectados nesse barramento, é necessário identificar o tipo de sensor. Uma vez identificado, é setada uma *flag* para identificar quais sensores se encontram conectados. Por último, é realizada a leitura dos sensores conectados e seus valores são colocados em uma lista que posteriormente será colocada dentro de um pacote LPP para ser enviado à TTN.

Outra função importante do nó é a responsável por identificar a tensão na bateria. Para que fosse possível realizar essa leitura foi necessário realizar uma modificação no *hardware* do nó para inclusão de um divisor de tensão entre a bateria e o pino 16⁴ do LoPy. Foram realizados cálculos para a escolha dos resistores do divisor de tensão de forma que, quando a bateria estivesse completamente carregada (3.7 V), a tensão na entrada do pino 16 não fosse superior a 1.1 V. Depois é aplicada uma fórmula para ajustar a leitura do ADC para a bateria de 3.7 V e, por fim, retorna o valor da tensão da bateria em mV.

```

1 # ADC 12 bits
2 # Conversao para mV
3 # Retorna o valor da tensao da bateria em mV
4 # Divisor de tensao: R1=680k, R2=100k
5 # ADC Pino 16, (input only; max voltage 1.1 V)
6
7 def get_batt_mV():
8     numADCreadings = const(100)
9
10    adc = ADC(0)
11    adcread = adc.channel(pin='P16')
12    samplesADC = [0.0]*numADCreadings; meanADC = 0.0
13    i = 0
14    while (i < numADCreadings):
15        adcint = adcread()
16        samplesADC[i] = adcint
17        meanADC += adcint
18        i += 1
19    meanADC /= numADCreadings
20

```

⁴ Pino somente entrada com tensão máxima de 1.1V e possui um ADC de 12 bits

```

21 mV = ((meanADC*1100/4096)*(680+100)/100)
22 mV_int = int(mV)
23 return mV_int
  
```

A Figura 45 mostra o diagrama lógico do software no nó mostrando suas interações desde sua inicialização até a transmissão do pacote e entrada no modo *deepsleep*.

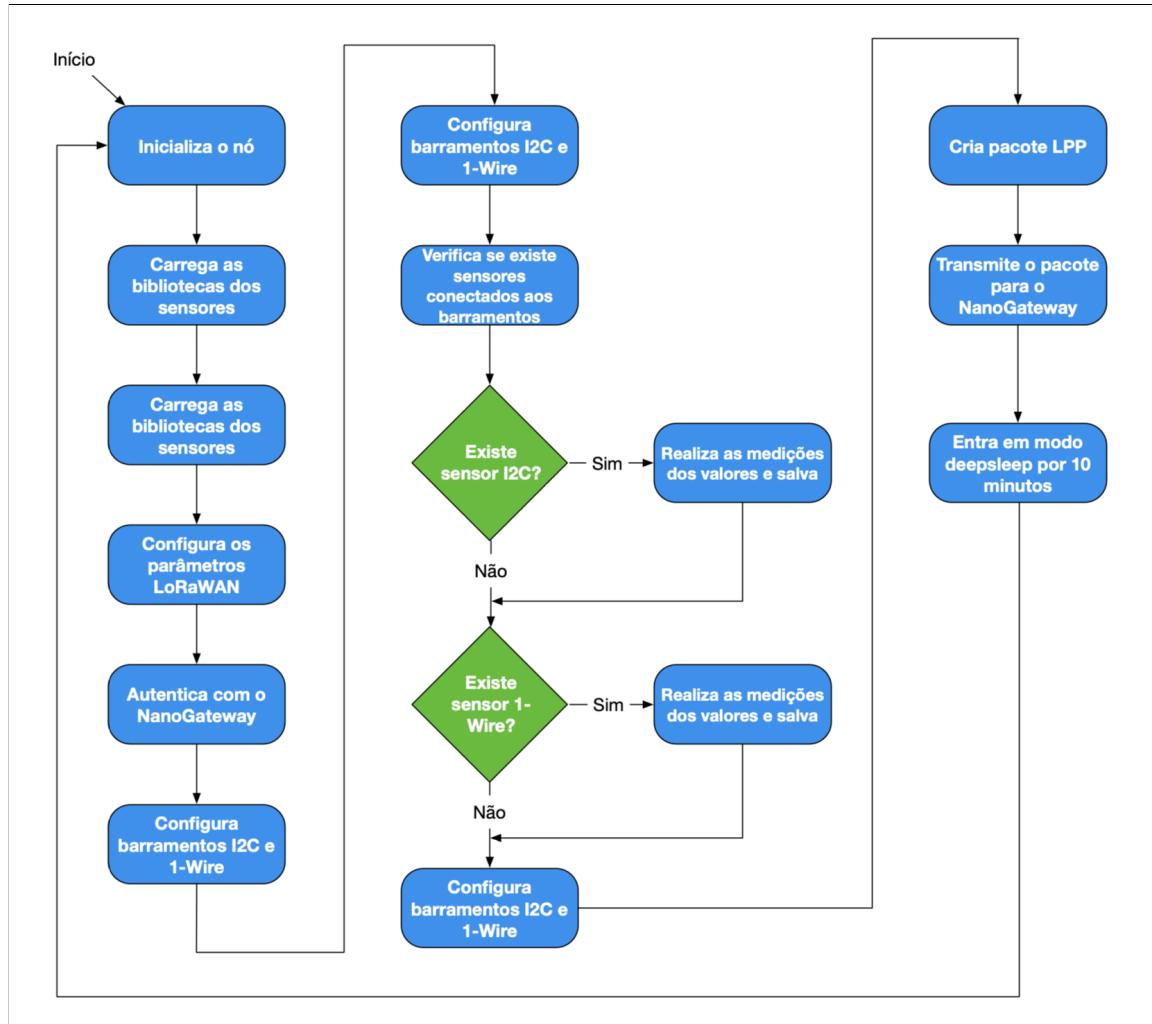


Figura 45 – Diagrama Lógico do funcionamento do nó

8 Avaliação e Resultados

É importante ressaltar que os testes realizados possuem o intuito de mostrar que a modularidade da EstAcqua permite que diferentes configurações possam ser implementadas, ou seja, diferentes configurações de sensores podem ser acoplados a EstAcqua. A configuração testada foi escolhida por contemplar sensores atmosféricos e oceanográficos.

Para avaliar as funcionalidades e atendimento aos requisitos do processo de coleta, transmissão e processamento dos dados do protótipo, foram realizados os seguintes testes: (i) comunicação via LoRaWAN, com o intuito de verificar como o protótipo se comporta em diferentes situações e decidir quais parâmetros utilizar na comunicação; (ii) interface de monitoramento com a nuvem, a fim de analisar como os dados enviados pelo nó são enviados para o *IoT Cloud* e são disponibilizados para o usuário; (iii) avaliação no lago Terra Alta, visando analisar o comportamento do protótipo em locais de difícil acesso e sem comunicação com a Internet; (iv) autonomia da bateria, objetivando calcular um valor teórico do tempo sem necessidade de recorrer a uma fonte externa de energia para recarregá-la e; (v) avaliação dos sensores utilizados, comparando-os com sensores comerciais.

8.1 Interface de Monitoramento via Nuvem

Foram realizados testes em campo na lagoa situada dentro da Universidade Federal do Espírito Santo (UFES), monitorando a temperatura da lagoa em diferentes profundidades, pressão, umidade, temperatura e iluminância do ambiente. Os dados foram enviados para o *nanogateway* localizado no Departamento de Oceanografia, a 130 m distância da lagoa e 11 m de altura, estando conectado à Eduroam (94) - rede sem fio na UFES. O nó foi posicionado próximo à lagoa e bastou ligá-lo para que os dados fossem transmitidos para o *nanogateway*, ratificando assim a sua fácil instalação e uso.

No caso de indisponibilidade de acesso à Internet por parte do *gateway*, os dados coletados pelo nó não seriam perdidos em virtude da gravação dos dados recebidos em cartão micro SD.

Os dados transmitidos pelo nó no teste da UFES foram visualizados através do site da aplicação *Cayenne* e também no aplicativo *Cayenne* para dispositivos móveis. A Figura 46 mostra a forma com que os dados são disponibilizados de forma *online* para o usuário. Também é possível a criação de gráficos com os valores lidos pelos sensores para acompanhar a evolução dos valores e a criação de alarmes para enviar *e-mails* caso algum dos sensores esteja com leitura acima ou abaixo do valor configurado pelo

usuário. Também é possível a visualização das mesmas informações em dispositivos móveis, conforme mostrado na Figura 47.

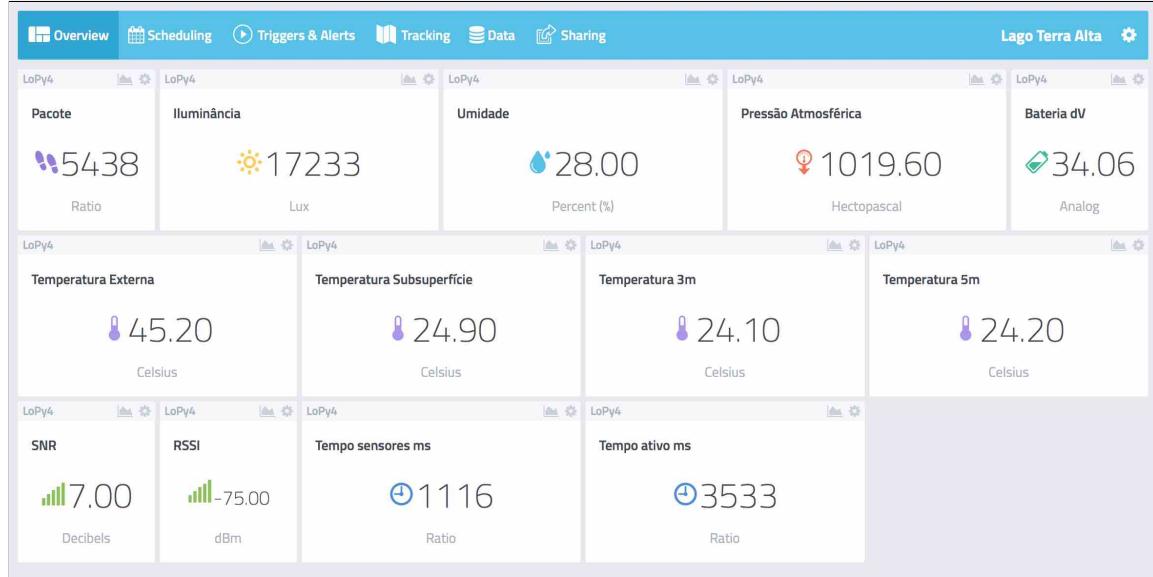


Figura 46 – Interface *Web* do Cayenne com os dados acessados *online*



Figura 47 – Interface *mobile* do Cayenne com os dados acessados *online*

As Figuras 48 e 49 mostram os gráficos de temperaturas externa e submersa na lagoa da UFES, onde pode-se perceber a inércia de variação da temperatura da água da lagoa em relação à temperatura ambiente.

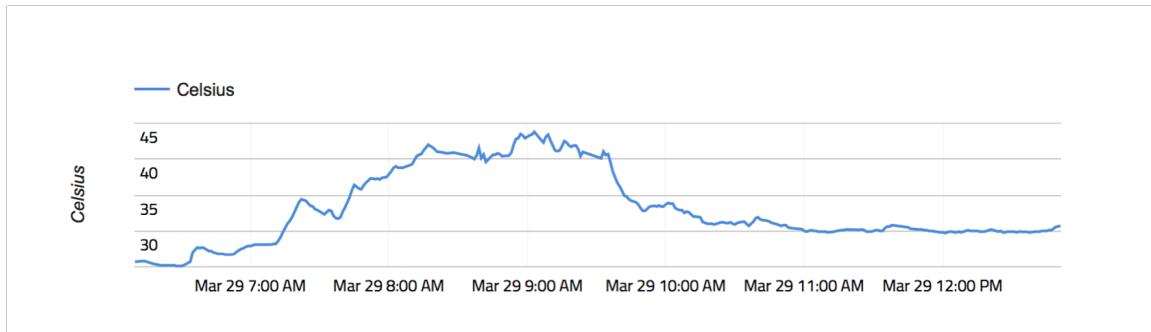


Figura 48 – Gráfico de variação de temperatura externa entre as 6 h e 13 h

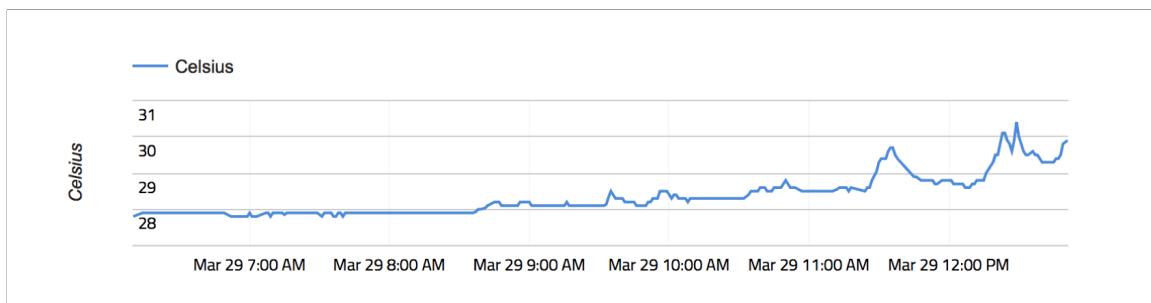


Figura 49 – Gráfico de variação de temperatura da lagoa entre as 6 h e 13 h

Também foi configurado um alarme para caso a temperatura de subsuperfície do lago estivesse com temperatura superior a 27 °C. Uma vez atingida essa temperatura, um *e-mail* é enviado para alertar sobre essa situação. A Figura 50 mostra um exemplo de *e-mail* enviado pelo *Cayenne* quando um alarme é disparado. Nela é mostrado o canal utilizado pelo sensor, a condição que fez disparar o alarme e em qual dispositivo o sensor encontra-se conectado.

