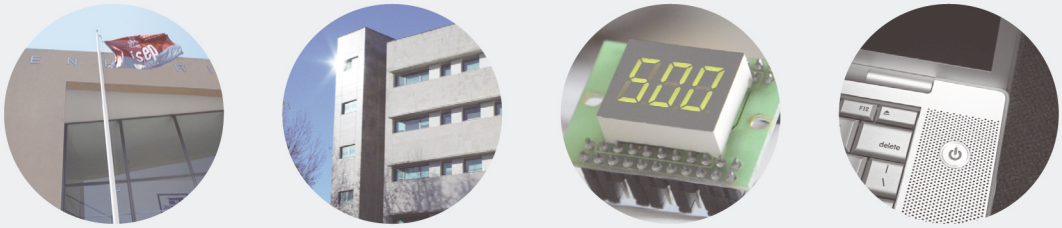
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Sistema IoT de Comunicação e Monitorização de Dados para Aplicação em Smart Cities

RUI PAULO DA SILVA CARVALHO
novembro de 2021

POLITÉCNICO
DO PORTO

SISTEMA IOT DE COMUNICAÇÃO E MONITORIZAÇÃO DE DADOS PARA APLICAÇÃO EM *SMART CITIES*

Rui Paulo da Silva Carvalho

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto

2021

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Rui Paulo da Silva Carvalho, Nº 1160961, 1160961@isep.ipp.pt

Orientação científica: Cecília Maria do Rio Fernandes Moreira Reis, cmr@isep.ipp.pt

Coorientação: Nuno Alexandre Neto Dias, nnd@isep.ipp.pt

Empresa: PAVNEXT – Technological Pavements, Lda

Supervisão: Francisco João Anastácio Duarte, fd.pavnext@gmail.com

Mestrado em Engenharia Electrotécnica e de Computadores

Área de Especialização de Automação e Sistemas

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

18 de novembro de 2021

Agradecimentos

Os meus agradecimentos são dirigidos à minha família, pelo apoio durante todo o meu percurso educativo, durante os bons e maus momentos. Quero agradecer também aos meus colegas de estágio, pela convivência e ajuda mútua durante o desenvolvimento dos projetos. Gostava ainda de salientar o papel de suporte dos orientadores que contribuíram com o seu *feedback* para enaltecer este projeto.

Resumo

Atualmente, a procura por soluções IoT (*Internet of Things*) está a crescer exponencialmente, uma vez que estes sistemas permitem o acesso e controlo de aplicações, assim como a monitorização, dos seus comportamentos. No âmbito das *smart cities*, estes sistemas podem ter uma grande relevância, em questões como a segurança de condutores e peões, assim como analisar dados gerados pelo próprio sistema.

Assim, o objetivo principal do sistema, apresentado nesta dissertação, consiste na receção e transmissão de informação, proveniente de sensores e de outros sistemas, com base em diversos protocolos de comunicação, para uma base de dados, permitindo igualmente a monitorização do sistema.

Neste projeto, uma parte substancial da informação provém de um sistema externo (pavimento NEXT-Road) que gera dados com a passagem dos veículos pelo pavimento. Desta forma, as velocidades de entrada e saída dos veículos, o peso e a contagem de veículos, são informações que se consideram relevantes para a monitorização do tráfego em circulação na rodovia. São também considerados dados de temperatura, humidade e energia gerada, para a obtenção de um *feedback* do estado do sistema NEXT-Road. Além do pavimento, foram também adicionados sensores ambientais com o objetivo de detetar o estado do tempo no local de implementação do sistema. Toda esta informação é transmitida, através de uma tecnologia LPWAN, de forma a conseguir encaminhar a informação, a distâncias consideráveis, até a um servidor.

A solução encontrada constituiu no desenvolvimento de *firmware* para um microcontrolador, o qual recebe a informação proveniente dos sensores e dos sistemas externos. Seguidamente, os dados são organizados para serem enviados através de LoRa para uma *gateway* que, consequentemente, encaminha as mensagens para um servidor LoRaWAN. Os dados são obtidos do servidor, através de mensagens MQTT e é efetuada uma segunda organização, de tal forma a serem guardados na respetiva na base de dados.

Adicionalmente, é usada a ferramenta de monitorização Grafana, no sentido de apresentar o conteúdo da base de dados em ambiente gráfico, com recurso a *dashboards* e de acordo uma linha temporal.

É possível afirmar que o sistema desenvolvido apresenta uma grande flexibilidade de implementação, pois tem a capacidade de se adaptar aos diferentes formatos dos dados.

Durante o desenvolvimento do projeto, foram realizados testes, de forma a validar as diferentes etapas. No fim, com todos os componentes operacionais, o sistema foi validado com os testes realizados em ambiente de laboratório e o um teste piloto realizado em Matosinhos.

O sistema desenvolvido teve os resultados esperados em ambiente de testes e no terreno, funcionando em aplicações reais, junto à via rodoviária.

Palavras-Chave

Sistema IoT, *smart cities*, LPWAN, LoRaWAN, séries temporais, Grafana, Monitorização.

Abstract

Currently, the demand for IoT (Internet of Things) solutions is growing exponentially, since these systems allow access and control of applications as well as monitoring their behavior. In the context of smart cities, these systems can have a great relevance in issues such as the safety of drivers and pedestrians, as well as analyze data generated by the system itself.

Thus, the main objective of the system, presented in this dissertation, consists in receiving and transmiting information from sensors and other systems, based on several communication protocols, to a database, also allowing the monitoring of the system.

In this project, a substantial part of the information comes from na external system (NEXT-Road flooring) that generates data as the vehicles pass over the flooring. Thus, vehicle entry and exit speeds, weight and vehicle count, are all the information that is considered relevant for monitoring the traffic on the road. Temperature, humidity, and generated energy data from the floor are also considered, to obtain feedback on the status of the NEXT-Road system. In addition to the flooring data, environmental sensors are added in order to detect the state of the weather at the system's implementation site. All this information is transmitted through LPWAN technology, in order to be able to route the information over considerable distances, to a server.

The solution was to develop firmware for a microcontroller, which receives the information from the sensors and external systems. The data is then organized to be sent via LoRa to a gateway, which consequently forwards the messages to a LoRaWAN server. The data are retrieved from the server, through MQTT messages, and a second arrangement is made, in such way that it is stored in the respective database.

Additionally, the monitoring tool Grafana is used, in order to present the database contents in a graphic environment, through dashboards, according to a timeline.

It is possible to state that the system developed presents a great flexibility of implementation, because it has the ability to adapt to diferente data formats.

During the project's development, tests were performed, in order to validate the diferente stages. At the end, with all the components operational, the system was validated with tests performed in a laboratory environment and a pilot test carried out in Matosinhos.

The developed system had the expected results in a test environment and in the field, working in real applications, close to the roadway.

Keywords

IoT system, smart cities, LPWAN, LoRaWAN, time series, Grafana, Monitoring.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

3GPP	–	3rd Generation Partnership Project
ABP	–	Activation By Personalization
ADC	–	Analog-to-Digital Converter
AGC	–	Automatic Gain Control
AMQP	–	Advanced Message Queuing Protocol
API	–	Application Programming Interface
BSEC	–	Bosch Sensortec Environmental Foundation
BSON	–	Binary JavaScript Object Notation
CAN FD	–	Controller Area Network Flexible Data-Rate
CCTV	–	Closed-Circuit Television
CNCF	–	Common Business Oriented Language
COBOL	–	Elemento de Rede
CODASYL	–	Committe on Data System Language
CR	–	Coding Rate
CSS	–	Chirp Spread Spectrum
DBMS	–	Database Management System
DBPSK	–	Differential Binary Phase Shift Keying
DNS	–	Domain Name System
FDMA	–	Frequency Division Multiple Access
FEC	–	Foward Error Correction

GMS – Global System for Mobile Communication

GPIO – General Purpose Input/Output

HTTP – Hypertext Transfer Protocol

IAQ – Index for Air Quality

IMS – Information Management System

ISM – Industrial, Scientific and Medical

IoT – Internet of Things

I²C – Inter-Integrated Circuit

JSON – JavaScript Object Notation

LCD – Liquid Crystal Display

LNA – Low Noise Amplifier

LORA – Long Range

LPWAN – Low-Power Wide-Area Network

LTE – Long Term Evolution

M2M – Machine to Machine

MAC – Media Access Control

MQTT – Message Queuing Telemetry Transport

NTP – Network Time Protocol

OFDMA – Orthogonal Frequency Division Multiple Access

OSI – Open System Interconnection

OTAA – Over-The-Air Activation

PCB – Printed Circuit Board

QPSK – Quadrature Phase Shift Keying

RAID – Redundant Array of Inexpensive Drives

RDBMS – Relational Database Management System

SAW – Surface Acoustic Wave

SF – Spreading Factor

SNR – Signal-to-noise ratio

SPI – Serial Peripheral Interface

SQL – Strutured Query Language

SSD – Solid-state Drive

TCP/IP – Transmission Control Protocol/Internet Protocol

TOA – Time on Air

TSBD – Time Series Database

TTN – The Things Network

TTS – The Things Stack

UDP – User Datagram Protocol

UI – User Interface

UNB – Ultra-Narrow Band

UPTEC – Science and Technology Park of University of Porto

USART – Universal Synchronous Asynchronous Receiver Transmitter

USB – Universal Serial Bus

VOCs – Volatile Organic Compounds

WLAN – Wireless Local Area Network

XML – Extensible Markup Language

1. INTRODUÇÃO

O interesse pela gestão eficiente de recursos disponíveis e pelo desenvolvimento de soluções inteligentes, fez do conceito *Internet of Things* (IoT), um foco de prosperidade e de oportunidade para empresas interessadas em elevar e disponibilizar serviços/produtos tecnologicamente avançados. O investimento e consequente avanço em sistemas embebidos, comunicação *wireless*, internet e gestão eficiente de dados, têm um papel fulcral para a realização de sistemas IoT [1].

A *Internet of Things* pode ser definida como um sistema de comunicação de informação em rede, entre dispositivos computacionais, não ocorrendo qualquer interação Homem-Máquina ou Homem-Homem. Estes dispositivos, "*things*", estão usualmente equipados com sensores e *software*, que permitem a agregação e transmissão de dados entre si. Estas ocorrências, possibilitam o desempenho de ações no sistema, ou a retenção da informação para posterior análise [1].

A conceção de dados, durante o funcionamento de grande parte destes sistemas, revela-se exorbitante, surgindo o conceito de *Big Data*. Este define-se pelos grandes volumes de informação estruturada ou não estruturada, gerados em tempo-real ou em curto período de tempo. Em aplicações mais frequentes, a informação é armazenada em bases de dados, que consigam gerir este grande volume de informação. Adicionalmente, podem ser utilizadas ferramentas de monitorização em conjugação com a informação disponível na base de dados [2]. A Figura 1 exibe um exemplo simples das etapas de um sistema IoT.

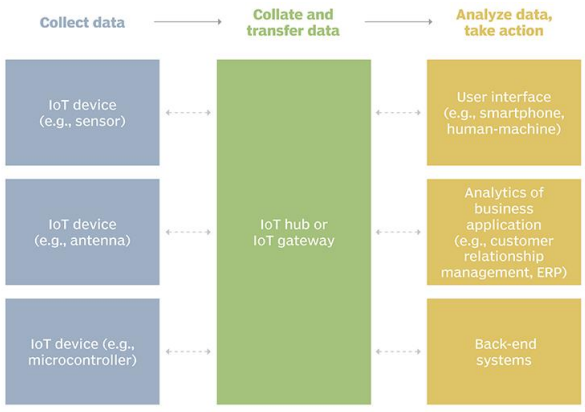


Figura 1 Exemplo de um sistema IoT. Adaptado de [3]

A implementação e desenvolvimento de tecnologias IoT, impulsionaram o aparecimento de novas soluções no âmbito das cidades, denominadas de *smart cities*. O aparecimento do conceito de *smart cities*, pode ser atribuído a um conjunto de fatores, como o crescimento populacional, o desenvolvimento das cidades, o crescimento económico dos países, entre outros [1].

De acordo com uma publicação do departamento de assuntos económicos e sociais das nações unidas, é espectável que no ano 2050 a população a habitar centros urbanos aumente para mais 2.5 mil milhões de pessoas, o que corresponderá a 68% da população [4]. Este crescimento nas cidades, promove a discussão de temas como a urbanização, melhoria da qualidade de vida e a sustentabilidade. Pelo que, a aposta para desafios de gestão e desenvolvimento das cidades, tem como foco as aplicações IoT [1].

As aplicações IoT nas cidades, têm como funções essenciais, a recolha e transmissão de informação, através dos sensores instalados nos dispositivos inteligentes, assim como, a atuação/automatização de tarefas, de acordo com o feedback de sensores [1]. Estas caraterísticas dos sistemas IoT, podem ser aplicadas em várias áreas das cidades, tais como [5], [6]:

- Transportes: Controlo e monitorização das vias rodoviárias, no sentido de haver um crescimento da segurança, tanto para condutores como para peões. Atualmente, presenciamos sistemas implementados em transportes públicos, que indicam a sua posição e rota prevista, em tempo real. Deteção de tráfego rodoviário para melhorar o fluxo de veículos nas cidades. Instalação de postos de carregamento elétricos para transportes públicos e veículos pessoais.

- Edifícios: Automatização de serviços energéticos como o aquecimento, iluminação e ventilação. Sistemas melhorados de deteção e extinção de incêndios. Controlo de áreas restritas com recurso a alarmes e CCTV (*closed-circuit television*). Gestão da energia produzida por turbinas e painéis fotovoltaicos.

- Ambiente: Monitorização das emissões de CO_2 e outros gases provenientes de fábricas e de veículos com motores de combustão. Previsão do estado do tempo e consequente deteção antecipada de terramotos e cheias.

- Serviços de utilidade pública: Monitorização e domínio da energia consumida. Controlo da qualidade da água canalizada e deteção de fugas, através de variações de pressão. Assinalar ocorrências de resíduos despejados para rios, proveniente de fábricas.

- Infraestruturas: Monitorização da estabilidade de infraestruturas. Adaptar a sinalização luminosa na via publica, de forma a existir uma poupança significativa em energia. Coordenação entre os sinais de transito de acordo com o estado do tempo.

É evidente o vasto número de utilidades que os sistemas IoT podem ter nos centros urbanos. Soluções, como as descritas, podem significativamente melhorar a gestão e sustentabilidade de uma cidade. A Figura 2 demonstra o resultado de um estudo da IoT Analytics [6], em que são indicadas as aplicações mais relevantes e de grande aderência, em agosto de 2020. Analisando cuidadosamente os valores da Figura 2, verifica-se que, as soluções nas áreas dos transportes e ambiente têm recebido um maior investimento em grande parte das cidades.

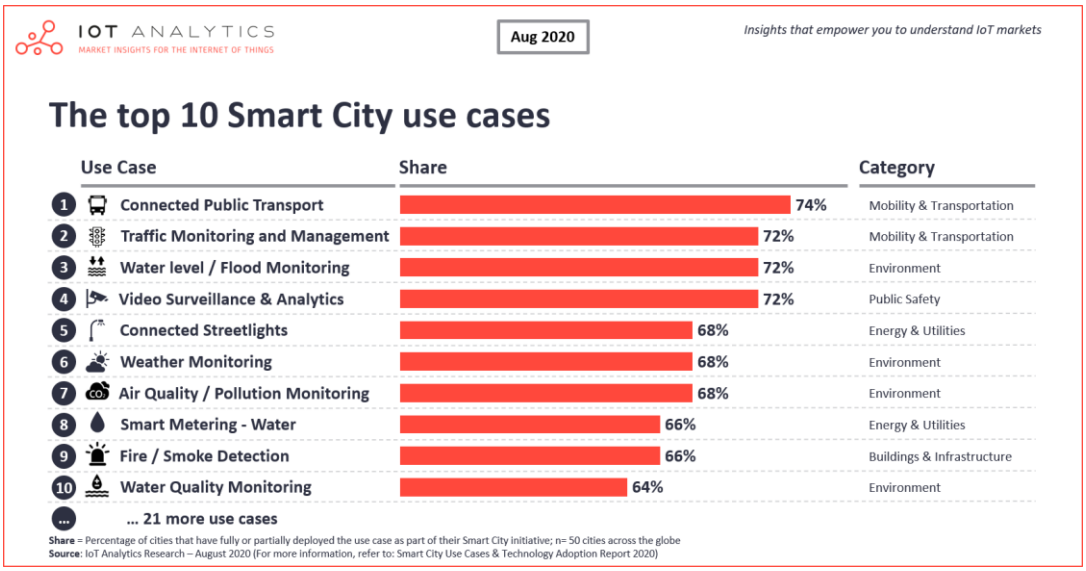


Figura 2 Casos de uso de sistemas IoT em *Smart Cities* [6]

A aposta em novas soluções para a circulação de veículos e gestão de fluxo, provém da questão de congestionamento das faixas de rodagem. O crescimento e desenvolvimento das cidades implica que exista uma circulação, de veículos e peões crescente. Atualmente, existem aplicações que nos conseguem indicar os níveis de transito, acidentes, postos de abastecimento, entre outras funcionalidades. Assim, os condutores podem usufruir de ferramentas, que permitem detetar e indicar o percurso com o menor tempo possível de deslocação, em tempo real. Consequentemente, a eficiência de circulação concede uma celeridade significativa, no desenvolvimento das cidades, o que justifica o forte investimento nestes sistemas [7].

1.1. CONTEXTUALIZAÇÃO

O projeto apresentado surge de uma proposta da empresa Pavnext, com presença na UPTEC (Parque de Ciência e Tecnologia da Universidade do Porto), que consiste em desenvolver um sistema IoT de organização, comunicação e monitorização de dados, para aplicações no contexto das *smart cities*.

Considera-se que o projeto é uma divisão de um sistema, que tem como propósitos:

- Assegurar um conforto às pessoas nas suas interações com veículos em circulação, prevenindo acidentes rodoviários;
- Recolher dados para posterior monitorização e análise, nomeadamente fatores internos e externos ao sistema.
- Gerar energia proveniente das ações dos veículos com o sistema, tornando-o sustentável.

Com estas intenções em mente, existe uma preocupação em conseguir interligar os diferentes segmentos do sistema, visto que, a monitorização e análise dos dados é a agregação e conclusão do mesmo.