8.2 Avaliação no lago Terra Alta

Outro teste foi realizado no lago Terra Alta (área de 3,9 km² e profundidade máxima de 22,1 m) por se tratar de um importante ecossistema lacustre profundo sob estresse ambiental da atividade de piscicultura (95), com produção estimada em 2017 de 550 toneladas de tilápia. Desde junho de 2014, o lago vem sendo monitorado com variáveis climáticas a cada 15 min e temperatura da água em oito profundidades a cada 60 min. O objetivo é avaliar o efeito dos fatores climáticos sobre o regime térmico do lago e propriedades hidroquímicas e hidrobiológicas. O lago Terra Alta situa-se em uma localidade remota e sem sinal de celular.

Na piscicultura existe uma balsa onde sensores ambientais já estão instalados. No lago Terra Alta, o monitoramento de variáveis limnológicas produz dados contínuos da estrutura térmica da coluna de água, conforme Figura 52. Nela é possível perceber que o

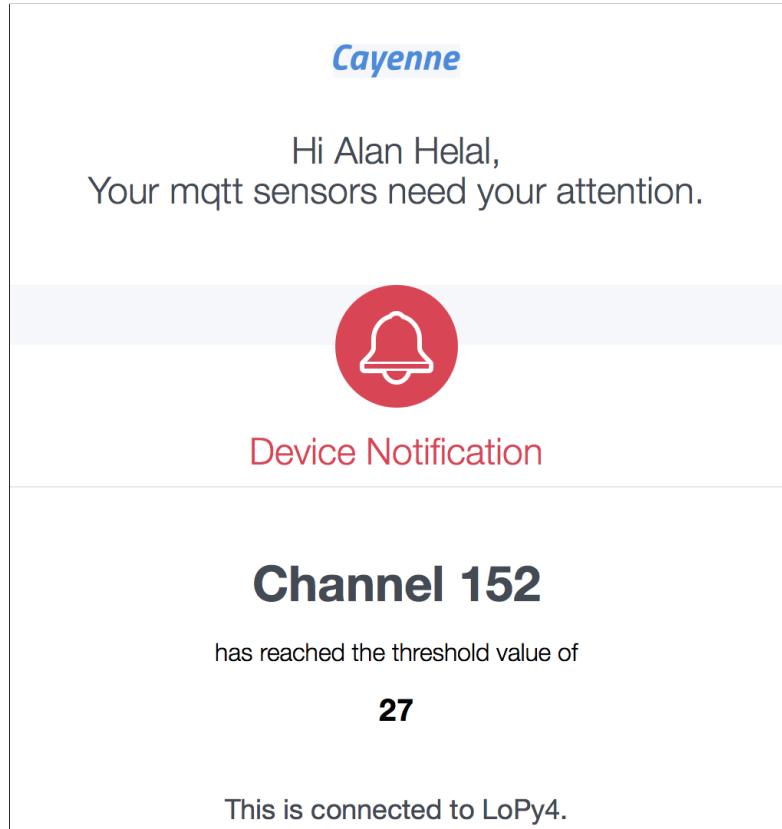


Figura 50 – Alarme enviado automaticamente pelo *Cayenne* alertando que uma condição crítica, definida pelo usuário, ocorreu.

lago Terra Alta encontra-se estratificado entre novembro e maio, quando se desestratifica, mantendo-se nesse estado entre maio e novembro. Essas informações são importantes para o estudo da influência de mudanças climáticas na estrutura térmica da coluna de água e seus reflexos para a ecologia desses ecossistemas.

A Figura 51 mostra a balsa situada no lago Terra Alta com a estação metereológica, e a EstAcqua posicionada na balsa para a coleta dos dados. Nessa figura é mostrada a Versão 2 da EstAcqua onde foram realizados testes movimentando o *nanogateway* pelo lago, chegando até sua margem localizada a 3,15 km de distância.

Esse teste, realizado em 02/03/2018, mostrou que em local remoto a instalação do protótipo foi rápida e fácil e mesmo sem comunicação com a Internet não houve perda de dados, pois os mesmos foram gravados no cartão micro SD, confirmando sua viabilidade mesmo em ambientes sem conexão com a Internet ou sinal de celular. Adicionalmente, como o *gateway* estava se movimentando em relação ao lago e não houve perda de dados, foi verificada a possibilidade de se instalar um nó em qualquer ponto do lago.

Posteriormente, com a versão final da EstAcqua construída, retornou-se ao lago Terra Alta para realização de um teste prolongado. Conforme dito anteriormente, a 200 m de distância da balsa há uma casa com rede sem fio para acesso à Internet. É abastecida

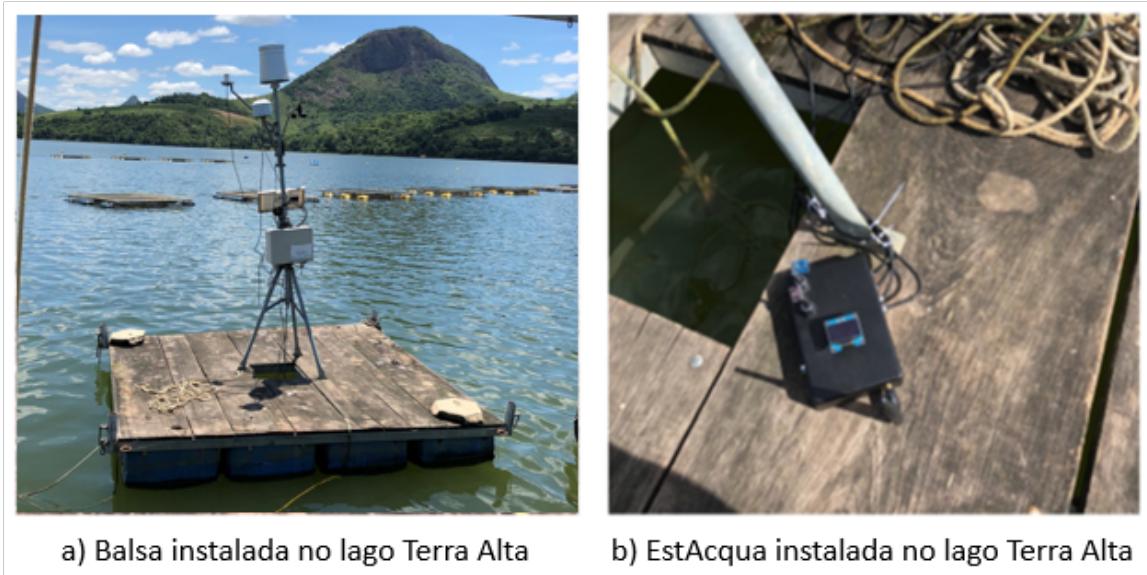


Figura 51 – Balsa localizada no lago Terra alta e o nó na balsa coletando dados

com Internet a Rádio e um link de 10 Mbps. Em termos de velocidade, essa conexão atende perfeitamente à EstAcqua, sendo possível transmitir os dados coletados para a nuvem *IoT* sem problemas. Entretanto, devido à localização rural, essa Internet a Rádio apresenta grande oscilação em seu funcionamento, apresentando desconexões frequentes. A EstAcqua foi instalada na base do tripé da estação metereológica existente na balsa, conforme mostrado na Figura 55.

Durante 1 mês, do dia 13/06/2018 até o dia 13/07/2018, foram transmitidos pacotes com os dados coletados pela EstAcqua a cada 10 minutos. Dessa forma foi possível analisar como a EstAcqua e os serviços de Aplicação e nuvem *IoT* se comportariam.

A Figura 54 mostra os gráficos de iluminância, umidade e pressão atmosférica gerados pelo *Cayenne* durante esse período de testes. Nela estão marcados quatro retângulos que representam momentos em que a Nuvem *IoT* não recebeu dados da EstAcqua. Após uma investigação do que causou esses momentos de indisponibilidade de dados, constatou-se que o problema não se encontrava na EstAcqua mas sim na combinação da oscilação da Internet e o modo como a TTN opera.

Hylke Visser (96), *Lead Stack Engineer* da TTN descreveu a causa desse problema no fórum da TTN (97):

O problema aqui é que alguns *gateways* de canal único não suportam mensagens de *downlink*. Como geralmente focamos nosso desenvolvimento em *gateways* reais (multicanais com suporte *downlink*), *gateways* de canal único e sem *downlink* são facilmente esquecidos.

Recentemente, implementamos uma otimização em nosso *back-end* que conecta os fluxos de *gateway* quando o *gateway* está ativo e desconecta quando o *gateway* se torna inativo. Como alguns *gateways* não veem muito tráfego, não podemos basear esse "estado" nas mensagens de *uplink*

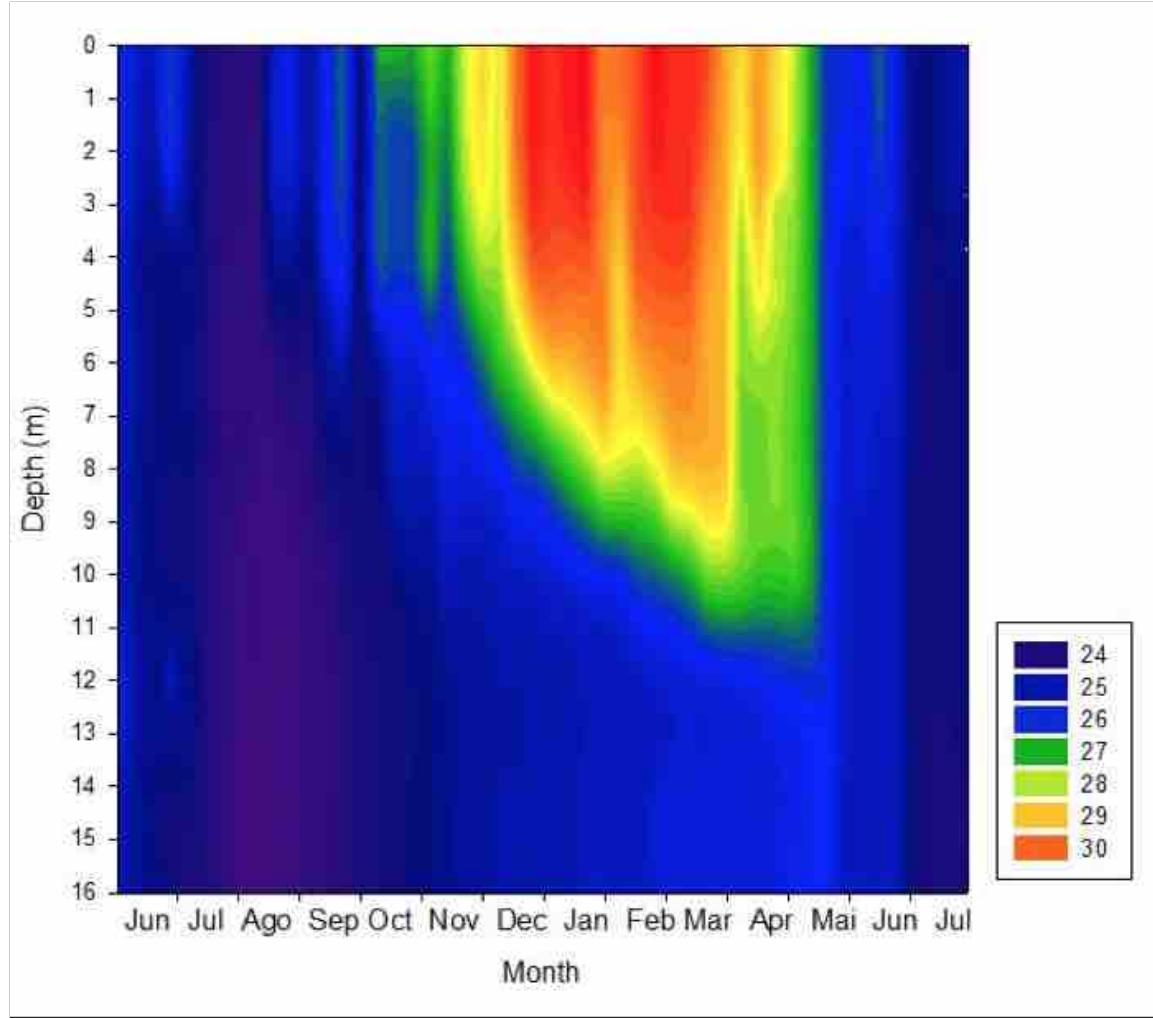


Figura 52 – Diagrama profundidade/tempo para a variação da temperatura na coluna de água do lago Terra Alta

Fonte: Barroso et al.(95)

recebidas desses *gateways*. No entanto, todos os *gateways* (exceto esses de um único canal, sem o *downlink*) fazem *ping* no servidor a cada 5 s, então decidimos usar esses *pings* para determinar se o *gateway* está ativo ou não.

Como resultado dessa escolha, os *gateways* que não "pingam" para mensagens de *downlink* são considerados "*off-line*", e isso resulta no tráfego de *uplink* recebido desses *gateways* que não está sendo tratado corretamente. Estou trabalhando em uma solução para isso. ([Hylke Visser\(97\)](#), tradução nossa).

Ou seja, uma tentativa de *uplink* do *gateway* ocorreu em um momento de indisponibilidade de Internet e devido ao método descrito acima, a TTN assume que o *gateway* se encontra desligado. Os próximos pacotes transmitidos, mesmo com a disponibilidade da Internet, não são recebidos pela TTN pois para ela o *gateway* continua no modo *off-line*. Essa situação permanece até o momento em que a TTN envia outra mensagem para o *gateway* para verificar se ele está *online*. Esse período dura por volta de 1 dia a 1 dia e

meio e, durante esse tempo, apesar dos dados não serem transmitidos para a TTN, eles foram gravados no cartão micro SD no *gateway*. Caso o local de instalação não possua uma conexão com a Internet estável, como é o caso do lago Terra Alta, uma alternativa é a utilização de um *gateway* LoRaWAN certificado, resultando em um aumento no custo do projeto, ou utilização de nuvem IoT com suporte a *gateway* mono canal.

Para a análise dos dados é possível escolher o período que se deseja analisar. Por exemplo, na Figura 53 é mostrada a variação da iluminância durante o dia 23/06/2018, possibilitando uma melhor representação devido à quantidade menor de dados.