1.2. DESCRIÇÃO DO PROBLEMA

O problema provém da necessidade de aquisição, tratamento/filtragem e monitorização/apresentação de dados provenientes de um conjunto de sensores que constituem o sistema NEXT-Road.

NEXT-Road, descreve-se como um pavimento aplicado nas faixas de rodagem, desenhado para suportar as forças de veículos em circulação. O objetivo deste sistema é transformar a energia cinética dos componentes mecânicos em energia elétrica, com o sentido de, posteriormente, servir de alimentação para sinalização luminosa ao redor, ou outro gênero de aplicações. Este sistema promove a redução da velocidade dos veículos durante a passagem no pavimento, sem intervenção do condutor. Com recurso a sensores no seu interior, o sistema consegue produzir um vasto número de dados de interesse, que deu origem ao desenvolvimento deste projeto.

A solução pretendida e lógica, tem a ver com o desenvolvimento de uma placa denominada de controlador principal. Esta, reterá a informação proveniente de sensores e através de comunicação *wireless*, que enviará para um servidor, via Internet. De forma que se proceda à monitorização e análise, é fulcral a utilização de uma base de dados, de onde a informação pode ser consultada. É ambicionado ter uma camada de apresentação da informação ao utilizador, através de uma API (*Application User Interface*). Tendo em consideração as diferentes formas possíveis de apresentação, esta consistirá em diferentes gráficos e cálculos estatísticos de acordo com uma linha temporal.

Pretende-se que a informação flua, desde o controlador principal até à API, ou seja, a informação será recolhida, tratada, enviada, retida e finalmente apresentada ao utilizador. No entanto, comunicação no sentido inverso pode ser útil em alguns casos específicos, como na verificação de erros ou pedir informação a outros controladores.

Com o projeto devidamente descrito, a concretização da Figura 3 intenciona identificar as etapas do desenvolvimento, assim como facilitar a observação das possíveis transferências de dados.

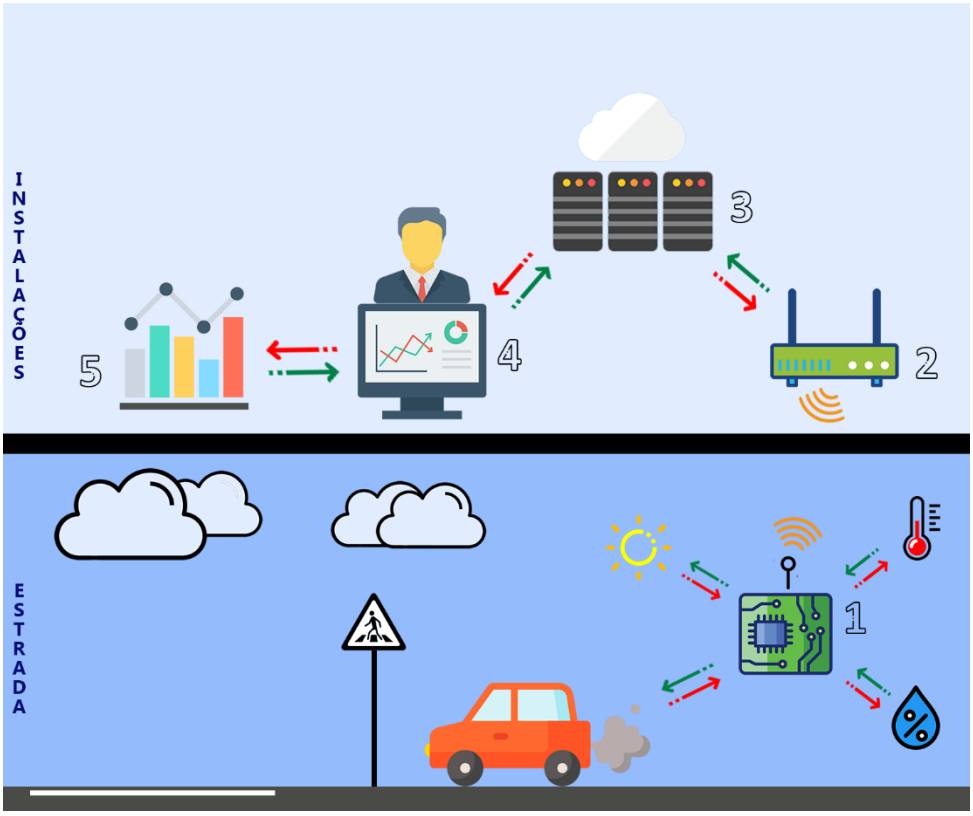


Figura 3 Exemplificação do projeto

A marcação das etapas, tem como objetivo, organizar o projeto durante seu desenvolvimento, assim como, guiar o leitor, numa fase inicial, a compreender o desafio adiante. Considerando uma ordem crescente para a numeração das etapas, estas podem ter as seguintes descrições:

- Etapa 1: Controlador principal – situado perto a uma faixa de rodagem, juntamente com o sistema NEXT-Road, é responsável por receber a informação proveniente de sensores incluídos neste sistema, assim como, de sensores ambientais. Este atua como uma camada de filtragem, organizando e formando pacotes para posterior envio, via *wireless*.

- Etapa 2: *Gateway* – situado no interior de instalações e dentro do alcance de comunicação com o controlador principal, tem como função capturar as mensagens enviadas pela placa no terreno. Descodifica o seu conteúdo e envia-o para um servidor.

- Etapa 3: Servidor – deteta e retém as mensagens que são recebidas da *gateway*. Com o conteúdo descodificado, a informação pode, posteriormente,

ser guardada numa base de dados. Existe a possibilidade de o servidor enviar pacotes de dados à *gateway* para que esta comunique com o controlador principal.

- Etapa 4: Base de dados – ferramenta que permite reter e organizar informação durante um período de tempo definido. Esta pode ser dividida em diversas tabelas de acordo com a organização da informação, que seja pretendida. É a partir destas tabelas que se podem fazer as análises ao conteúdo que é guardado.

- Etapa 5: API/UI – ferramenta/plataforma que usa o conteúdo das tabelas da base de dados para formar uma interface para o utilizador. A informação apresentada deve ser clara e objetiva. Devido à natureza dos dados que serão recolhidos, a sua disposição em gráficos é a mais relevante, permitindo uma análise de acordo com uma linha temporal.

A proposta foi apresentada de uma forma clara e objetiva, onde já era esperado o desenvolvimento de certos elementos do projeto. Desta forma, e pelo conhecimento adquirido, a Figura 4 demonstra, a priori, um esquema dos segmentos práticos do projeto.

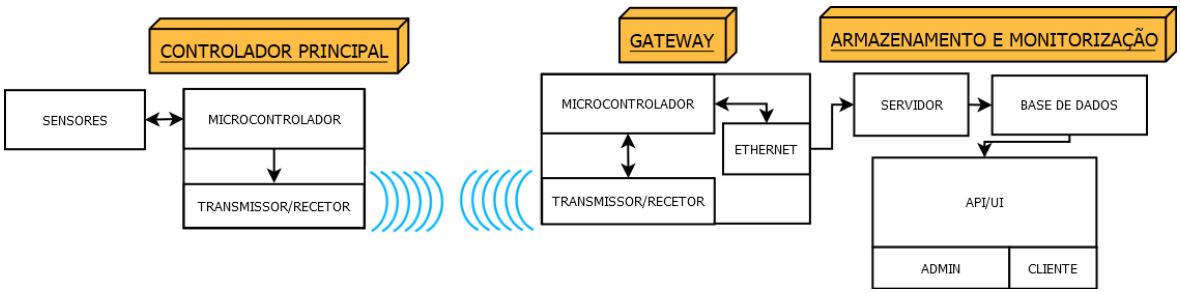


Figura 4 Esquema prático do projeto

O método de apresentação da informação é o último segmento do projeto e como tal, perspetiva-se a possibilidade de análise dos dados armazenados, de um modo simples e intuitivo. A monitorização de dados, pode porventura, ser considerado como um elemento do produto que está a ser desenvolvido pela Pavnext. Assim, opções de gestão do mesmo são necessárias, de forma a manter a funcionalidade do sistema e fornecer suporte, no que à

análise da informação diz respeito. Com esta ideia em mente, a Figura 5 é um exemplo de uma API com uma informação fictícia, em formato gráfico.

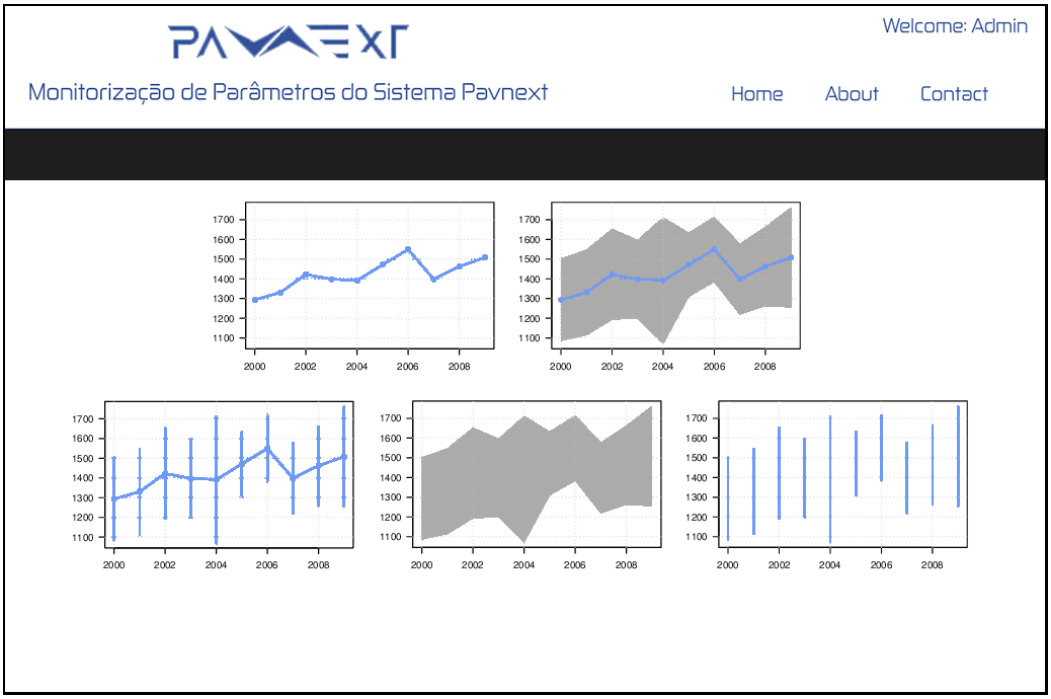


Figura 5 Exemplificação de uma API.

1.3. OBJETIVOS

Reforçando e resumindo o conteúdo descrito no ponto 1.2, o objetivo deste projeto é desenvolver uma solução, que permita desempenhar um conjunto de funções:

- Aquisição de parâmetros ambientais, através de sensores e respetiva filtragem;
- Receção de informação proveniente do sistema NEXT-Road, por protocolos de comunicação, como CAN FD (*Controller Area Network Flexible Data-Rate*) ou RS485.
- Organização da informação por pacotes, para posterior envio, via *wireless*. Esta tem como destino um servidor e consequentemente, é reenviada para uma base de dados.

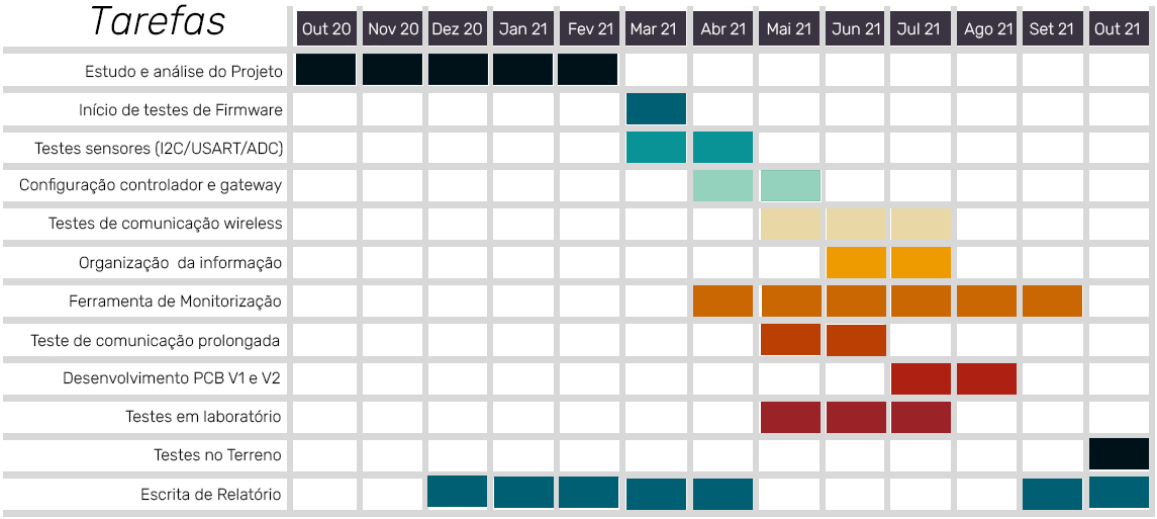
- Divisão da base de dados por tabelas, de acordo com a relação entre estes. Incluir um campo de tempo que represente o momento em que a informação for inserida na base de dados.

- Apresentação de dados com recurso a uma API/UI, que tenha como referência a linha temporal presente na base de dados. Pretende-se que a disposição da informação seja realizada através de gráficos, existindo intersecção de informação entre tabelas da base de dados.

1.4. CALENDARIZAÇÃO

O percurso de desenvolvimento do projeto é apresentado na Tabela 1. Nesta, são exibidas as diferentes tarefas, por ordem cronológica, desde outubro de 2020 a outubro de 2021. Durante a realização do projeto, existiram sobreposições de tarefas, devido à interligação dos elementos do sistema, alterações necessárias na reta final do projeto e a períodos de espera de material necessário para o avanço do mesmo.

Tabela 1 Calendarização do projeto



1.5. ORGANIZAÇÃO DO RELATÓRIO

No capítulo 1 é realizada uma abordagem ao conceito de IoT assim como a implementação e desenvolvimento de sistemas IoT em *smart cities*. No segundo capítulo são estudados sistemas de retenção e apresentação de informação. No capítulo 3 são descritas três tecnologias LPWAN, assim como a sua comparação direta. No capítulo 4 é apresentada

a arquitetura do sistema com a descrição dos seus componentes. No quinto capítulo é descrita a implementação do projeto, tanto da parte de *hardware* como *firmware*. No capítulo 6 são exibidos os testes de maior relevância durante o período de desenvolvimento. O capítulo 7 apresenta as devidas conclusões do projeto desenvolvido.

2. GESTÃO DA INFORMAÇÃO EM *SMART CITIES*

Nos dias de hoje é possível observar um crescimento exponencial de sistemas IoT e o impacto que estes têm na indústria e no mercado. Conseguem alterar a forma de contextualizar, planear e desenvolver soluções viáveis para vários casos. Termos como rede, *cloud, end node,* dados e monitorização, começaram a ser mais frequentes quando estes sistemas são referidos. Permitem que robôs, sensores, máquinas, entre outros, consigam conectar-se à Internet. A consequente evolução de sistemas IoT em grande escala, propulsionou o desenvolvimento de estratégias de comunicação a longas distâncias com o menor consumo energético possível, as LPWAN (*Low Power Wireless Area Networks*) [8].

Os sistemas IoT conseguem ser aplicados a vários setores da sociedade, numa variedade de profissões e vincadamente na indústria (ver Figura 6). Verifica-se que, estes sistemas e as tecnologias inerentes, são proveitosas em aplicações onde estejam incluídos, primordialmente sensores. A integração de dispositivos a uma rede possibilita o armazenamento de dados e parâmetros de interesse, culminando num ponto central onde a informação se aglomera. Este ponto central, que será um servidor de base de dados, pode ser implantado fisicamente ou remotamente (*cloud*). Desta forma, o acesso à informação e consequente apresentação, é simples, devido a um vasto conjunto de ferramentas de monitorização disponíveis, que podem ainda, recorrer a inteligência artificial para otimizar as tomadas de decisão do utilizador [8].

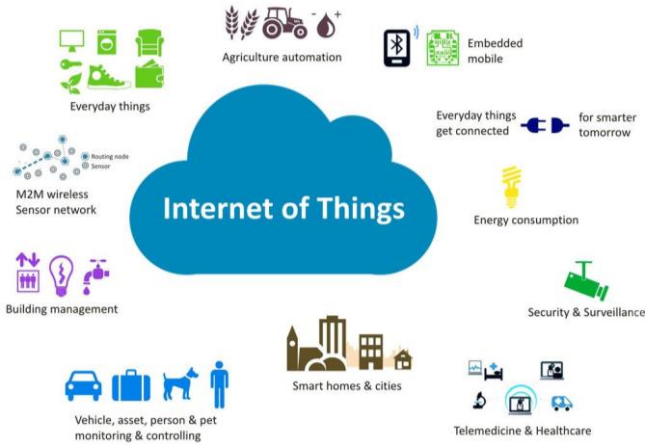


Figura 6 Diferentes aplicações de sistemas IoT [9]

2.1. ESTRUTURA E ORGANIZAÇÃO DE UM SISTEMA IOT EM *SMART CITIES*

Independentemente da aplicação de um sistema IoT, este pode ser composto por um conjunto de camadas, que permitem a transmissão e tratamento da informação. Este conceito descomplica todo o processo, pela qual a informação atravessa. Desta forma, considera-se uma divisão que contempla as camadas: sensorial (*Sensing Layer*), rede (*Network Layer*), intermédia (*Middleware Layer*) e aplicação (*Application Layer*) (ver Figura 7) [10].