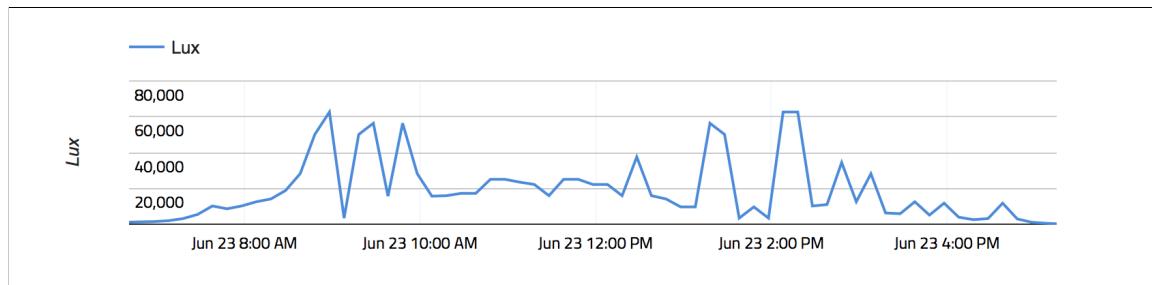


Figura 53 – Variação da iluminância ao decorrer do dia 23/06/2018

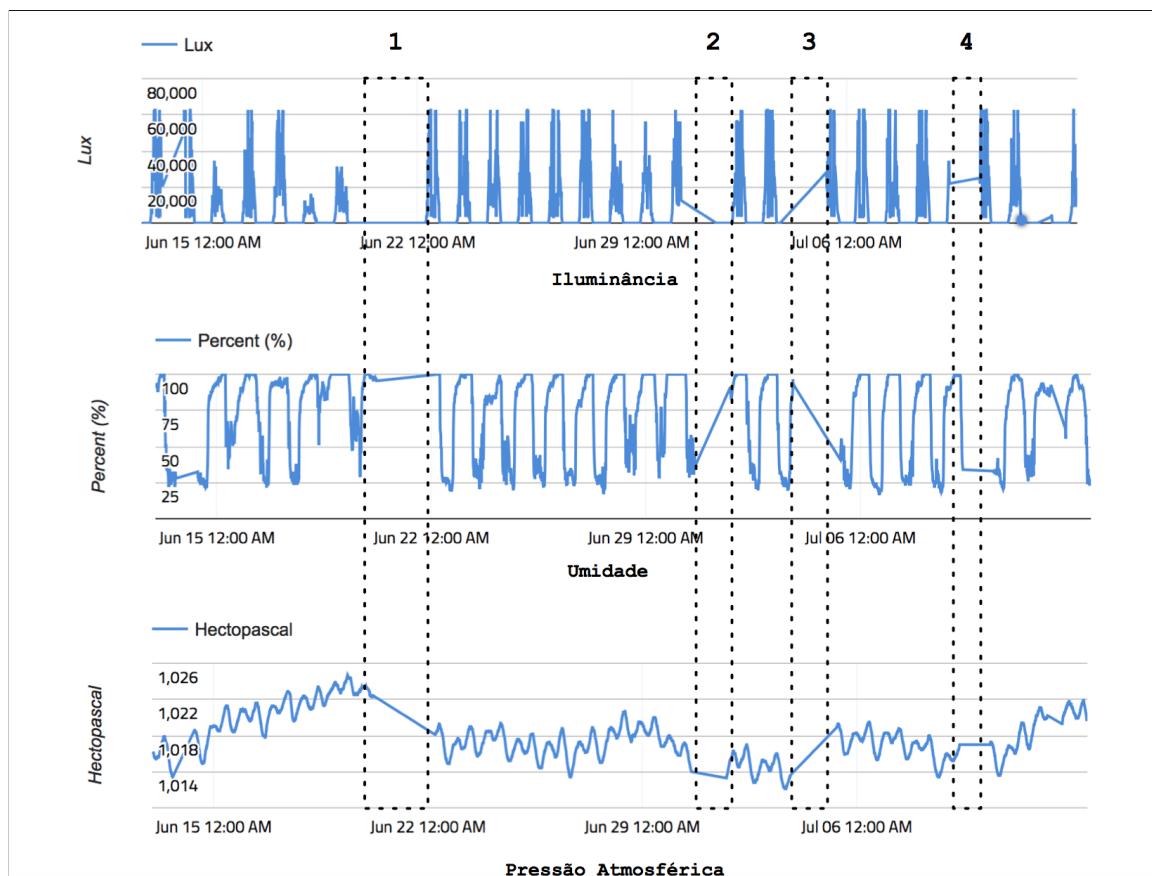


Figura 54 – Gráficos de iluminância, umidade e pressão atmosférica gerados pelo *Cayenne* no período de 13/06/2018 a 13/07/2018



Figura 55 – Versão final da EstAcqua instalada no tripé existente na balsa situada no lago Terra Alta

A nível de curiosidade, a Figura 56 mostra a constância da EstAcqua, onde o tempo necessário para carregar os *drivers* dos sensores e efetuar a leitura dos mesmos foi de 1,1 s em todas as medições.

8.3 Autonomia de Bateria

Para estimar a autonomia da bateria T_b através de cálculos teóricos, são necessárias a capacidade da bateria C (em mAh), a corrente elétrica consumida no modo ativo I_a (em mA), no modo inativo (*deep sleep*) I_d (em mA) e a fração de tempo D (em %) em que o

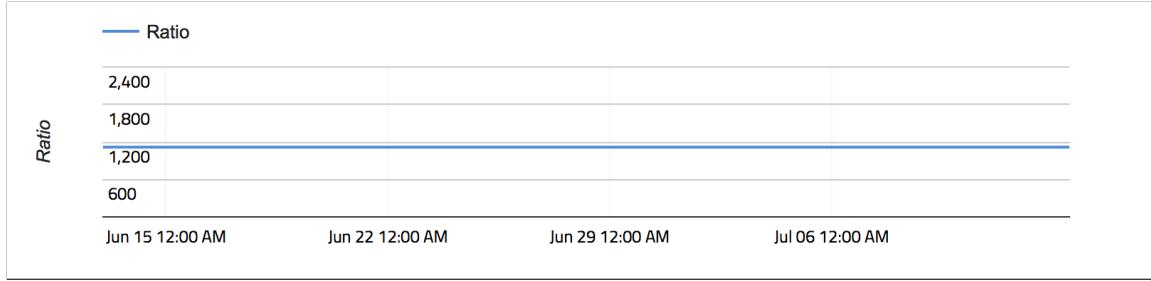


Figura 56 – Tempo de carregamento dos *drivers* dos sensores e leitura dos dados no período de 13/06/2018 a 13/07/2018

sistema está no modo ativo (denominada *Duty Cycle* ou ciclo de trabalho) :

$$T_b = \frac{100 C}{24[I_a D + I_d(100 - D)]} , \quad (8.1)$$

que mostra que se deve minimizar I_a e I_d para maximizar a autonomia da bateria T_b .

Foi utilizado o sensor de corrente contínua INA219 para medir a corrente consumida no modo ativo e multímetro no modo inativo. Na última versão da EstAcqua, quando foram realizadas as melhorias de consumo de bateria, a corrente de *deep sleep* ficou com magnitude de μA e o INA219 não consegue medir correntes nessa magnitude (98).

Para os novos cálculos de corrente foi utilizado um multímetro Minipa ET-2082D. A Tabela 8 mostra os valores de precisão e resolução para medidas de corrente contínua com o multímetro (99).

Tabela 8 – Resolução e precisão por faixa de corrente contínua para o Minipa ET-2082D

Faixa DC	Resolução	Precisão
200 μA	0,1 μA	$\pm(0,8\% + 10\text{D})$
2 mA	0,001 mA	$\pm(0,8\% + 10\text{D})$
20 mA	0,01 mA	$\pm(0,8\% + 10\text{D})$
200 mA	0,1 mA	$\pm(0,8\% + 10\text{D})$
20 A	0,01 A	$\pm(2,0\% + 5\text{D})$

Foram realizadas 10 medidas de corrente no modo ativo e *deep sleep*, onde foram encontrados os seguintes consumos médios, considerando as incertezas da Tabela 8, de $I_a = (50,0 \pm 0,50)$ mA e $I_d = (94,8 \pm 1,75)$ μA , respectivamente. Utilizando o tempo em que o nó EstAcqua permanece no estado ativo, 3,5 s, em relação ao período do ciclo de operação de 600 s, obtemos um *Duty Cycle* $D \simeq 0,583\%$.

Utilizando uma bateria com capacidade nominal de 2.600 mAh resultou em longa autonomia de bateria $T_b \simeq 280$ dias, vide a Figura 57, que mostra claramente o comportamento não-linear da Equação (8.1) de autonomia da bateria. Diante disso é possível estabelecer uma relação de compromisso entre o *Duty Cycle* e a autonomia da bateria. Por exemplo, se alterar a frequência de leitura dos sensores para 1 h ao invés de 10 min, o

Duty Cycle reduz para $D \simeq 0,0972\%$ e, conforme mostrado na Figura 58, a autonomia de bateria passa para 755 dias, aproximadamente.

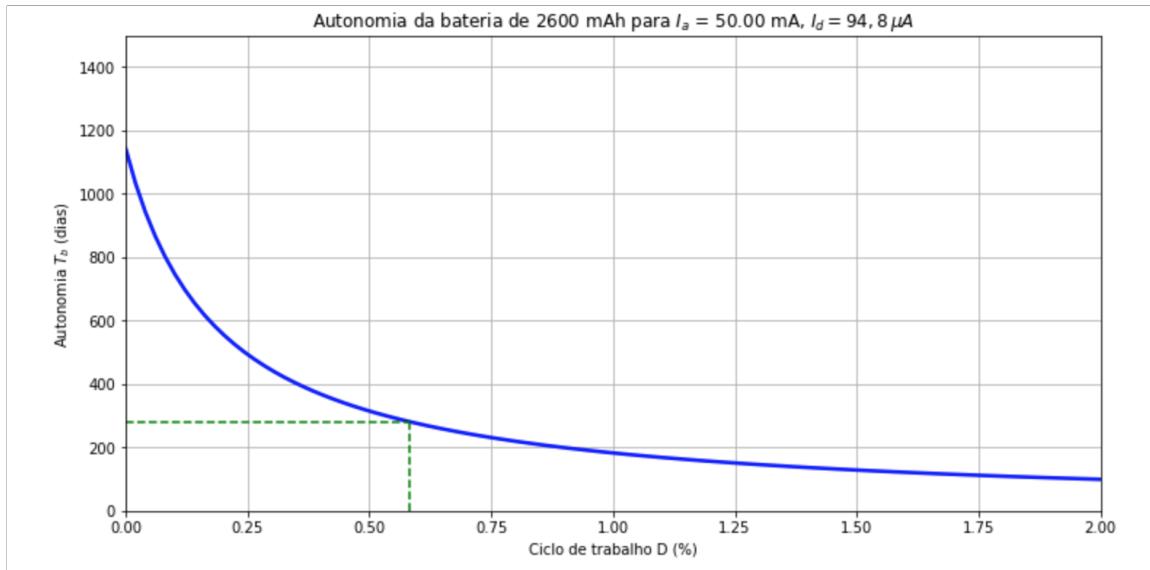


Figura 57 – Autonomia da bateria de 2.600 mAh versus ciclo de trabalho para correntes de modo ativo $I_a = 50,0$ mA, *deep sleep* $I_d = 0,0948$ mA e *Duty Cycle* $D \simeq 0,583\%$

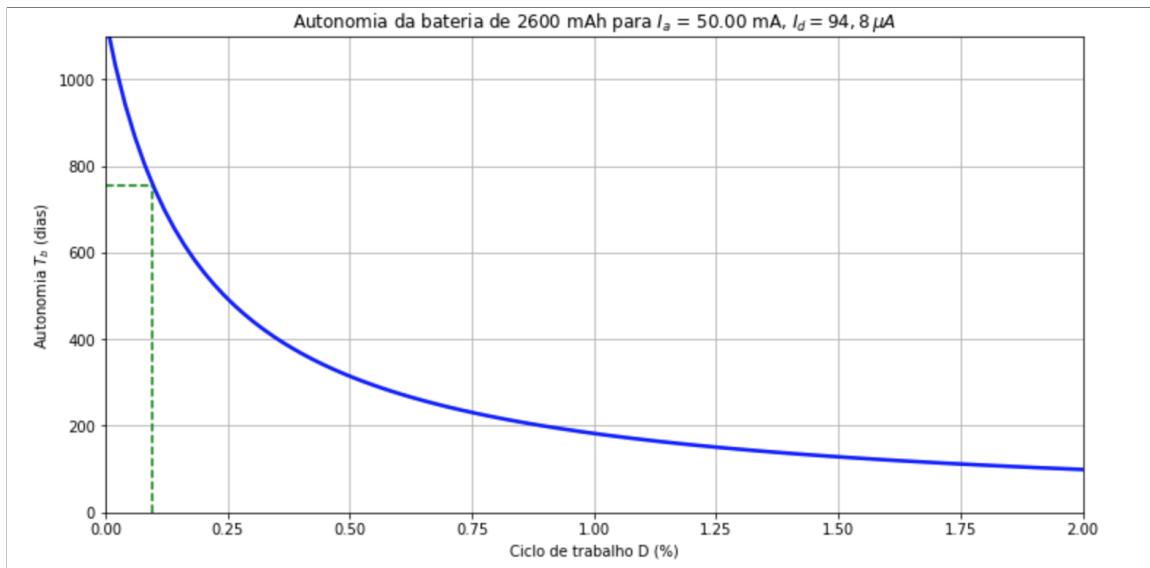


Figura 58 – Autonomia da bateria de 2.600 mAh versus ciclo de trabalho para correntes de modo ativo $I_a = 50,0$ mA, *deep sleep* $I_d = 0,0948$ mA e *Duty Cycle* $D \simeq 0,0972\%$

Conforme dito anteriormente, o código foi otimizado para tentar obter a maior autonomia de bateria. Esse requisito se deve ao fato de que, muitas vezes, os sensores estão em lugares de difícil acesso ou distantes. A autonomia dependerá da quantidade de sensores (e tipos) que estão conectados a EstAcqua. É importante ressaltar que a escolha da quantidade de sensores em um nó, se trata de uma escolha de compromisso com a

autonomia da bateria. Caso um nó esteja sobrecarregado de sensores, comprometendo a autonomia da bateria, é interessante dividir em dois nós.

Novamente voltando ao requisito custo, o uso de baterias de Li-ion possui um baixo custo e conseguem uma autonomia considerável, como mostrado nos gráficos acima. Caso seja necessário maior longevidade da bateria deve-se considerar a utilização de outros tipos de baterias (por exemplo as citadas anteriormente). Devido a auto descarga da bateria (também chamado de tempo de prateleira da bateria) não é recomendado deixar baterias adicionais para serem acionadas quando a bateria principal se esgotar (100). A auto descarga da bateria é extremamente sensível à temperatura, onde uma bateria de Li-ion à 20°C sofre uma auto descarga entre 5 e 10 % por mês de sua carga (100).

8.4 Avaliação dos Sensores usados na EstAcqua

No Quadro 2 temos um comparativo entre os sensores conectados à EstAcqua com alguns produtos comerciais da empresa *Onset*, sendo eles: (i) *TidbiT v2 Water Temperature Data Logger*¹; (ii) *HOBO U23 Pro v2 External Temperature/Relative Humidity Data Logger*²; e (iii) *HOBO Pendant Temperature/Light 64K Data Logger*³.

Quadro 2 – Comparativo com sensores comerciais

	Luminosidade		Temperatura externa e umidade relativa		Temperatura submersa	
	MAX44009 (EstAcqua)	Onset HOBO Pendant	BME280 (EstAcqua)	Onset HOBO U23 Pro	DS18B20 (EstAcqua)	TidBit v2
Acurácia	1 lux	Desenvolvido para medidas relativas	Temperatura: ± 1°C Umidade: ± 3%	Temperatura: ± 0,21°C Umidade: ± 2,5%	± 0,5 °C	± 0,21 °C
Resolução	0,045 lux	1 lux	Temperatura: 0,01 °C Umidade: 0,08%	Temperatura: 0,02 °C Umidade: 0,05%	0,04 a 0,17 °C	0,02 °C
Temperatura de operação	-40 a 85 °C	-20 a 70 °C	-40 a 85 °C	-40 a 70 °C	-55 a 125 °C	-20 a 70 °C
Fundo de escala	0 a 188.000 lux	0 a 320.000 lux	Temperatura: -40 a 85 °C Umidade: 0 a 100%	Temperatura: -40 a 70 °C Umidade: 0 a 100%	-55 a 125 °C	-20 a 70 °C
Preço	U\$ 6,17	U\$ 64,00	U\$ 7,56	U\$ 199,00	U\$ 1,37	U\$ 133,00
Bateria	Recarregável	Não recarregável, com duração de 1 ano	Recarregável	Não recarregável, com duração de 3 anos	Recarregável	Não recarregável, com duração de 5 anos

Como mostrado no Quadro 2, podemos verificar que os sensores⁴ utilizados na EstAcqua custam aproximadamente 1% do valor dos produtos comerciais, por isso o prejuízo em caso de furto ou dano dos sensores é extremamente baixo. Além disso, permite a aquisição de novos sensores, não ficando restrita a um fabricante.

Com uma frequência de coleta de dados de 10 min pela EstAcqua, são transmitidos 144 pacotes. Salvando os dados desses pacotes no cartão micro SD ocupa-se um espaço de aproximadamente 32 KB por dia. Dessa forma, para esgotar o espaço de 32 GB do cartão

¹ <http://www.onsetcomp.com/products/data-loggers/utbi-001>

² <http://www.onsetcomp.com/products/data-loggers/u23-002>

³ <http://www.onsetcomp.com/products/data-loggers/ua-002-64>

⁴ Dados obtidos através do *datasheet* de cada sensor

micro SD, seriam necessários mais de 2800 anos. Obviamente esse tempo depende da capacidade do cartão micro SD e da frequência de leitura dos sensores. Mantendo a coleta de 10 em 10 minutos, porém com um cartão micro SD de 128 MB, o tempo necessário para ocupar todo seu espaço é de aproximadamente 11 anos. Dessa forma não há necessidade de se preocupar com falta de espaço no cartão.

Os sensores utilizados na EstAcqua apresentam uma acurácia e resolução próximas dos sensores comerciais e em alguns casos, melhor. Analisando o fundo de escala e temperatura de operação vemos que os sensores da EstAcqua são, na maioria, melhores que os comerciais. Apesar da acurácia e resolução serem em alguns casos, inferiores aos produtos comerciais analisados, não torna a substituição pela EstAcqua inviável, pois para medições ambientais os valores analisados com precisão de uma casa decimal são suficientes, conforme Figura 52, que mostra o gráfico de temperatura gerado pelo equipamento comercial *Sontek CastAway-CTD*⁵, cujo incremento de temperatura é de 1 °C.

Apesar de possuir autonomia de bateria de aproximadamente 280 dias, é possível acoplar painel solar portátil em cada nó da EstAcqua, resultando em autonomia limitada somente pela vida útil da bateria e dos outros componentes eletrônicos.

Por fim, a EstAcqua ainda registra os dados coletados em cartão micro SD e no *IoT Cloud*, tornando possível a visualização dos dados em tempo real e, em caso de perda ou furto do equipamento, os dados coletados não são perdidos pois já se encontram na *IoT Cloud*. Vale ressaltar que nas soluções comerciais apresentadas, caso não haja uma coleta regular, os dados começam a ser sobreescritos. No caso da EstAcqua, mesmo utilizando um cartão micro SD de 128 MB, levaria 11 anos para ocupar todo o espaço. Diante disso, caso essa situação ocorra, a EstAcqua para de salvar os dados no cartão micro SD e somente o transmite para a nuvem *IoT*. A EstAcqua salva os dados separados por dia, dessa forma, caso haja a necessidade de sobreescriver, pode-se implementar via código uma verificação para apagar os arquivos de *log* mais antigos. Tal função não foi implementada devido ao longo tempo necessário para ocupá-lo completamente.

8.5 Custo

Conforme demonstrado anteriormente, para que a EstAcqua funcione, é necessário pelo menos um *NanoGateway* e um nó. Assim, o valor do investimento⁶ para aquisição dos componentes da EstAcqua estão dividos da seguinte forma.

O *NanoGateway* é cosntituido de um LoPy junto com uma placa de expansão e duas antenas: uma para a transmissão LoRa e outra para se comunicar via Wi-Fi. O conjunto é então colocado em uma caixa específica para o LoPy. A Figura 59 mostra tais

⁵ <https://www.sontek.com/productsdetail.php?CastAway-CTD-11>

⁶ Preços consultados na data de 27/08/2018

componentes e o valor total de U\$ 86,00 para aquisição dos mesmos.



Figura 59 – Preços dos componentes do *NanoGateway*

Para o nó foi confeccionado uma placa de circuito impresso (desenvolvida pelo autor) para eliminar o uso da placa de expansão (devido ao alto consumo de bateria). É necessário a aquisição de uma antena LoRa e uma bateria de Li-ion para energizar o LoPy. No caso da EstAcqua foram utilizados 4 sensores DS18B20, um MAX44009 e um BME280, conforme demonstrado na Figura 60. Todo o sistema é colocado em uma caixa certificada IP56, custando no total U\$ 86,20.



Figura 60 – Preços dos componentes do Nô

Dessa forma, o total gasto para se ter uma versão funcional da EstAcqua é a soma do valor de um *NanoGateway*, do valor de um nó e demais componentes utilizados, tais como:

cabos, conectores, alicates, placa de fenolite, solda etc (estimado em aproximadamente U\$ 20,00). Resultando em um total de U\$ 192,20.

Vale ressaltar que o mais caro é o LoPy e que uma vez adquirido pode-se trocar os sensores que serão conectados a ele de forma rápida e barata (como foi demonstrado, o sensor mais caro custou U\$ 7,00). Como comparação com um equipamento comercial, podemos utilizar o sensor de temperatura TidBit apresentado na seção anterior. O TidBit trata-se de um sensor de temperatura subaquático que faz um log dos valores lidos, sendo necessário que a cada um mês esses valores sejam coletados do TidBit caso contrário haverá uma sobrescrita dos dados. Entretanto, para que se possa utilizar o TidBit, é necessário a aquisição do mesmo juntamente com um acoplador (vendido somente pelo fabricante), uma Base também vendida somente pelo fabricante, um cabo também proprietário que irá se conectar a base e ao USB do computador e por fim, um software para que possa ser realizada essa coleta dos dados do TidBit, conforme mostrado na Figura 61.

Dessa forma, para que se tenha apenas um TidBit funcionando, é necessário adquirir U\$356,00 em equipamentos(101), conforme demonstrado na Figura 61.



Figura 61 – Equipamento necessário para efetuar a leitura do TidBit

Dessa forma, para que se tenha apenas um TidBit funcionando, é necessário adquirir U\$356,00 em equipamentos(101), conforme demonstrado na Figura 62.

Total price: \$356.00

Add all three to Cart

Add all three to List

i These items are shipped from and sold by different sellers. [Show details](#)

- This item:** Onset BASE-U-4, Optic USB Base Station **\$124.00**
- Onset HOBO UTBI-001 Tidbit Waterproof Temperature Data Logger** **\$133.00**
- HOBOWare Pro Data Logger CD** **\$99.00**

Figura 62 – Valor total dos equipamentos para efetuar a leitura do TidBit

A princípio pode parecer que a diferença é pequena, mas ela se torna significante a medida que vão se adquirindo novos sensores comerciais. Na solução da EstAcqua apresentada, é possível medir a temperatura de uma coluna d'água em 4 pontos distintos. Para realizar a mesma função com TidBit, é necessário adquirir esse primeiro pacote de U\$356,00 mais três TidBits de U\$133,00, totalizando U\$755,00. Já a EstAcqua, com U\$192,20 é capaz de realizar a mesma ação sem ter o inconveniente de voltar ao local de instalação a cada mês e ainda possibilitando o acesso remoto aos dados a medida que os mesmos são coletados.

9 Conclusão

A criação da Internet mudou completamente o mundo e a forma com que nos comunicamos. Atualmente estamos vivendo uma expansão desse horizonte com a criação dos dispositivos inteligentes que se comunicam através da Internet, chamados de Internet das Coisas, possibilitando que diversas aplicações sejam desenvolvidas.

Com a facilidade de obtenção desses dispositivos inteligentes e a proliferação de aplicações envolvendo Internet das Coisas, foi desenvolvida a EstAcqua: uma estação com sensores de superfície e submersos para monitoramento ambiental e oceanográfico de lagos e lagoas, com alta autonomia de bateria e longo alcance de transmissão. Além disso, os dados coletados pela EstAcqua são transmitidos para um servidor de Internet das Coisas na nuvem, onde são repassados para uma aplicação, possibilitando que o usuário tenha acesso aos dados.

Normalmente, lagos e lagoas se encontram em lugares de difícil acesso e afastados de locais com infraestrutura de rede, por isso foi necessário escolher uma tecnologia com longo alcance de transmissão com baixo consumo de energia. Dessa forma, foi escolhido LoRa como tecnologia de transmissão, onde a EstAcqua consegue transmitir os dados coletados para um *gateway* localizado a poucos quilômetros de distância. Foram realizados testes com a EstAcqua para testar seu funcionamento e também o comportamento da nuvem Internet das Coisas e a aplicação escolhida.

Os testes realizados mostrou que a tecnologia LoRa atende perfeitamente os propósitos da EstAcqua, conseguindo transmitir dados para um *gateway* situado a até 3 km de distância. O servidor Internet das Coisas na nuvem possibilitou a criação de gráficos para visualização dos dados coletados, disponibilizando tanto via *web* quanto *mobile*, além de permitir a criação de alarmes para alertar a ocorrência de situações anômalas. Em ambientes sem conexão com a Internet, os dados não são perdidos, pois são salvos em um cartão micro SD instalado no *gateway*. Após todas as melhorias realizadas na EstAcqua, chegando em sua versão final, a autonomia de bateria ficou em 280 dias para amostragens a cada 10 min, e em aproximadamente 755 dias caso a amostragem seja diária. Os sensores utilizados pela EstAcqua possuem um custo extremamente menor, com acurácia e resolução próximas dos sensores comerciais analisados. A EstAcqua ainda conta com uma redundância de armazenamento para os dados, pois mesmo se o sensor falhar, for danificado e/ou furtado, os dados coletados já foram transferidos para a nuvem, diferente das soluções atuais, onde se perderia todos os dados coletados.

O custo de montagem de uma estação igual a EstAcqua mostrou-se menor em relação a aquisição de equipamentos individuais para realização dos mesmos procedimentos.

Essa diferença se torna ainda mais atrativa quando há uma escalabilidade nos sensores utilizados, onde as aplicações comerciais se tornam uma alternativa exageradamente cara, comparada com a EstAcqua.

Os resultados obtidos na avaliação da EstAcqua mostram que ela se trata de uma solução viável não somente para o uso acadêmico, mas também como um substituto de menor custo, confiável, com maior integração e mais funcionalidades do que a maioria das soluções encontradas no mercado atualmente.

São necessários novos estudos para analisar o comportamento da EstAcqua com a utilização de placas solares para carregar a bateria utilizada na EstAcqua, a adição de novos sensores e o uso de atuadores com mensagens de *downlink* para controle dos mesmos, como por exemplo: ligar um aerador caso a concentração de oxigênio dissolvido no lago esteja abaixo do normal.

Esses resultados sugerem que a utilização de microcontroladores de baixo custo, com sensores de prateleira e uso de LoRa para transmissão dos dados é uma solução viável e possível de se realizar. Dessa forma, diminui-se o gasto com tecnologias e permite que o usuário escolha quais sensores deseja utilizar, não ficando engessado aos padrões de um fabricante.