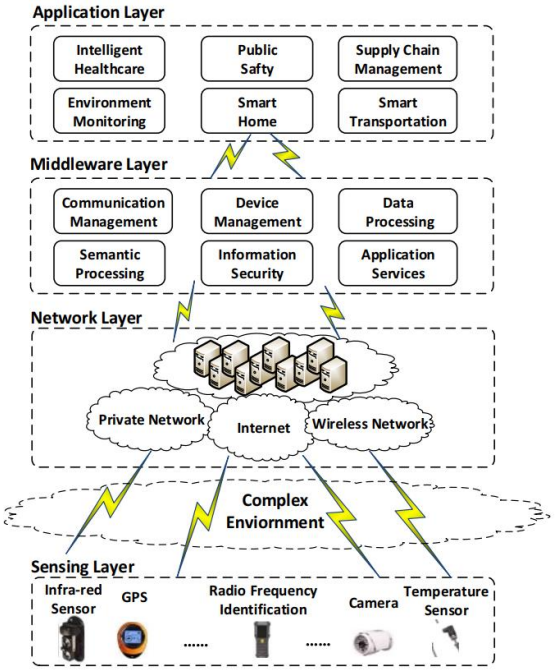


Figura 7 Conceito de Sistema IoT em camadas [10]

A camada sensorial é o conjunto de sensores utilizados no sistema, que convertem informação do meio físico em informação digital. No contexto das *smart cities*, podem ser usados uma vasta gama de sensores, de acordo com o interesse da informação que é obtida. Assim, considerando o exemplo de uma estação meteorológica, são usados sensores ambientais, que fornecem informações de temperatura, humidade, luminosidade, visibilidade, concentração de gases, entre outros. A informação gerada pode, eventualmente, ser sujeita a filtragem, de modo que seja transmitido o essencial [10].

Seguidamente, a camada de rede serve como meio de receção e transmissão da informação às camadas superiores. Assim, podem ser usados os protocolos IPv4/IPv6, 3G/4G/5G, ou uma rede privada, através de *routers* de forma a reencaminhar os pacotes com a informação pretendida [10].

A camada intermédia trata de receber e gerir a informação que vem da camada de rede. Esta gestão pode ter fatores de transformação dos dados e de como estes podem ser interpretados. É gerida a comunicação entre dispositivos e assegurada a retenção da informação através de bases de dados [10].

Por fim, a camada de aplicação tem como objetivo ser o meio de contacto do utilizador com os dados adquiridos. A forma de apresentação varia de acordo com a utilidade dos mesmos e do contexto em que estes se inserem. Assim, esta camada deve ter como referência, o nível de experiência dos utilizadores e a sua finalidade [10].

Em *smart cities* são usadas tecnologias de comunicação de grande alcance, de forma a conseguir cobrir a cidade, ou parte dela, como as LPWAN. Aspetos como área de cobertura, custo e consumo energético, são os alicerces para a implementação de um sistema IoT focado em melhorar a qualidade e a inteligência das cidades.

O grande número de parâmetros a considerar, implica que a rede LPWAN tenha muitos nós distribuídos pela área de cobertura, ou seja, a rede deve ser escalável. Numa zona urbana, os dispositivos instalados podem ser alimentados pela rede elétrica ou por bateria. Devido ao baixo consumo das LPWAN, as baterias têm uma estimativa de duração de 10 anos, pelo que não é necessário efetuar a troca de baterias frequentemente. Num ambiente com uma grande heterogeneidade de dispositivos e plataformas de software, é importante a coexistência e atenuação de interferências no envio da informação.

2.2. BASE DE DADOS

A necessidade de uma base de dados está presente em várias aplicações, onde o ato de registo e retenção da informação é essencial. O conceito de uma base de dados surgiu antes do primeiro computador ser desenvolvido, em que o armazenamento da informação, era predominantemente realizado em papel. Com o aparecimento de sistemas computorizados e devido à sua popularidade, as bases de dados rapidamente foram alvo de desenvolvimentos. Assim, empresas privadas tinham como grande preocupação o aumento da capacidade de armazenamento dos seus sistemas [11].

Uma base de dados consiste num sistema computadorizado que permite a retenção de informação. Um repositório que contém coleções de ficheiros digitais, onde o utilizador consegue efetuar diversas operações sobre estes e no seu conteúdo. Operações de adicionar, remover e apagar, são essenciais para qualquer modelo de base de dados [12].

Em meados dos anos 60, já existiam dois modelos principais de retenção dos dados. Um modelo em rede CODASYL (*Committe on Data System Language*) e um modelo hierárquico IMS (*Information Management System*). O primeiro, tratava-se de um consórcio responsável pela linguagem de programação COBOL (*Common Business Oriented Language*) e por extensões da mesma para as bases de dados. A segunda, desenvolvida pela IBM, tornou-se a primeira base de dados a ser comercializada e ainda se encontra presente em grandes empresas da atualidade [11], [13].

Nos anos 70, Edgar F. Codd publicou o seu estudo e conceito de um modelo da base de dados relacional, que alterou a perceção e o raciocínio sobre as bases de dados. Dez anos depois, este modelo teve um sucesso comercial e era usado globalmente. Nesses mesmos anos a linguagem SQL (*Strutured Query Language*) tornou-se standard para o uso de *queries* [14], [11].

Hoje em dia, a presença de bases de dados, em muitos ramos da indústria, é essencial para o seu funcionamento. Com os rápidos desenvolvimentos na área dos sistemas IoT, a necessidade de armazenar a informação para posterior análise, monitorização, previsão e/ou aplicação, tem um relevo enorme. Estes sistemas conseguem receber um grande volume de dados, provenientes de sensores e dispositivos que a este estejam interligados. Existem amplas aplicações destes sistemas, por exemplo, em ambiente industrial, hospitalar e em cidades inteligentes. O aumento do uso de soluções IoT significa, também, um aumento de

aplicações em tempo real, onde a gestão da informação e o respetivo armazenamento são a grande dificuldade [15].

Pretende-se que a informação capturada consiga ser apresentada em tempo real, ou com uma baixa latência para que a sua monitorização seja o mais eficiente possível. Assim, no âmbito de uma aplicação IoT e deste projeto, a decisão de qual sistema de retenção de dados deve ser usado, tem influência no seu comportamento. Com um ponto de observação generalizado, podem ser considerados dois tipos principais de bases de dados (Figura 8), relacionais (SQL) e não relacionais (NoSQL) [15].

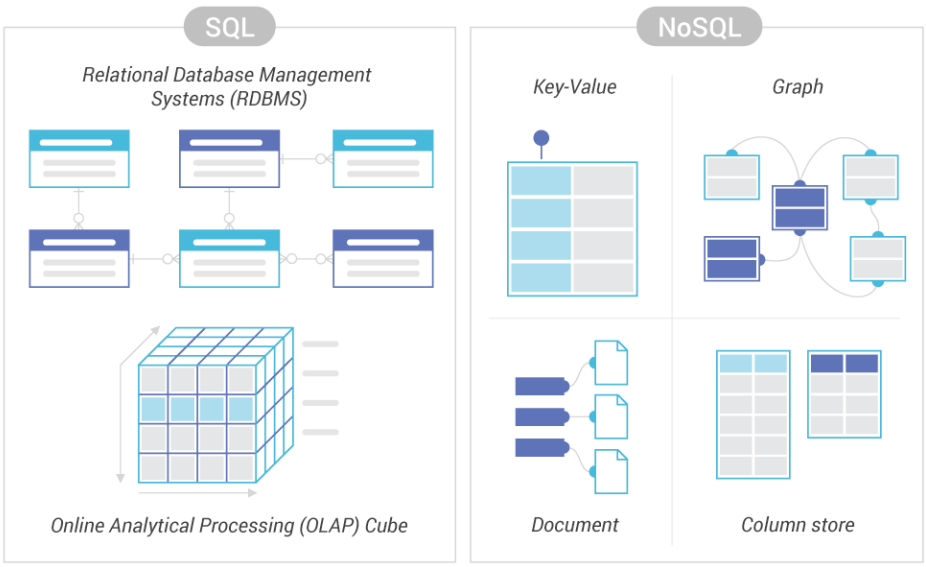


Figura 8 Diferença entre bases de dados SQL e NoSQL [16]

Um tradicional DBMS (*Database Management System*) usa a linguagem SQL e segue o modelo relacional onde os dados são armazenados em linhas consecutivas em tabelas. Estas tabelas podem ser interligadas, de acordo com a relação que existe entre os dados. No sentido inverso, as bases de dados NoSQL seguem um modelo não relacional. Suportam vários formas de armazenamento da informação, sem um esquema restrito [16]–[18]:

- *Key-Value* – Este tipo de armazenamento é o mais simples e consiste num nome (*key*) e num valor (*value*);

- *Column-Oriented* – A organização da informação consta numa tabela que aumenta horizontalmente, consoante a introdução de nova informação. A pesquisa limita-se às colunas em causa, ao contrário de uma base de dados relacional em que a pesquisa é realizada linha a linha. Assim, evita-se o consumo de recursos ao incluir informação não desejada. Beneficiam ainda de uma compressão dos dados mais eficiente, melhorando os tempos de pesquisa;

- *Document* – A informação é armazenada em ficheiros JSON (*JavaScript Object Notation*), BSON (*Binary JavaScript Object Notation*) ou XML (*Extensible Markup Language*). Estes documentos podem ser agrupados e os seus elementos indexados, para aumentar a rapidez de consulta. É um tipo de base de dados flexível em que é possível alterar as suas estruturas de documentos assim que necessário, acelerando o seu desenvolvimento;

- *Graph* – Neste tipo são usados nós para armazenar entidades de dados e são realizadas relações através de bordas, entre várias entidades. As bordas descrevem o tipo de relação entre as entidades (nós). A pesquisa é focada nas relações que existem entre entidades. Numa base de dados relacional, as ligações são implícitas, onde é usada a informação para expressar essas ligações e onde são consultadas várias tabelas. É um tipo de base de dados usado em redes sociais, deteção de fraude, mecanismos de recomendação, entre outros;

Sistemas de base de dados NoSQL têm aumentado em popularidade nestes últimos anos. Destinam-se, geralmente para casos de uso mais específicos, no entanto conseguem fornecer uma elevada escalabilidade horizontal e podem ser facilmente distribuídas. A sua popularidade é justificada pelo fácil acesso, arquitetura distribuída e pela gestão competente de dados não estruturados (em aplicações IoT, correspondem geralmente, aos dados obtidos pelos sensores) [17].