9.1 Trabalhos Futuros

Seria desejável explorar a validade dessas conclusões em ambientes mais extremos, como em florestas, onde o nível de interferência é maior. Além disso, o estudo para a inclusão de novos sensores para monitoramento de gases na atmosfera e oxigênio dissolvido também deve ser realizado. E também estudar sobre LoRa Mesh, anunciado pela Pycom em julho de 2018, que permite aumentar o alcance com nós retransmitindo pacotes de outros nós, minimizando a quantidade de *gateways* LoRaWAN em uma área. Com o intuito de deixar a EstAcqua mais completa (em detrimento da autonomia da bateria), considerar a adição de mais sensores subaquático oceanográficos tais como: temperatura, pressão, salinidade, pH, oxigênio dissolvido, clorofila A (faixa visível), turbidez e luz visível. Outra função interessante é a criação de um método de retransmitir os pacotes para a TTN, ou seja, caso algum pacote recebido pelo *gateway* não seja transmitido para a nuvem IoT, por exemplo, devido a uma falha momentânea no acesso à Internet, o mesmo entraria em uma fila e seria retransmitido posteriormente. Adicionalmente considerar a utilização de bateria descartável do tipo $LiSOCl_2$, com grande autonomia, a ser calculada exatamente, girando em torno de 5 a 10 anos.

APÊNDICE A – *Low Power Packet*

Um grande diferencial para aplicações que envolvem LoRa é a existência do *Cayenne Low Power Packet* (LPP), que fornece uma maneira fácil e conveniente de enviar dados através de redes LoRaWAN. Como em redes LoRaWAN há restrições de *Duty Cycle* e tamanho máximo do pacote, podendo chegar a 11 *Bytes* dependendo do DR utilizado, o Cayenne LPP é uma forma eficaz para economizar bytes, permitindo que o dispositivo envie vários dados do sensor de uma só vez.

Como é possível enviar dados de diferentes sensores no mesmo pacote, cada sensor é pré-fixado com 2 *Bytes*, sendo 1 *Byte* para o canal, que identifica exclusivamente cada sensor no dispositivo, e 1 *Byte* para o tipo de dado sendo transmitido. Os tipos de dados estão em conformidade com as diretrizes *Internet Protocol for Smart Objects* (IPSO), que identifica cada tipo de dado.

A IPSO Alliance([102](#)) é um fórum global composto por diversos membros internacionais focados em permitir que dispositivos IoT se comuniquem, compreendam e confiem uns aos outros com interoperabilidade global baseada em padrões abertos. O Quadro 3 mostra a codificação utilizada pelo LPP([75](#)):

Quadro 3 – Codificação utilizada pelo LPP

Tipo de dado	IPSO	LPP	Hex	Bytes	Resolução
Entrada Digital	3200	0	0	1	1
Saída Digital	3201	1	1	1	1
Entrada Analógica	3202	2	2	2	0,01
Saída Analógica	3203	3	3	2	0,01
Iluminância	3301	101	65	2	1
Presença	3302	102	66	1	1
Temperatura	3303	103	67	2	0,1
Umidade	3304	104	68	1	0,5
Acelerômetro	3313	113	71	6	0,001
Barômetro	3315	115	73	2	0,1
Giroscópio	3334	134	86	6	0,01
GPS	3336	136	88	9	0,0001

A *The Things Network* possui suporte a pacotes LPP e existem bibliotecas para *Python* e C que facilitam a transmissão dos dados usando LPP. Dessa maneira, o programador não precisa se preocupar em como montar o pacote, podendo criá-lo através de funções.

A Figura 63 mostra um exemplo do formato do pacote LPP após serem inseridos os

dados dos sensores. Cada dado sempre começa com 1 *Byte* representando o canal, seguido de 1 *Byte* representando o tipo do dado e depois N *Bytes* dos dados. O valor de N depende do tipo de dado sendo transmitido conforme mostrado no Quadro 3.

1 Byte	1 Byte	N Byte	1 Byte	1 Byte	M Byte	...
Canal Dado 1	Tipo Dado 1	Dado 1	Canal Dado 2	Tipo Dado 2	Dado 2	...

Figura 63 – Formato do pacote LPP

Supondo que se deseja transmitir os dados de dois sensores de temperatura e um de iluminância. As temperaturas lidas pelos sensores foram 24,5 °C e 2,1 °C, e, pelo sensor de iluminância, 23.450 lux. O pacote LPP formado será:

03 67 00 F5 05 67 00 15 01 65 5B 9A (A.1)

Esse pacote ao chegar na TTN será decodificado da seguinte maneira:

- O primeiro *Byte* (03) indica que o canal a ser utilizado é 3. O segundo *Byte* (67) indica que o tipo de dado é temperatura, conforme Quadro 3. Como o dado de temperatura possui 2 *Bytes*, os próximos 2 *Bytes* (00 F5) representam o valor lido. O valor 00F5 em hexadecimal representa 245 em decimal. Novamente recorrendo ao Quadro 3 vemos que a resolução para temperatura é de 0,1, logo o valor recebido foi de 24,5 °C.
- Ao ler o quinto *Byte* (05) sabe-se que o canal utilizado é o 5. O sexto *Byte* (67) indica que o tipo de dado também é temperatura. De forma análoga acima, o sétimo e oitavo *Bytes* (00 15) são usados para saber a temperatura. Convertendo 0015 para decimal tem-se 21, como a resolução é 0,1, a temperatura recebida foi 2,1 °C.
- O nono *Byte* (01) indica que o canal utilizado é o 1, o décimo *Byte* (65) indica que se trata de um dado de iluminância. Pelo Quadro 3, iluminância requer dois *Bytes*, logo os últimos dois *Bytes* (5B 9A) representam o valor lido. Convertendo para decimal resulta em 23.450. Como a resolução é 1, o valor recebido é de 23.450 lux.

Neste exemplo, com apenas 12 *Bytes* foram transmitidos dados de 3 sensores diferentes, de dois tipos distintos e já prontos para serem recebidos na aplicação e disponibilizados para uso.

APÊNDICE B – *Nanogateway*

```

1 #####
2 # Modificado por Alan Helal
3 # alan@helal.com.br
4 # Universidade Federal do Espírito Santo
5 # https://github.com/HelalBR/EstAcqua
6 #####
7
8 import errno
9 import machine
10 import ubinascii
11 import ujson
12 import uos
13 import usocket
14 import utime
15 import _thread
16 from micropython import const
17 from network import LoRa
18 from network import WLAN
19 from machine import Timer
20 import pycom
21 from machine import SD
22 import os
23
24 # Tenta montar o cartão SD
25 try:
26     sd = SD()
27     os.mount(sd, '/sd')
28     print("microSD card mounted with free {:.3f} GB".format(os.getfree('/sd')/(1024*1024)))
29     flaguSDcard = True
30 except OSError as e:
31     pycom.rgbled(0xFF0000)
32     print("Caught exception while mounting microSD. {}".format(e))
33     flaguSDcard = False
34
35 try:
36     f = open('/sd/recv.txt', "r")
37     exists = True
38     f.close()
39 except OSError as e:
40     exists = False
41
42

```

```

43 PROTOCOL_VERSION = const(2)
44
45 PUSH_DATA = const(0)
46 PUSH_ACK = const(1)
47 PULL_DATA = const(2)
48 PULL_ACK = const(4)
49 PULL_RESP = const(3)
50
51 TX_ERR_NONE = 'NONE'
52 TX_ERR_TOO_LATE = 'TOO_LATE'
53 TX_ERR_TOO_EARLY = 'TOO_EARLY'
54 TX_ERR_COLLISION_PACKET = 'COLLISION_PACKET'
55 TX_ERR_COLLISION_BEACON = 'COLLISION_BEACON'
56 TX_ERR_TX_FREQ = 'TX_FREQ'
57 TX_ERR_TX_POWER = 'TX_POWER'
58 TX_ERR_GPS_UNLOCKED = 'GPS_UNLOCKED'
59
60 UDP_THREAD_CYCLE_MS = const(10)
61
62 STAT_PK = {
63     'stat': {
64         'time': '',
65         'lati': 0,
66         'long': 0,
67         'alti': 0,
68         'rxnb': 0,
69         'rxok': 0,
70         'rxfw': 0,
71         'ackr': 100.0,
72         'dwnb': 0,
73         'txnb': 0
74     }
75 }
76
77 RX_PK = {
78     'rxpk': [
79         'time': '',
80         'tmst': 0,
81         'chan': 0,
82         'rfch': 0,
83         'freq': 0,
84         'stat': 1,
85         'modu': 'LORA',
86         'datr': '',
87         'codr': '4/5',
88         'rssr': 0,
89         'lsnr': 0,

```

```
90         'size': 0,
91         'data': ''
92     }]
93 }
94
95 TX_ACK_PK = {
96     'txpk_ack': {
97         'error': ''
98     }
99 }
100
101
102 class NanoGateway:
103     """
104     Nano gateway class , set up by default for use with TTN, but can be
105     configured
106     for any other network supporting the Semtech Packet Forwarder .
107     Only required configuration is wifi_ssid and wifi_password which are
108     used for
109     connecting to the Internet .
110     """
111
112     def __init__(self, id, frequency, datarate, ssid, password, server,
113                  port, ntp_server, ntp_period):
114         self.id = id
115         self.server = server
116         self.port = port
117
118         self.frequency = frequency
119         self.datarate = datarate
120
121         self.ssid = ssid
122         self.password = password
123
124         self.server_ip = None
125
126         self.rxbn = 0
127         self.rxok = 0
128         self.rxfw = 0
129         self.dwnb = 0
130         self.txnb = 0
131
132         self.sf = self._dr_to_sf(self.datarate)
133         self.bw = self._dr_to_bw(self.datarate)
```

```
134
135     self.stat_alarm = None
136     self.pull_alarm = None
137     self.uplink_alarm = None
138
139     self.wlan = None
140     self.sock = None
141     self.udp_stop = False
142     self.udp_lock = _thread.allocate_lock()
143
144     self.lora = None
145     self.lora_sock = None
146
147     self rtc = machine.RTC()
148
149 def start(self):
150     """
151     Starts the LoRaWAN nano gateway.
152     """
153
154     self._log('Starting LoRaWAN nano gateway with id: {}', self.id)
155
156     # setup WiFi as a station and connect
157     self.wlan = WLAN(mode=WLAN.STA)
158     self._connect_to_wifi()
159
160     # get a time sync
161     self._log('Syncing time with {} ...', self.ntp_server)
162     self.rtc.ntp_sync(self.ntp_server, update_period=self.ntp_period)
163     while not self.rtc.synced():
164         utime.sleep_ms(50)
165     self._log("RTC NTP sync complete")
166
167     # get the server IP and create an UDP socket
168     self.server_ip = usocket.getaddrinfo(self.server, self.port)[0][-1]
169     self._log('Opening UDP socket to {} ({}), port {}...', self.server,
170             self.server_ip[0], self.server_ip[1])
171     self.sock = usocket.socket(usocket.AF_INET, usocket.SOCK_DGRAM,
172                               usocket.IPPROTO_UDP)
173     self.sock.setsockopt(usocket.SOL_SOCKET, usocket.SO_REUSEADDR, 1)
174     self.sock.setblocking(False)
175
176     # push the first time immediately
177     self._push_data(self._make_stat_packet())
178
179     # create the alarms
180     self.stat_alarm = Timer.Alarm(handler=lambda t: self._push_data(
```

```

    self._make_stat_packet() , s=60, periodic=True)
179      self.pull_alarm = Timer.Alarm(handler=lambda u: self._pull_data() ,
180      s=25, periodic=True)

181      # start the UDP receive thread
182      self.udp_stop = False
183      _thread.start_new_thread(self._udp_thread, ())
184

185      # initialize the LoRa radio in LORA mode
186      self._log('Setting up the LoRa radio at {} Mhz using {}'.format(
187          self.freq_to_float(self.frequency), self.datarate))
188      self.lora = LoRa(
189          mode=LoRa.LORA,
190          frequency=self.frequency,
191          bandwidth=self.bw,
192          sf=self.sf,
193          preamble=8,
194          coding_rate=LoRa.CODING_4_5,
195          tx_iq=True
196      )

197      # create a raw LoRa socket
198      self.lora_sock = usocket.socket(usocket.AF_LORA, usocket.SOCK_RAW)
199      self.lora_sock.setblocking(False)
200      self.lora_tx_done = False

201      self.lora.callback(trigger=(LoRa.RX_PACKET_EVENT | LoRa.
202 TX_PACKET_EVENT), handler=self._lora_cb)
203      self._log('LoRaWAN nano gateway online')

204  def stop(self):
205      """
206      Stops the LoRaWAN nano gateway.
207      """
208

209      self._log('Stopping ...')

210

211      # send the LoRa radio to sleep
212      self.lora.callback(trigger=None, handler=None)
213      self.lora.power_mode(LoRa.SLEEP)

214

215      # stop the NTP sync
216      self.rtc.ntp_sync(None)

217

218      # cancel all the alarms
219      self.stat_alarm.cancel()
220      self.pull_alarm.cancel()

```

```
222
223     # signal the UDP thread to stop
224     self.udp_stop = True
225     while self.udp_stop:
226         utime.sleep_ms(50)
227
228     # disable WLAN
229     self.wlan.disconnect()
230     self.wlan.deinit()
231
232     def _connect_to_wifi(self):
233         self.wlan.init(antenna=WLAN.EXT_ANT)
234         self.wlan.connect(self.ssid, auth=(None, self.password))
235         while not self.wlan.isconnected():
236             utime.sleep_ms(50)
237             self._log('WiFi connected to: {}', self.ssid)
238             pycom.rgbled(0x00ff00)
239
240     def _dr_to_sf(self, dr):
241         sf = dr[2:4]
242         if sf[1] not in '0123456789':
243             sf = sf[:1]
244         return int(sf)
245
246     def _dr_to_bw(self, dr):
247         bw = dr[-5:]
248         if bw == 'BW125':
249             return LoRa.BW_125KHZ
250         elif bw == 'BW250':
251             return LoRa.BW_250KHZ
252         else:
253             return LoRa.BW_500KHZ
254
255     def _sf_bw_to_dr(self, sf, bw):
256         dr = 'SF' + str(sf)
257         if bw == LoRa.BW_125KHZ:
258             return dr + 'BW125'
259         elif bw == LoRa.BW_250KHZ:
260             return dr + 'BW250'
261         else:
262             return dr + 'BW500'
263
264     def _lora_cb(self, lora):
265         """
266             LoRa radio events callback handler.
267         """
268
```

```

269     events = lora.events()
270     if events & LoRa.RX_PACKET_EVENT:
271         self.rxbn += 1
272         self.rxok += 1
273         rx_data = self.lora_sock.recv(256)
274         stats = lora.stats()
275         packet = self._make_node_packet(rx_data, self.rtc.now(), stats.
276             rx_timestamp, stats.sfrx, self.bw, stats.rssi, stats.snr)
277         # pacote recebido! pisca o led em azul para mostrar que o
278         # NanoGateway recebeu o dado
279         # mesmo que nao esteja conectado na TTN ele ira salvar o dado
280         # no cartao de memoria
281         # e esse dado depois pode ser recuperado depois
282         pycom.rgbled(0x0000ff)
283         utime.sleep_ms(100)
284         pycom.rgbled(0x00ff00)
285         # se o cartao estiver montado ele grava o pacote recebido
286         if flaguSDcard:
287             logfile = open('/sd/recv.txt', 'a')
288             logfile.write('{}\n'.format(packet))
289             logfile.close()
290             os.sync()
291             self._push_data(packet)
292             self._log('Received packet: {}', packet)
293             self.rfw += 1
294         if events & LoRa.TX_PACKET_EVENT:
295             self.txnb += 1
296             lora.init(
297                 mode=LoRa.LORA,
298                 frequency=self.frequency,
299                 bandwidth=self.bw,
300                 sf=self.sf,
301                 preamble=8,
302                 coding_rate=LoRa.CODING_4_5,
303                 tx_iq=True
304             )
305
306     def _freq_to_float(self, frequency):
307         """
308             MicroPython has some inprecision when doing large float division.
309             To counter this, this method first does integer division until we
310             reach the decimal breaking point. This doesn't completely eliminate
311             the issue in all cases, but it does help for a number of commonly
312             used frequencies.
313         """
314
315         divider = 6

```

```

313     while divider > 0 and frequency % 10 == 0:
314         frequency = frequency // 10
315         divider -= 1
316     if divider > 0:
317         frequency = frequency / (10 ** divider)
318     return frequency
319
320 def _make_stat_packet(self):
321     now = self.rtc.now()
322     STAT_PK["stat"]["time"] = "%d-%02d-%02d %02d:%02d:%02d GMT" % (now
323     [0], now[1], now[2], now[3], now[4], now[5])
324     STAT_PK["stat"]["rxnb"] = self.rxnb
325     STAT_PK["stat"]["rxok"] = self.rxok
326     STAT_PK["stat"]["rxfw"] = self.rxfw
327     STAT_PK["stat"]["dwnb"] = self.dwnb
328     STAT_PK["stat"]["txnb"] = self.txnb
329     return ujson.dumps(STAT_PK)
330
331 def _make_node_packet(self, rx_data, rx_time, tmst, sf, bw, rssi, snr):
332     RX_PK["rxpk"][0]["time"] = "%d-%02d-%02dT%02d:%02d.%dZ" % (
333     rx_time[0], rx_time[1], rx_time[2], rx_time[3], rx_time[4], rx_time[5],
334     rx_time[6])
335     RX_PK["rxpk"][0]["tmst"] = tmst
336     RX_PK["rxpk"][0]["freq"] = self._freq_to_float(self.frequency)
337     RX_PK["rxpk"][0]["datr"] = self._sf_bw_to_dr(sf, bw)
338     RX_PK["rxpk"][0]["rssi"] = rssi
339     RX_PK["rxpk"][0]["lsnr"] = snr
340     RX_PK["rxpk"][0]["data"] = ubinascii.b2a_base64(rx_data)[:-1]
341     RX_PK["rxpk"][0]["size"] = len(rx_data)
342     return ujson.dumps(RX_PK)
343
344 def _push_data(self, data):
345     token = uos.urandom(2)
346     packet = bytes([PROTOCOL_VERSION]) + token + bytes([PUSH_DATA]) +
347     ubinascii.unhexlify(self.id) + data
348     with self.udp_lock:
349         try:
350             self.sock.sendto(packet, self.server_ip)
351         except Exception as ex:
352             self._log('Failed to push uplink packet to server: {}', ex)
353
354 def _pull_data(self):
355     token = uos.urandom(2)
356     packet = bytes([PROTOCOL_VERSION]) + token + bytes([PULL_DATA]) +
357     ubinascii.unhexlify(self.id)
358     with self.udp_lock:
359         try:

```

```

355         self.sock.sendto(packet, self.server_ip)
356     except Exception as ex:
357         self._log('Failed to pull downlink packets from server: {}', ex)
358
359     def _ack_pull_rsp(self, token, error):
360         TX_ACK_PK["txpk_ack"]["error"] = error
361         resp = ujson.dumps(TX_ACK_PK)
362         packet = bytes([PROTOCOL_VERSION]) + token + bytes([PULL_ACK]) +
363         ubinascii.unhexlify(self.id) + resp
364         with self.udp_lock:
365             try:
366                 self.sock.sendto(packet, self.server_ip)
367             except Exception as ex:
368                 self._log('PULL RSP ACK exception: {}', ex)
369
370     def _send_down_link(self, data, tmst, datarate, frequency):
371         """
372             Transmits a downlink message over LoRa.
373         """
374         self.lora.init(
375             mode=LoRa.LORA,
376             frequency=frequency,
377             bandwidth=self._dr_to_bw(datarate),
378             sf=self._dr_to_sf(datarate),
379             preamble=8,
380             coding_rate=LoRa.CODING_4_5,
381             tx_iq=True
382         )
383         while utime.ticks_cpu() < tmst:
384             pass
385         self.lora_sock.send(data)
386         self._log(
387             'Sent downlink packet scheduled on {:.3f}, at {:.3f} Mhz using
388             {}: {}'.format(tmst / 1000000,
389             self._freq_to_float(frequency),
390             datarate,
391             data
392         )
393
394     def _udp_thread(self):
395         """
396             UDP thread, reads data from the server and handles it.
397         """

```

```

399     while not self.udp_stop:
400         try:
401             data, src = self.sock.recvfrom(1024)
402             _token = data[1:3]
403             _type = data[3]
404             if _type == PUSH_ACK:
405                 self._log("Push ack")
406             elif _type == PULL_ACK:
407                 self._log("Pull ack")
408             elif _type == PULL_RESP:
409                 self.dwnb += 1
410                 ack_error = TX_ERR_NONE
411                 tx_pk = ujson.loads(data[4:])
412                 tmst = tx_pk["txpk"]["tmst"]
413                 t_us = tmst - utime.ticks_cpu() - 15000
414                 if t_us < 0:
415                     t_us += 0xFFFFFFFF
416                 if t_us < 20000000:
417                     self.uplink_alarm = Timer.Alarm(
418                         handler=lambda x: self._send_down_link(
419                             ubinascii.a2b_base64(tx_pk["txpk"]["data"]))
420                         ,
421                         tx_pk["txpk"]["tmst"] - 50, tx_pk["txpk"][""
422                         datr"]),
423                         int(tx_pk["txpk"]["freq"] * 1000) * 1000
424                         ),
425                         us=t_us
426                     )
427                 else:
428                     ack_error = TX_ERR_TOO_LATE
429                     self._log('Downlink timestamp error!', t_us: {}, t_
430                     us)
431                     self._ack_pull_rsp(_token, ack_error)
432                     self._log("Pull rsp")
433             except usocket.timeout:
434                 pass
435             except OSError as ex:
436                 if ex errno != errno.EAGAIN:
437                     self._log('UDP recv OSError Exception: {}, ex')
438             except Exception as ex:
439                     self._log('UDP recv Exception: {}, ex')
440
441             # wait before trying to receive again
442             utime.sleep_ms(UDP_THREAD_CYCLE_MS)
443
444             # we are to close the socket
445             self.sock.close()

```

```
443     self.udp_stop = False
444     self._log( 'UDP thread stopped' )
445
446     def _log( self , message , *args ):
447         """
448             Outputs a log message to stdout .
449         """
450
451         print( '[{:>10.3f}] {}' .format(
452             utime.ticks_ms() / 1000 ,
453             str( message ) .format(*args)
454         ))
```

APÊNDICE C – Nó

```

1 #####
2 # Autor: Alan Helal
3 # alan@helal.com.br
4 # Universidade Federal do Espírito Santo
5 # https://github.com/HelalBR/EstAcqua
6 #####
7
8
9 flagDebug = False
10 flagRun = True
11 if flagDebug or flagRun:
12     from machine import Timer
13     chrono = Timer.Chrono()
14     chrono.start()
15
16 # Builtin modules
17 from network import LoRa
18 import socket
19 import binascii
20 import struct
21 import time
22 from machine import Pin, I2C, ADC, deepsleep
23 import machine
24 from onewire import DS18X20, OneWire
25 import pycom
26 import os
27
28 pycom.heartbeat(False)
29
30 # Additional modules
31 import bme280
32 import max44009
33 import cayenneLPP
34 import config
35 import myfuncs
36
37
38 if flagDebug:
39     lap1 = chrono.read_ms()
40     print("Tempo de importação dos módulos: {} ms".format(lap1))
41
42
43 #

```

```
44 # Inicializacao de LoRaWan com ABP
45 #
46
47 lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.US915)
48
49 # create an ABP authentication params
50 dev_addr = struct.unpack(">1", binascii.unhexlify('26XXXX'))[0]
51 nwk_swkey = binascii.unhexlify('50BDE0FD219E1XXXXXXXXXXXXF92812')
52 app_swkey = binascii.unhexlify('F9E434860F560XXXXXXXXXXXXB830ED')
53
54 # remove all the channels
55 for channel in range(0, 72):
56     lora.remove_channel(channel)
57
58 for channel in range(0, 72):
59     lora.add_channel(channel, frequency=config.LORA_FREQUENCY, dr_min=0,
60                       dr_max=3)
61
62 # join a network using ABP (Activation By Personalization)
63 lora.join(activation=LoRa.ABP, auth=(dev_addr, nwk_swkey, app_swkey))
64
65 # create a LoRa socket
66 s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
67
68 # set the LoRaWAN data rate
69 s.setsockopt(socket.SOL_LORA, socket.SO_DR, config.LORA_NODE_DR)
70
71 # make the socket blocking
72 s.setblocking(False)
73
74 if flagDebug:
75     lap2 = chrono.read_ms()
76     print("Tempo apos inicializaco LoRA: {} ms".format(lap2))
77
78 #
79 # Inicializacao e leitura dos sensores
80 #
81 if flagRun:
82     lap1 = chrono.read_ms()
83
84 i2c = I2C(0, I2C.MASTER, pins=('G10', 'G9'), baudrate=400000)
85 connectedI2C = i2c.scan()
86 if flagDebug:
87     print("Connected I2C devices: " + str(connectedI2C))
88
89 ow = OneWire(Pin('G8'))
```

```
90 temp = DS18X20(ow)
91 connectedOW = ow.scan()
92 if flagDebug:
93     print("Connected 1-Wire devices: " + str(connectedOW))
94
95 ilum = [] # Lista com as luminosidades lidas com o MAX44009
96 bmet = [] # Lista com as temperaturas lidas com o BME280
97 bmeh = [] # Lista com as umiadades lidas com o BME280
98 bmes = [] # Lista com as pressões lidas com o BME280
99 owTemp = [] # Lista com as temperaturas lidas com o OneWire
100
101 light_s = False # Flag para indicar se o MAX44009 esta conectado
102 bme_s = False # Flag para indicar se o BME280 esta conectado
103 ow_s = False # Flag para indicar se possui algum sensor 1-Wire conectado
104 if len(connectedOW) > 0:
105     ow_s = True
106
107 connected_i2c = False # Flag para indicar se possui algum sensor I2C
108         conectado
109 if len(connectedI2C) > 0:
110     connected_i2c = True
111
112 if connected_i2c:
113     for device in connectedI2C:
114         if device == 0x4A: # MAX44009 - 74
115             light_sensor = max44009.MAX44009(i2c)
116             light_s = True
117         elif device == 0x76: # BME280 - 118
118             bme = bme280.BME280(i2c=i2c, pressure_mode=bme280.OSAMPLE_8,
119             iir=bme280.FILTER_8)
120             bme_s = True
121         else:
122             if flagDebug:
123                 print("I2C nao reconhecido") # Dispositivo nao cadastrado
124
125 if ow_s:
126     count = 0
127     for sensors in connectedOW:
128         temp.start_conversion(temp.roms[count])
129         count += 1
130
131 if light_s:
132     # Le iluminancia em lux do MAX44009
133     data = int(light_sensor.illuminance_lux)
134     ilum.append(data)
135
136 if bme_s:
137     # Le valores BME280 com media para ter maior precisao :
```

```

135     numreadings = 15
136     samples_temperature = [0.0]*numreadings; mean_temperature = 0.0
137     samples_pressure = [0.0]*numreadings; mean_pressure = 0.0
138     samples_humidity = [0.0]*numreadings; mean_humidity = 0.0
139     for i in range (numreadings):
140         samples_temperature [i] , samples_pressure [i] , samples_humidity [i] =
bme.values
141         mean_temperature = sum( samples_temperature )/len( samples_temperature )
142         mean_pressure = sum( samples_pressure )/len( samples_pressure )
143         mean_humidity = sum( samples_humidity )/len( samples_humidity )
144         t = mean_temperature
145         p = mean_pressure/100    # Pa -> hectoPa
146         h = mean_humidity
147         bmet.append( t )
148         bmep.append( p )
149         bmeh.append( h )
150
151 if ow_s:
152     count = 0
153     for sensor in connectedOW:
154         tempOW = temp.read_temp_async(temp.roms[ count ])
155         owTemp.append( tempOW )
156         count += 1
157
158 if flagDebug:
159     lap3 = chrono.read_ms()
160     print ("Tempo apos sensores: {} ms".format(lap3))
161
162 if flagRun:
163     lap2 = chrono.read_ms()
164     timeSensors=lap2-lap1
165
166 #
167 # Criando pacote Cayenne LPP
168 #
169
170 lpp = cayenneLPP.CayenneLPP( size = 100, sock = s)
171
172 if bme_s:
173     lpp.add_temperature(bmet[0])
174     lpp.add_barometric_pressure(bmep[0])
175     lpp.add_relative_humidity(bmeh[0])
176 if light_s:
177     lpp.add_luminosity(ilum[0])
178
179 owChannel = 150
180