No sentido de escolher o melhor sistema para aplicações IoT, a Tabela 2 demonstra as diferenças entre SQL e NoSQL, em caraterísticas predominantes como a escalabilidade, a facilidade de recuperar os dados e o nível de desenvolvimento (maturidade).

Tabela 2 Diferenças DBMS SQL vs. NoSQL [17]

	SQL	**NoSQL**
Escalabilidade	Vertical: capacidade de aumentar o desempenho num nó (servidor)	Horizontal: capacidade de partilhar a carga por vários nós (servidores)
Recuperação de dados	Através de JOINS que combinam os resultados de várias tabelas. Pesquisa mais morosa	São criados objetos que contêm toda a informação. Pesquisa mais eficiente
Maturidade do sistema	Sistema mais desenvolvido em questões como: segurança, privacidade, integridade dos dados	Sistema mais recente com possíveis falhas e falta de caraterísticas de segurança dos dados

Segundo um estudo da *ScaleGrid*, realizado no evento *DeveloperWeek* de 2019, é notório o aumento do uso de bases de dados NoSQL por parte das empresas para colmatar a falta de funcionalidades e características de uma base de dados SQL. Conclui-se que, 44,5% das empresas questionadas usam vários sistemas de base de dados. Desta parcela, cerca de 75,6% utiliza a combinação SQL com NoSQL (Figura 9) [19].

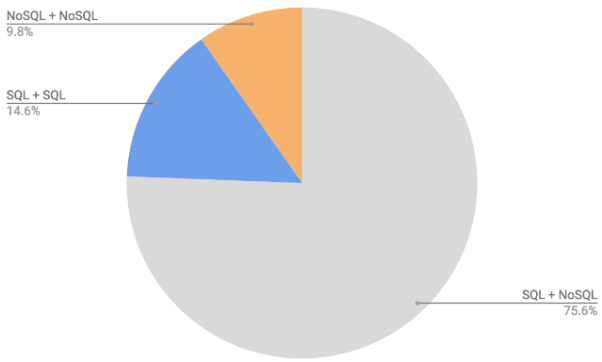


Figura 9 Combinações de uso de várias bases de dados [19]

2.3. *TIME SERIES DATABASE* (TSDB)

Nestes últimos anos, têm surgido grandes desenvolvimentos para ser possível receber dados de séries temporais, de maneira eficiente. Estes, correspondem a uma sequência discreta de valores que são recebidos a uma frequência constante. Os dados são assim organizados de acordo com uma linha temporal, sendo possível fazer previsões de acordo com os dados do passado. Pretende-se que a informação recebida, esteja dependente de um

tempo associado. Por exemplo, medições dos diversos sensores de uma estação meteorológica, podem ser considerados dados de séries temporais [20].

De uma forma sucinta, séries temporais são dados que coletivamente representam como um sistema/processo ou comportamento varia com o tempo. Estes podem ser utilizados em vários contextos como [20]:

- Análises de séries temporais – Monitorização de uma variável ao longo do tempo;
- Análises de regressão – Monitorização em alterações de variáveis e como estas têm efeito noutras variáveis durante o mesmo período;
- Previsão de séries temporais – Monitorização de variáveis em que são analisadas alterações recentes ou padrão previamente detetados, para prever um evento no futuro;

Uma TSBD (*Time Series Database*) é um tipo de base de dados que é desenvolvida, especificamente, para conseguir agregar dados de séries temporais ou eventos com um tempo associado. São otimizadas para conseguir gerir, de maneira eficiente, a quantidade de métricas que são introduzidas num curto intervalo de tempo. Assim, este tipo de base de dados consegue estar presente em várias empresas que estejam dependentes de séries temporais [20]. Costumam ser concebidas para aplicações de monitorização, no entanto, num sistema abrangente, em que sejam consideradas variáveis que não são séries temporais, pode não ser a melhor opção [15].

O armazenamento da informação em TSDB (*Time Series Database*) pode ser um problema, devido à sua acumulação com o passar do tempo. Desta forma, o foco para estas bases de dados, está em aumentar a eficiência de armazenamento da informação. Outra desvantagem é a falta de apoio semântico, visto que, a informação é condensada e limitada ao necessário. Assim, só quem projeta a base de dados consegue interpretar os dados que a constituem, limitando a eficiência e processamento quando se pretende estabelecer relações entre a informação [15].

A utilização de uma TSDB pode ser justificada de acordo com as suas funcionalidades gerais [20]:

- A informação é armazenada em blocos (*chunks*), no mesmo espaço físico, que pertencem ao mesmo intervalo de tempo, permitindo um rápido acesso e uma análise mais eficiente;

- Rapidez de escrita e leitura da informação mesmo em momentos limite, para ser possível receber métricas a cada segundo, ou até a intervalos inferiores;

- É realizada uma compressão nos dados devido ao grande volume de métricas;

- Estão incluídas funcionalidades que permitem focar no tempo, aumentando a escalabilidade e desempenho (taxas de inserção, consultas rápidas e compressão).

- Possuem funções e operações para definir políticas de retenção, realizar *queries* continuas ou *queries* num intervalo de tempo;

O uso de TSDB tem sofrido um grande crescimento em termos de popularidade, devido ao aparecimento de sistemas IoT (ver Figura 10). Estes são desenvolvidos para recolher e partilhar dados, provenientes de sensores, entre dispositivos. Por norma, estes sistemas geram um grande volume de informação, pelo que uma TSDB consegue geri-los de forma eficiente.

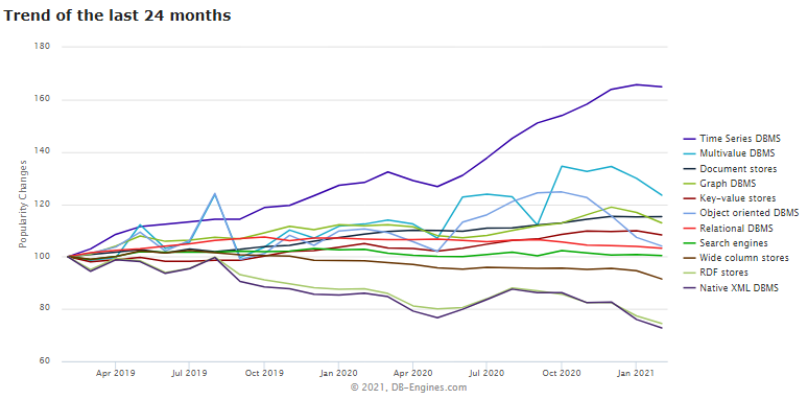


Figura 10 Popularidade de diferentes tipos de bases de dados em 2 anos [21]

No entanto, o impacto de bases de dados relacionais, ainda nos dias de hoje é inegável. Continuam a ser as mais populares devido à organização estruturada da informação e das relações entre os dados em diferentes tabelas (ver Figura 11).

Rank			DBMS	Database Model
Feb 2021	Jan 2021	Feb 2020		
1.	1.	1.	Oracle	Relational, Multi-model
2.	2.	2.	MySQL	Relational, Multi-model
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model
4.	4.	4.	PostgreSQL	Relational, Multi-model
5.	5.	5.	MongoDB	Document, Multi-model
6.	6.	6.	IBM Db2	Relational, Multi-model
7.	7.	↑ 8.	Redis	Key-value, Multi-model
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model
9.	9.	↑ 10.	SQLite	Relational
10.	10.	↑ 11.	Cassandra	Wide column

Figura 11 Ranking de bases de dados a fevereiro de 2021 [22]

Representando a grande parte das bases de dados em uso, as RDBMS (*Relational Database Management System*) informam o estado das entidades sem que haja histórico associado. Já com as TSDB, é possível observar as variações dos estados com o tempo. Assim, a Tabela 3, apresenta alguns contextos para o uso destes dois tipos de bases de dados [23].

Tabela 3 Casos de usos possíveis para RDBMS e TSBD [23]

Base de dados relacional (RDBMS)	**Base de dados de séries temporais (TSBD)**
Sistemas de gestão de conteúdo	Sistemas de monitorização ao longo do tempo
Armazenamento de dados transacionais	Dados analíticos / de relatório
Armazenamento de dados que necessitam de ser atualizados ao longo do tempo	Anexar alterações apenas (escrever uma vez, ler várias)
Armazenamento de longo prazo	Conjunto de dados de curta duração

2.3.1. INFLUXDB

InfluxDB é uma base de dados *open-source* da empresa InfluxData, desenvolvida para receber e organizar dados de séries temporais. Criada com linguagem de programação Go, criada em 2007 pela Google, sendo semanticamente semelhante à linguagem C com funcionalidades adicionais [20]. Para a execução de *queries* e escrita, foi criada uma linguagem de *script*, Flux, semelhante a SQL, que suporta informação de bases de dados SQL e ficheiros CSV [24].