```

```

181 if ow_s:
182     for tempValue in owTemp:
183         val = float(tempValue)
184         lpp.add_temperature(value = val, channel = owChannel)
185         owChannel += 1
186
187 VBAT = myfuncs.get_batt_mV()
188
189 lpp.add_generic(lpp_id=2, values = VBAT, channel = 13, data_size = 2,
190                 is_signed = False, precision = 1)
191
192 if machine.wake_reason()[0] == machine.PWRON_WAKE:
193     pycom.nvs_erase_all()    # 1st time power on
194
195 payloadcount = pycom.nvs_get('count')
196 if payloadcount is not None:
197     payloadcount += 1
198     pycom.nvs_set('count', payloadcount)
199 else:
200     payloadcount = 1
201     pycom.nvs_set('count', 1)  # Starts from 1
202 if flagDebug:
203     print("# pacote LoRaWan = {}".format(payloadcount))
204
205 lpp.add_luminosity( value = payloadcount, channel = 155) # Numero do Pacote
206             enviado
207
208 if flagDebug:
209     print("Tamanho do pacote LPP = {} bytes".format(lpp.get_size()))
210
211 if flagDebug:
212     lap4 = chrono.read_ms()
213     print("Tempo antes do LPP send: {} ms".format(lap4))
214
215 if flagRun:
216     lap3 = chrono.read_ms()
217     totalAwake = lap3+210+3
218
219 lpp.add_luminosity( value = timeSensors, channel = 158) # Tempo dos
220             sensores (inicializacao + leitura)
221 lpp.add_luminosity( value = totalAwake, channel = 159) # Tempo total
222             acordado
223 lpp.send(reset_payload = True)    # Envio do pacote LoRaWan usando Cayenne
224             LPP
225
226 if flagDebug:
227     lap5 = chrono.read_ms()

```

```
223     print("Tempo apos LPP send: {} ms".format(lap5))  
224  
225 time.sleep_ms(300)    # pausa para garantir o envio do pacote  
226  
227 #  
228 # Entrando em deepsleep  
229 #  
230  
231 if flagDebug:  
232     print("Entrando em deepsleep por 600s = 10 minutos...")  
233  
234 machine.deepsleep(600*1000)    # deep sleep por 10 minutos
```

Referências

- 1 ADRIAN, R. et al. Lakes as sentinels of climate change. *Limnology and Oceanography*, v. 54, p. 2283–2297, 2009. Citado na página 20.
- 2 SCHINDLER, D. W. Lakes as sentinels and integrators for the effects of climate change on watersheds, airsheds and landscapes. *Limnology and Oceanography*, v. 54, p. 2349–2358, 2009. Citado na página 20.
- 3 STRASKRABA, M.; TUNDISI, J. G. Gerenciamento da qualidade da água em reservatórios. *International Lake Environmental Committee*, São Carlos, p. 280, 2000. Citado na página 20.
- 4 GLEON. *Global Lake Ecological Observatory Network*. 2018. <<http://gleon.org>>. [Online; acessado em 06 de Agosto de 2018]. Citado na página 20.
- 5 LOUREIRO, A. A. F. Minicursos XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. In: FREITAS FABÍOLA GONÇALVES PEREIRA GREVE, F. A. S. A. E. S.; LUNG, L. C. (Ed.). *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.]: Sociedade Brasileira de Computação, 2016. p. 1–50. Citado na página 21.
- 6 HELAL, A. A. et al. Estacqua: Proposta de solução integrada de hardware, software e internet das coisas para monitoramento ambiental. *XLV Seminário Integrado de Software e Hardware*, XXVI Congresso da Sociedade Brasileira de Computação, Natal, Júlio 2018. Citado na página 24.
- 7 TOSE, T. *Rede de Sensores sem fio Zigbee Aplicada em uma Estação de Tratamento de Esgoto*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Av. Fernando Ferrari, 514 - Goiabeiras, Vitória - ES, 29075-073, 11 2012. Citado na página 25.
- 8 OLIVEIRA, F. F. de. EstGeoMag: Integrando soluções de Hardware, Software e Internet das Coisas na medição de grandezas Geomagnéticas. *Sociedade Brasileira de Computação*, p. 2598–2609, 2017. Citado na página 25.
- 9 MOIS, G.; FOLEA, S.; SANISLAV, T. Analysis of three IoT-Based wireless sensors for environmental monitoring. *IEEE Transactions on Instrumentation and Measurement*, v. 66, n. 8, p. 2056–2064, Aug 2017. ISSN 0018-9456. Citado na página 25.
- 10 TZORTZAKIS, K.; PAPAFOTIS, K.; SOTIRIADIS, P. P. Wireless self powered environmental monitoring system for smart cities based on lora. In: *2017 Panhellenic Conference on Electronics and Telecommunications (PACET)*. [S.l.: s.n.], 2017. p. 1–4. Citado na página 25.
- 11 AUGUSTIN, A. et al. A study of lora: Long range and low power networks for the internet of things. *Sensors*, v. 16, n. 9, 2016. ISSN 1424-8220. Disponível em: <<http://www.mdpi.com/1424-8220/16/9/1466>>. Citado na página 26.
- 12 BHARATHKUMAR, V. et al. Microcontroller based digital meter with alert system using GSM. *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, p. 444–448, Jan 2017. Citado na página 26.

- 13 TSCHOFENIG, H. et al. *Architectural Considerations in Smart Object Networking*. [S.l.], 2015. 1-24 p. Disponível em: <<https://www.rfc-editor.org/rfc/rfc7452.txt>>. Citado 4 vezes nas páginas 27, 28, 29 e 30.
- 14 ROSE, K.; ELDRIDGE, S.; CHAPIN, L. *The Internet Of Things: An Overview*. [S.l.], 2015. 1-24 p. Disponível em: <<https://www.internetsociety.org/resources/doc/2015/iot-overview>>. Citado 3 vezes nas páginas 27, 28 e 29.
- 15 Link-Labs. *Low Power, Wide Area Networks*. [S.l.], 2016. [Online; acessado em 18 de Agosto de 2018]. Citado 5 vezes nas páginas 29, 30, 31, 32 e 33.
- 16 WIKIPÉDIA. *Redes Mesh — Wikipédia, a encyclopédia livre*. 2018. [Online; acessado em 19 de Agosto de 2018]. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Redes_Mesh&oldid=52737683>. Citado na página 31.
- 17 Link-Labs. *New Industrial Internet of Things Products*. [S.l.], 2016. [Online; acessado em 19 de Agosto de 2018]. Citado 2 vezes nas páginas 31 e 32.
- 18 Sigfox. *Messages per day/hour*. 2017. <<https://ask.sigfox.com/questions/2968/messages-per-dayhour.html>>. [Online; acessado em 19 de Agosto de 2018]. Citado na página 33.
- 19 Weightless. *Ubiik Weightless Starter Kit*. 2018. <<https://www.ubiik.com/starterkit>>. [Online; acessado em 19 de Agosto de 2018]. Citado na página 33.
- 20 Pycom. *LoPy4*. 2018. <<https://pycom.io/product/lopy4/>>. [Online; acessado em 19 de Agosto de 2018]. Citado na página 33.
- 21 ALLIANCE, L. *A technical overview of LoRa and LoRaWAN*. [S.l.], 2015. Citado 8 vezes nas páginas 33, 34, 36, 39, 40, 41, 43 e 45.
- 22 RUANO, E. *LoRa protocol Evaluations, limitations and practical test*. [S.l.], 2016. Citado 3 vezes nas páginas 33, 37 e 43.
- 23 Semtech. *LoRa Technology*. 2018. <<https://www.semtech.com/technology/lora>>. [Online; acessado em 05 de Julho de 2018]. Citado 5 vezes nas páginas 33, 37, 39, 40 e 43.
- 24 LoRa Alliance. *LoRa Alliance*. 2018. <<https://www.lora-alliance.org>>. [Online; acessado em 05 de Julho de 2018]. Citado na página 33.
- 25 International Telecommunications Union. *International Telecommunications Union*. 2018. <<https://www.itu.int/net/ITU-R/terrestrial/faq/index.html>>. [Online; acessado em 05 de Julho de 2018]. Citado na página 34.
- 26 P. Lea. *Internet of Things for Architects*. 1. Birmingham: Packt, 2018. 514 p. ISBN 978-1-78847-059-9. Citado 4 vezes nas páginas 34, 36, 42 e 43.
- 27 Wikipedia contributors. *Chirp — Wikipedia, The Free Encyclopedia*. 2018. <<https://en.wikipedia.org/w/index.php?title=Chirp&oldid=848499687>>. [Online; acessado em 7 de Julho de 2018]. Citado 2 vezes nas páginas 34 e 35.
- 28 HAYKIN, S.; VEEN, B. V. *Signals and Systems*. Segunda. [S.l.]: Wiley, 2005. 820 p. ISBN 978-0471707899. Citado 2 vezes nas páginas 35 e 40.

- 29 MAGRIN, D. *Network level performances of a LoRa system*. Dissertação (Mestrado) — Università degli Studi di Padova, Itália, 2016. Citado 2 vezes nas páginas 35 e 36.
- 30 Wikipedia contributors. *Code rate* — Wikipedia, The Free Encyclopedia. 2017. <https://en.wikipedia.org/w/index.php?title=Code_rate&oldid=805689736>. [Online; acessado em 10 de Julho de 2018]. Citado na página 35.
- 31 Wikipedia contributors. *Orthogonal frequency-division multiplexing* — Wikipedia, The Free Encyclopedia. 2018. <https://en.wikipedia.org/w/index.php?title=Orthogonal_frequency-division_multiplexing&oldid=846026302>. [Online; acessado em 10 de Julho de 2018]. Citado na página 36.
- 32 PETÄJÄRVI, J. et al. Performance of a low-power wide-area network based on lora technology: Doppler robustness, scalability, and coverage. *International Journal of Distributed Sensor Networks*, v. 13, n. 3, p. 1–16, Fevereiro 2017. Citado na página 36.
- 33 CORPORATION, S. *SX1272/3/6/7/8: LoRa® Guide*. [S.l.], 2013. Citado na página 38.
- 34 CORPORATION, S. *LoRaWAN 1.1 Regional Parameters*. [S.l.], 2017. Citado 2 vezes nas páginas 39 e 65.
- 35 Agência Nacional de Telecomunicações. *Resolução número 506, de 1 de julho de 2008*. 2008. <<http://www.anatel.gov.br/legislacao/resolucoes/23-2008/104-resolucao-506>>. [Online; acessado em 10 de Julho de 2018]. Citado na página 39.
- 36 Wikipedia contributors. *Frequency-shift keying* — Wikipedia, The Free Encyclopedia. 2018. <https://en.wikipedia.org/w/index.php?title=Frequency-shift_keying&oldid=849129099>. [Online; acessado em 13 de July de 2018]. Citado na página 40.
- 37 EUROPEAN TELECOMMUNICATIONS STANDARD INSTITUTE. Electromagnetic compatibility and radio spectrum matters (erm); short range devices (srd); radio equipment to be used in the 25 mhz to 1 000 mhz frequency range with power levels ranging up to 500 mw; part 2: Harmonized en covering essential requirements under article 3.2 of the r&tte directive. Sophia Antipolis, France, Janeiro 2012. Citado na página 41.
- 38 SEMTECH. Fcc regulations for ism band devices: 902 - 928 mhz. Sophia Antipolis, France, Janeiro 2006. Citado na página 41.
- 39 AGÊNCIA NACIONAL DE TELECOMUNICAÇÕES. Resolução nº 680, de 27 de junho de 2017. aprova o regulamento sobre equipamentos de radiocomunicação de radiação restrita e altera o regulamento dos serviços de telecomunicações, o regulamento de gestão da qualidade do serviço de comunicação multimídia, o regulamento do serviço de comunicação multimídia e o regulamento do serviço limitado privado. Brasília, DF, Junho 2017. Citado na página 41.
- 40 AGÊNCIA NACIONAL DE TELECOMUNICAÇÕES. Radiofreqüências nas faixas de 800 mhz, 900 mhz, 1.800 mhz, 1.900 mhz e 2.100 mhz. Brasília, DF, Dezembro 2006. Citado 3 vezes nas páginas 41, 42 e 70.
- 41 Agência Nacional de Telecomunicações. *Ocupação faixa 900 MHz*. 2018. <https://en.wikipedia.org/w/index.php?title=Effective_radiated_power&oldid=840164259>. [Online; acessado em 12 de Julho de 2018]. Citado na página 41.

- 42 Electronic Code of Federal Regulations. *Operation within the bands 902-928 MHz, 2400-2483.5 MHz, and 5725-5850 MHz*. 2018. <https://www.ecfr.gov/cgi-bin/text-idx?SID=87778306ac92bc1fb4bf00a16ca7f90b&mc=true&node=se47.1.15_1247&rgn=div8>. [Online; acessado em 13 de Julho de 2018]. Citado na página 42.
- 43 Agência Nacional de Telecomunicações. *Proposta de Requisitos Técnicos para a Avaliação da Conformidade dos Equipamentos de Radiocomunicação de Radição Restrita*. 2018. <https://sei.anatel.gov.br/sei/modulos/pesquisa/md_pesq_documento_consulta_externa.php?eEP-wqk1skrd8hSlk5Z3rN4EVg9uLJqrLYJw_9INcO6bCnKD6vz-ByUJW_O78ARThdLvuOBPx7Jf17lNA_7xNLNv7d4eYLPmX6qHb_tI84VXpIVtizqvYjwvxjjSb6r>. [Online; acessado em 13 de Julho de 2018]. Citado na página 42.
- 44 INTERNET OF THINGS ALLIANCE AUSTRALIA. Spectrum available for iot. Sophia Antipolis, France, Maio 2016. Citado na página 42.
- 45 Wikipedia contributors. *Node (networking) — Wikipedia, The Free Encyclopedia*. 2018. <[https://en.wikipedia.org/w/index.php?title=Node_\(networking\)&oldid=848629184](https://en.wikipedia.org/w/index.php?title=Node_(networking)&oldid=848629184)>. [Online; acessado em 13 de Julho de 2018]]. Citado na página 42.
- 46 Wikipedia contributors. *Star network — Wikipedia, The Free Encyclopedia*. 2018. <https://en.wikipedia.org/w/index.php?title=Star_network&oldid=848944891>. [Online; acessado em 14 de Julho de 2018]. Citado na página 42.
- 47 A. S. Tanenbaum and D. J. Wetherall. *Computer Networks*. 5. Massachusetts: Pearson, 2011. 962 p. ISBN 978-0-13-212695-3. Citado na página 43.
- 48 R. Merritt. *ST to Make LoRa IoT Chips*. 2015. <https://www.eetimes.com/document.asp?doc_id=1328489>. [Online; acessado em 14 de Juho de 2018]. Citado na página 46.
- 49 Johan Stokking. *Firmware Updates over Low-Power Wide Area Networks*. 2017. [Online; acessado em 12 de Agosto de 2018]. Disponível em: <<https://www.thethingsnetwork.org/article/firmware-updates-over-low-power-wide-area-networks>>. Citado na página 47.
- 50 Particle. *Selecting a microcontroler for your IoT product*. 2018. <<https://www.particle.io/mcu-vs-soc-vs-microprocessor-for-iot/>>. [Online; acessado em 19 de Agosto de 2018]. Citado na página 47.
- 51 WIKIPÉDIA. *System-on-a-chip — Wikipédia, a encyclopédia livre*. 2017. [Online; acessado em 19 de Agosto de 2018]. Disponível em: <<https://pt.wikipedia.org/w/index.php?title=System-on-a-chip>>. Citado na página 47.
- 52 GERBER, A. Choosing the best hardware for your next iot project. *IBM*, 2018. Disponível em: <<https://www.ibm.com/developerworks/library/iot-lp101-best-hardware-devices-iot-project/iot-lp101-best-hardware-devices-iot-project-pdf.pdf>>. Citado na página 48.
- 53 ARDUINO. *Arduino Uno Rev3*. 2018. [Online; acessado em 20 de Agosto de 2018]. Disponível em: <<https://store.arduino.cc/usa/arduino-uno-rev3>>. Citado na página 49.
- 54 PYBOARD. *MicroPython pyboard feature table*. 2018. [Online; acessado em 20 de Agosto de 2018]. Disponível em: <<https://store.micropython.org/pyb-features>>. Citado na página 49.

- 55 Pycom. *LoPy4*. [S.l.], 2017. [Online; acessado em 27 de Junho de 2018]. Citado 2 vezes nas páginas 49 e 72.
- 56 DRAGINO. *LoRa Shield for Arduino*. 2018. [Online; acessado em 20 de Agosto de 2018]. Disponível em: <<http://www.dragino.com/products/module/item/102-lora-shield.html>>. Citado na página 48.
- 57 DRAGINO. *LoRa Shield for Arduino*. 2018. [Online; acessado em 20 de Agosto de 2018]. Disponível em: <<https://www.tindie.com/products/edwin/lora-shield-long-distance-wireless-433868915mhz/>>. Citado na página 48.
- 58 ADAFRUIT. *Adafruit RFM95W LoRa Radio Transceiver Breakout - 868 or 915 MHz - RadioFruit*. 2018. [Online; acessado em 20 de Agosto de 2018]. Disponível em: <<https://www.adafruit.com/product/3072>>. Citado na página 48.
- 59 Pycom. *Pycom - Next Generation Internet of Things Platform*. 2018. <<https://pycom.io/>>. [Online; acessado em 14 de Junho de 2018]. Citado 2 vezes nas páginas 51 e 52.
- 60 Espressif. *ESP32: A Different IoT Power and Performance*. 2018. <<https://www.espressif.com/en/products/hardware/esp32/overview>>. [Online; acessado em 09 de Agosto de 2018]. Citado na página 51.
- 61 Espressif. *Sleep Modes*. 2018. <http://esp-idf.readthedocs.io/en/latest/api-reference/system/sleep_modes.html>. [Online; acessado em 10 de Agosto de 2018]. Citado na página 52.
- 62 MAXIM Integrated. *Industry's Lowest-Power Ambient Light Sensor with ADC*. [S.l.], 2011. Citado 2 vezes nas páginas 53 e 54.
- 63 eBay. *MAX44009 Ambient Light Sensor I2C Digital Output Module Development Board Module*. 2018. <https://www.ebay.com/itm/MAX44009-Ambient-Light-Sensor-I2C-Digital-Output-Module-Development-Board-Module/142215092528?hash=item211cae1930%3Ag%3ANXIAOSw0w9YVQ2H&_sacat=0&_nkw=max44009&_from=R40&rt=nc&LH_TitleDesc=0>. [Online; acessado em 15 de Junho de 2018]. Citado na página 54.
- 64 Bosch. *BME280: Final data sheet*. [S.l.], 2016. Citado 2 vezes nas páginas 55 e 56.
- 65 DX. *BME280-3.3 High Precision Barometric Pressure Module*. 2018. <<http://www.dx.com/p/gy-bme280-3-3-high-precision-barometric-pressure-module-purple-436672#W0uzTy2ZOqQ>>. [Online; acessado em 15 de Junho de 2018]. Citado na página 55.
- 66 Amazon. *DS18B20 Stainless Steel package 1 Meters Waterproof DS18b20 Temperature Probe Sensor 18B20 Regard*. 2018. <https://www.amazon.co.uk/DS18B20-Stainless-Waterproof-Temperature-Regard/dp/B07BFT5K23/ref=sr_1_8?s=computers&ie=UTF8&qid=1531697106&sr=1-8&keywords=ds18b20>. [Online; acessado em 15 de Junho de 2018]. Citado na página 56.
- 67 MAXIM Integrated. *DS18B20: Programmable Resolution 1-Wire Digital Thermometer*. [S.l.], 2015. Citado 2 vezes nas páginas 57 e 58.
- 68 The Things Network. *The Things Network*. [S.l.], 2018. [Online; acessado em 16 de Junho de 2018]. Citado 3 vezes nas páginas 59, 60 e 72.

- 69 The Things Network. *Ground breaking world record! LoRaWAN packet received at 702 km (436 miles) distance.* 2017. <<https://www.thethingsnetwork.org/article/ground-breaking-world-record-lorawan-packet-received-at-702-km-436-miles-distance>>. [Online; acessado em 16 de Junho de 2018]. Citado na página 59.
- 70 The Things Network. *Packetworx and The Things Network expand LoRaWAN deployment with over 2500 gateways across Philippines.* 2018. <<https://thethingsnetwork.pr.co/167432-packetworx-and-the-things-network-expands-lorawan-deployment-with-over-2500-gateways-ac>>. [Online; acessado em 16 de Junho de 2018]. Citado na página 60.
- 71 TTN. *LoRaWAN Network Stack.* 2018. [Online; acessado em 22 de Agosto de 2018]. Disponível em: <<https://www.thethingsnetwork.org/tech-stack>>. Citado na página 61.
- 72 LORIOT. *The future of Internet of Things is at long range.* 2018. [Online; acessado em 21 de Agosto de 2018]. Disponível em: <<https://loriot.io>>. Citado na página 62.
- 73 LORASERVER. *LoRa Server, open-source LoRaWAN network-server.* 2018. [Online; acessado em 21 de Agosto de 2018]. Disponível em: <<https://www.loraserver.io>>. Citado na página 62.
- 74 LORAWAN-SERVER. *Compact server for private LoRa networks.* 2018. [Online; acessado em 21 de Agosto de 2018]. Disponível em: <<https://gotthardp.github.io/lorawan-server/>>. Citado na página 62.
- 75 Cayenne. *Cayenne.* 2018. <<https://mydevices.com/cayenne/docs/intro/>>. [Online; acessado em 16 de Junho de 2018]. Citado 2 vezes nas páginas 62 e 110.
- 76 Wikipedia contributors. *Lithium polymer battery — Wikipedia, The Free Encyclopedia.* 2018. <https://en.wikipedia.org/w/index.php?title=Lithium_polymer_battery&oldid=855559272>. [Online; accessado em 24 de Agosto de 2018]. Citado na página 63.
- 77 PowerTech. *Primary cells Lithium Thionyl Chloride (Li-SOCl₂).* 2018. <<https://www.powertechsystems.eu/home/products/primary-cells-lithium-thionyl-chloride-li-socl2/>>. [Online; accessado em 24 de Agosto de 2018]. Citado na página 64.
- 78 The Things Network. *LoRaWAN Frequencies Overview.* 2018. <<https://www.thethingsnetwork.org/docs lorawan/frequency-plans.html>>. [Online; acessado em 18 de Junho de 2018]. Citado na página 65.
- 79 Semtech. *SX1272/3/6/7/8 LoRa Modem Design Guide.* [S.I.], 2018. [Online; acessado em 16 de Junho de 2018]. Citado na página 66.
- 80 The Things Network. *Duty Cycle for LoRaWAN Devices.* 2018. <<https://www.thethingsnetwork.org/docs lorawan/duty-cycle.html#maximum-duty-cycle>>. [Online; acessado em 20 de Junho de 2018]. Citado na página 67.
- 81 LoRaTools. *Calculate the air time of your LoRa frame.* 2018. <<https://www.loratools.nl/#airtime>>. [Online; acessado em 19 de Junho de 2018]. Citado na página 67.
- 82 The Things Network. *Limitations of LoRaWAN.* 2018. <<https://www.thethingsnetwork.org/docs lorawan/limitations.html>>. [Online; acessado em 24 de Junho de 2018]. Citado na página 72.

- 83 Semtech. *LoRaTM Modulation Basics*. [S.l.], 2015. [Online; acessado em 27 de Junho de 2018]. Citado na página 72.
- 84 VOIGT, T. et al. Mitigating inter-network interference in lora networks. In: _____. *EWSN '17 Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*. [S.l.]: ACM Press, 2017. p. 323–328. ISBN 9780994988614. © Junction Publishing, 2017. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in EWSN '17 Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks. Citado 2 vezes nas páginas 72 e 75.
- 85 Cisco. *Cisco Wireless Gateway for LoRaWAN Data Sheet*. [S.l.], 2018. [Online; acessado em 27 de Junho de 2018]. Citado na página 72.
- 86 The Things Network. *Semtech UDP Packet Forwarder*. 2018. <<https://www.thethingsnetwork.org/docs/gateways/packet-forwarder/semtech-udp.html>>. [Online; acessado em 30 de Julho de 2018]. Citado na página 72.
- 87 The Things Network. *Semtech UDP protocol*. 2018. <<https://www.thethingsnetwork.org/docs/gateways/start/connection.html#semtech-udp-protocol>>. [Online; acessado em 30 de Julho de 2018]. Citado na página 72.
- 88 The Things Network. *Single-channel gateways*. 2018. <<https://www.thethingsnetwork.org/docs/gateways/start/single-channel.html>>. [Online; acessado em 30 de Julho de 2018]. Citado na página 73.
- 89 Lancaster University. *LoRaSim*. 2018. <<http://www.lancaster.ac.uk/scc/sites/lora/lorasim.html>>. [Online; acessado em 30 de Junho de 2018]. Citado na página 73.
- 90 BOR, M. et al. Do lora low-power wide-area networks scale? In: _____. *MSWiM '16 Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. [S.l.]: ACM Press, 2016. p. 59–67. A Corrected version. A bug was found in the simulator, whereby previously collided packets would be in some cases marked as ‘not collided’, thereby overestimating the goodput of the network. The bug was fixed and all figures and numbers have been updated accordingly. The overall conclusions, however, has stayed the same. Citado 3 vezes nas páginas 73, 74 e 75.
- 91 M. Grudsky and R. Gilson. *LoRaWAN CAPACITY TRIAL IN DENSE URBAN ENVIRONMENT*. 2018. <https://s26168.pcdn.co/wp-content/uploads/2018/03/machineQ_LoRaWan_Capacity_Trial.pdf>. [Online; acessado em 31 de Julho de 2018]. Citado 2 vezes nas páginas 74 e 75.
- 92 B. Ray. *Use Cases and Considerations for LoRaWAN*. 2016. <<https://www.link-labs.com/blog/use-cases-and-considerations-for-lorawan>>. [Online; acessado em 31 de Julho de 2018]. Citado na página 75.
- 93 GitHub. *EstAcqua*. 2018. <<https://github.com/helalbr/estacqua>>. [Online; acessado em 10 de Agosto de 2018]. Citado na página 80.
- 94 EduRoam. *EduRoam*. 2018. <<https://www.eduroam.org>>. [Online; acessado em 11 de Agosto de 2018]. Citado na página 93.

- 95 BARROSO, G. F. et al. Estudos integrados no sistema lacustre do Baixo Rio Doce (Espírito Santo). *I Seminário Nacional de Gestão Sustentável de Ecossistemas Aquáticos: Complexidade, Interatividade e Ecodesenvolvimento*, Arraial do Cabo, 2012. Citado 2 vezes nas páginas 95 e 98.
- 96 Hylke Visser. *LinkedIn*. 2018. <<https://www.linkedin.com/in/htdvisser>>. [Online; acessado em 12 de Agosto de 2018]. Citado na página 97.
- 97 Hylke Visser. *Some Single-Channel Gateways stopped working*. [S.l.], 2017. [Online; acessado em 12 de Agosto de 2018]. Citado 2 vezes nas páginas 97 e 98.
- 98 TI. *INA219*. [S.l.], 2015. [Online; acessado em 10 de Agosto de 2018]. Citado na página 101.
- 99 Minipa. *ET-2082D*. [S.l.], 2018. [Online; acessado em 11 de Agosto de 2018]. Citado na página 101.
- 100 Texas Instrument. *Characteristics of Rechargeable Batteries*. 2011. <<http://www.ti.com/lit/an/snva533/snva533.pdf>>. [Online; accessado em 13 de Novembro de 2018]. Citado na página 103.
- 101 Amazon. *HOBO by Onset*. 2011. <<https://www.amazon.com/Onset-BASE-U-4-Optic-Base-Station/dp/B00UZED8BY>>. [Online; accessado em 27 de agosto de 2018]. Citado na página 106.
- 102 IPSO. *Enabling IoT Devices Hardware and Software Interoperability*. [S.l.], 2018. [Online; acessado em 16 de Junho de 2018]. Citado na página 110.