Devido ao aumento de dados gerados pelos sistemas recentes, esta base de dados foi concebida para suportar grandes quantidades de dados, em tempo real. Assim, torna-se ideal

para aplicações de monitorização e análise de métricas que sejam introduzidas, de forma rápida e eficiente. Demarca-se de outras TSDB pela facilidade e simplicidade de implementação num sistema, não descartando a qualidade do seu funcionamento. Comparando com uma DBMS relacional, o seu funcionamento é claramente diferente, no entanto conceitos semelhantes têm designação diferentes [20].

Uma base de dados em InfluxDB é composta por *measurements* (semelhante ao conceito de tabelas em SQL). Cada *measurement* possui uma coluna por defeito, com o registo do tempo em que a informação é introduzida na base de dados (*timestamp*). Para além da coluna *time*, a informação é inserida como *fields*(atributos) ou *tags*, que ainda se dividem em *field keys*/*field values* e em *tag keys*/*tag values*, respetivamente. A Figura 12 demonstra um exemplo da constituição de um *measurement*, com as respetivas *fields* [20], [25].

```
name: census
-----------------
time                  butterflies honeybees   location    scientist
2015-08-18T00:00:00Z  1           30          1           perpetua
```

Figura 12 Exemplo de um *measurement* [25]

Com a figura anterior como referência, o *measurement* designa-se de “census” e possui dois *fields sets* e dois *tag sets*, que são pares *key-value*. Identificam-se como *fields keys*, *butterflies* e *honeybees*, com os *fields values* 1 e 30, respetivamente. As *tags keys* são, *location* e *scientist*, com os *tag values* de 1 e “perpetua”, respetivamente [20], [25].

As *tags* são retidas como *strings* e indexadas de forma a permitir uma rápida análise no momento em que se executam *queries*. Assim, conforme a aplicação, não é aconcelhado criar *fields* em informação onde se pretende executar uma variedade de *queries*. No entanto, a informação não pode ser inserida no *measurement* sem possuir *fields*. À medida que se introduzem os dados, as colunas podem crescer verticalmente ou horizontalmente [25],

- Verticalmente, sempre que existe um novo *field value* ou *tag value* para um *field key* ou *tag key* existente;
- Horizontalmente, sempre que são introduzidos novos *field keys* ou *tag keys*.

Trata-se de uma base de dados sem esquema definido, podendo assim, ser adicionados novos *measurements*, *fields* e *tags*, sem qualquer impedimento, o que torna esta TSDB flexível. Possui ainda, um conjunto de configurações como [26],

- Autenticação – Suporta vários utilizadores com o respetivo *username* e *password*. Consequentemente, cada utilizador tem a possibilidade de criar as bases de dados que desejar;

- Formato da informação – A informação pode ser retida em formato json, csv ou coluna;

- Precisão – Relativo ao *timestamp*, este pode ter o formato rfc3339, horas, minutos, segundo, milissegundos, microssegundos e nanossegundos;

- Política de retenção – Período de tempo em que a informação é retida na base de dados, aplicada todos os *measurements* dentro da base de dados.

Recentemente, foi lançada a versão InfluxDB 2.0, que inclui uma API de monitorização. Esta permite o uso de *dashboards*, onde podem ser criados gráficos, cálculos, previsões, localização, histogramas, alertas, com recurso à informação que é inserida na base de dados (ver Figura 13) [27].

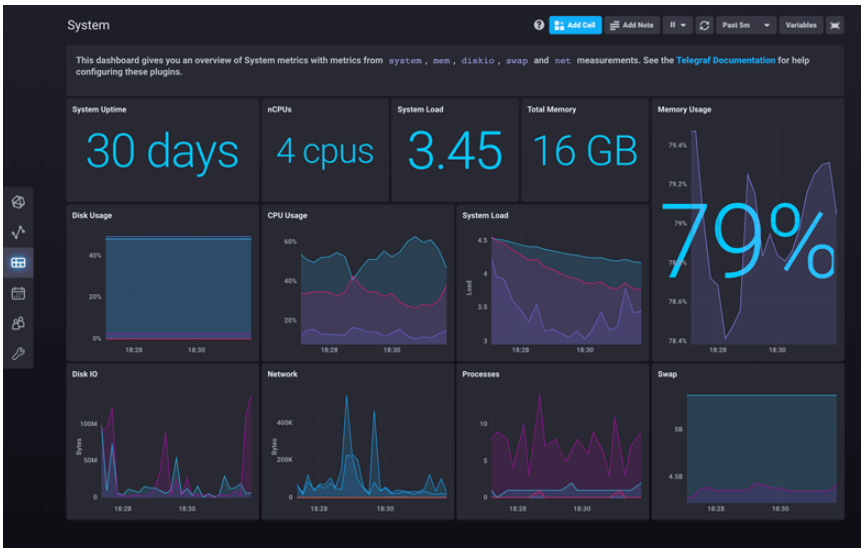


Figura 13 Exemplo de dashboard InfluxDB [27]

2.3.2. TIMESCALE

Timescale é uma extensão *open-source* para PostgreSQL que permite adicionar caraterísticas de uma base de dados de séries temporais numa base de dados SQL comum. Isto permite que ter uma rapidez de 10 a 100 vezes nas *queries* em comparação com PostgreSQL, InfluxDB (NoSQL) e MongoDB (NoSQL). Tem uma interface inteiramente SQL o que permite usar esta linguagem de base de dados para executar *queries* (JOINS) entre os dois tipos de informação (relacional e de séries temporais). Como é desenvolvido a partir das funcionalidades do PostgreSQL, o Timescale consegue conectar a qualquer cliente ou ferramenta que também suporte PostgreSQL [28].

O PostgreSQL perde performance de acordo com o volume de dados que são inseridos, isto deve-se ao facto de os índices das tabelas não caberem mais em memória (ver Figura 14). Sempre que existe um dado a entrar na base de dados os índices necessitam de ser atualizados. Isto tem implicações na utilização do disco que armazena os dados. Aumentar a capacidade de armazenamento não resolve o problema, visto que com o acumular das linhas, a performance reduz novamente. O Timescale contorna este problema através de *time-space partitioning*, isto é, as escritas dentro de um intervalo de tempo são só incluídas em tabelas que ficam em memória, tornando as atualizações de índices secundários mais rápida [29].

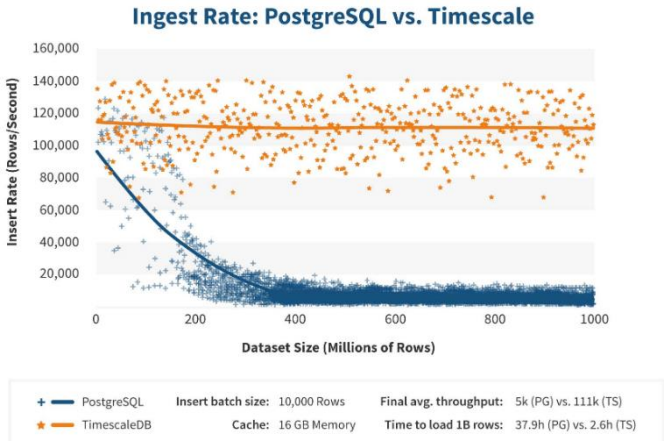


Figura 14 Frequência de receção de acordo com o volume de dados [29]

Conclui-se que, o Timescale corre um mil milhão de linhas, num décimo do tempo que o PostgreSQL o faz. Indica ainda que a taxa de transferência é vinte vezes superior ao PostgreSQL para grandes volumes de dados [29].

Em termos de arquitetura o Timescale é composto por uma *hypertable* e por vários *chunks:*

- *Hypertables* – Primeira instância que recebe as métricas, apresenta-se como uma única tabela em que é possível executar *queries* com SQL padrão. Esta tabela contém no mínimo duas colunas, uma que identifica o tempo associado e o respetivo valor. Todas as intervenções do utilizador são com a *hypertable* (alteração de tabelas, criação de índices, a seleção de dados, etc.) [30];

- *Chunks* - A *hypertable* é internamente dividida em vários *chunks* (ver Figura 15). Estes correspondem a informação recebida num intervalo de tempo e são implementadas como uma base de dados comum. Todos os *chunks* têm o mesmo tamanho (via manual ou automática) garantindo que todas as *b-trees* de índices possam estar em memória [30].

De acordo com as políticas de retenção, os dados são apagados quando chegam a um limite de tempo. Ao fazer uma divisão por *chunks*, significa que a informação é apagada em blocos e não por linhas de uma tabela. Evitando *chunks* de grandes dimensões, é possível tornar esta operação mais eficiente.

Figura 15 *Hypertable* e *Chunks* [30]

A versão atual do Timescale suporta particionamento em implementações single-node. O facto de ser uma única máquina torna o dimensionamento da base de dados um problema, pois é necessário considerar a relação custo/desempenho da mesma. Com o passar do tempo a ingestão dos dados é tão grande que não cabe em memória. Uma vez que isto se verifica, atualizar uma parte da *b-tree* pode envolver troca de dados com o disco. Base de dados, como o PostgreSQL, possuem uma estrutura em *b-tree* para cada índice de tabela para que a sua pesquisa seja o mais eficiente. No entanto com o aumento das colunas estes índices aumentam [30].

Como o Timescale divide a informação em *chunks*, estes são armazenados como se uma base de dados independente em que os índices são referentes a esses *chunks* e não a uma tabela inteira. Logo o correto dimensionamento dos *chunks* permite ter os índices recentes em memória, evitando que sejam imediatamente escritos em disco [30].

Vantagens do uso do Timescale em relação às bases de dados NoSQL:

- Uso da linguagem SQL – As bases de dados NoSQL são desenvolvidas com recurso a linguagens já existentes ou usam a sua própria linguagem. Este implica que o desenvolvedor terá de se dedicar a aprender uma linguagem de *queries* diferente. Considerando até linguagens mais semelhantes ao SQL, estas podem apresentar problemas de compatibilidade. O Timescale usa o SQL standard para as *queries* em dados de séries temporais [31];

- Simplicidade e rapidez – O Timescale permite que o utilizador tenha o melhor dos dois tipos de base de dados. Tanto os dados de séries temporais como os dados relacionais estão na mesma base de dados onde é possível executar *queries (JOINS)* sobre esta informação. Esta caraterística permite descartar soluções em que são usadas duas bases de dados distintas. As *queries* são otimizadas para que os resultados sejam apresentados o mais rápido possível, independentemente da sua complexidade. Dependendo da *query*, grande maioria das bases de dados NoSQL necessitam de fazer uma pesquisa pela tabela inteira o que, com o acumular dos dados torna-se lento [31];

- Ferramentas PostgreSQL e de terceiros – Qualquer ferramenta que suporta SQL pode ser integrado e utilizado no Timescale. Como se trata de uma extensão para o PostgreSQL, suporta naturalmente os diversos tipos de dados e índices (*B-tree*, *hash*, *range*, BRIN, GiST, GIN) [31].

2.3.3. PROMETHEUS

Prometheus é um sistema *open-source* de monitorização que recebe métricas de *endpoints* HTTP (*Hypertext Transfer Protocol*). Foi desenvolvido pela empresa *SoundCloud* e lançado publicamente em 2015. O propósito deste sistema era lidar com dados de séries temporais para uma monitorização de serviços em *cloud*, *conteiners* ou de aplicações. É uma ferramenta que se foca no presente e considera que os dados a apresentar são de atividade

recente. Desta forma, resultados de *queries* e de alertas podem ter apenas horas de idade. Segundo um estudo realizado da empresa Facebook, 85% das *queries* tinham menos de 26 horas de idade. Assim, apesar de possuir uma linguagem de *query* avançada (PromQL), os tempos de retenção da informação são baixos, cerca de 15 dias localmente. Para casos em que a retenção da informação seja importante, é aconselhado enviar a informação para um serviço remoto. No entanto é possível armazenar os dados em bancos de armazenamento [32], [33] .

Desenvolvido em linguagem Go, possui licença Apache 2.0 e pertence à *Cloud Native Computing Foundation* (CNCF). Fundação Linux responsável pela agregação de sistemas, desenvolvedores, consumidores e vendedores numa comunidade empresarial [32].

A arquitetura do sistema Prometheus é composto pelos seguintes componentes (Figura 5), pelo que, a maioria é de implementação opcional [32], [34]:

- *Prometheus Server* – Servidor que retém a informação na base de dados.
- *Libraries* – Bibliotecas que implementadas no cliente, permitem a leitura das métricas via HTTP.
- *Push Gateway* – Componente que atua como cache para que métricas de pouca duração possam ser inseridas no servidor.
- *Exporters* – Bibliotecas e servidores que permitem exportar métricas de sistemas de terceiros (bases de dados, drivers, sistemas de mensagem, armazenamento, HTTP, API, sistemas de monitorização, etc.).
- *Alertmanager* – Gere os alertas enviados pelos clientes como o servidor Prometheus. Agrupa e reencaminha os alertas por email, PagerDuty ou OpsGenie.

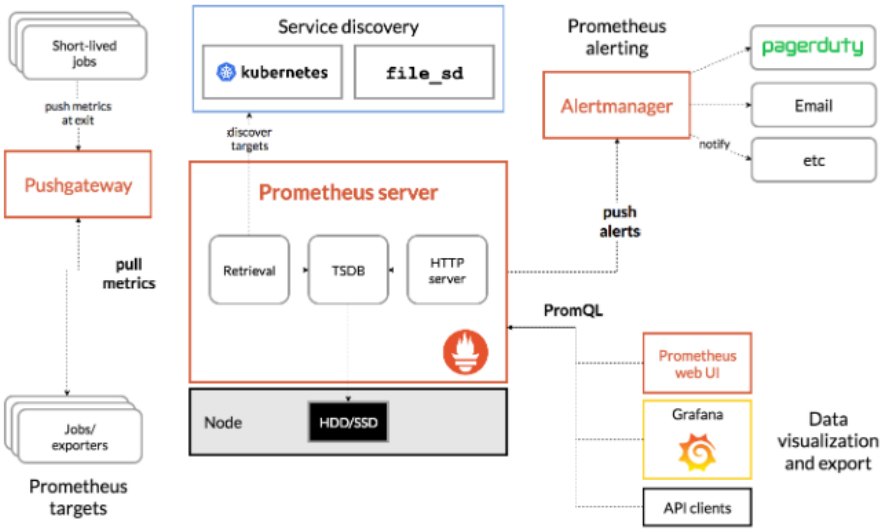


Figura 16 Arquitetura Prometheus [32]

O Prometheus possui no servidor uma barra de pesquisa destinada a *queries*. Os resultados tanto podem ser apresentados em tabela ou por um gráfico com o decorrer do tempo. No entanto, este tipo de apresentação é básico e muito limitado. Assim, a ferramenta de visualização de dados usualmente utilizada em conjunto com o Prometheus é a Grafana. Esta oferece suporte de desenvolvimento ao projeto Prometheus de forma a garantir compatibilidade entre as duas ferramentas [32], [35].

Uma das características desta ferramenta é a sua confiabilidade, isso verifica-se no número de empresas que a utilizam e nos variados casos de uso. No entanto, é desaconselhado o uso do Prometheus para casos de uso em que a necessidade da exatidão dos dados seja importante [32].

Prometheus reconhece 4 tipos gerais de métricas:

- *Gauge* - métricas que representam um único valor numérico que pode variar (aumentam ou diminuem). No sentido do projeto, este tipo de dados deve ser usado em parâmetros como a temperatura, humidade, intensidade luminosa, peso de veículo, energia gerada, o nível da bateria, entre outros [36].

- *Counter* - métricas que representam um valor que só aumenta de acordo com os eventos. A contagem pode ser reiniciada a qualquer momento, iniciando no valor 0. Este tipo de dados pode ser usado na contagem do número de carros que passam pelo sistema [36].

- *Histogram* - engloba uma amostra dos dados para apresentação, de acordo com a sua frequência, com intervalos configuráveis. Cada amostra é contada e posta em *buckets.* Isto permite agrupar diversas métricas (um *bucket* por métrica) e apresentar a sua soma de todos os valores. Em termos gráficos, este tipo de dados costuma ser apresentado num gráfico de barras [36].

- *Summary* - de forma semelhante ao *histogram*, o *summary* apresenta amostras dos dados recebidos. No entanto este tipo de dados permite a aplicação de funções/transformações matemáticas como: contagens, somas, médias, medianas, percentagens, desvios e taxas de variações. É ainda possível agregar dados em casos gráficos em que se esteja a comprar mais que uma métrica em simultâneo. Útil para prever alterações gerais das métricas [36].

Para cada um destes tipos de dados o Prometheus fornece bibliotecas nas linguagens Go, Java, Python e Ruby, para serem implementadas do lado do cliente.

De acordo com a generalidade das bases de dados, o Prometheus reconhece uma notação que contenha um identificador da métrica e um valor associado (ver Figura 17). No entanto pode receber instâncias (identificam o *host*/aplicação que fornece a informação) e *jobs* (conjunto de instâncias com o mesmo propósito).

Listing 2.2: Example time series

```
total_website_visits{site="MegaApp", location="NJ", instance="
webserver",job="web"}
```

Figura 17 Exemplo notação Prometheus [[36]]

O sistema é configurado através de *flags* em terminal e de um ficheiro de configuração, prometheus.yml. As *flags* permitem alterar parâmetros como locais de armazenamento, quantidade de dados a armazenar, etc. O ficheiro de configuração permite gerir *jobs* e respetivas instâncias, assim como ficheiros de regras adicionais.

O prometheus.yml é composto por um primeiro bloco, *global* que contém as configurações gerais para o controlo do servidor Prometheus, como a frequência de aquisição dos dados. O segundo bloco, a*lerting* é constituído pelas configurações de alerta

do Prometheus. O terceiro bloco, *rules_files* corresponde a uma lista ficheiros que contêm regras a aplicar sobre as métricas. E por último, *scrape_config*, que define os locais onde o Prometheus obterá as métricas.

2.3.4. GRAPHITE

Graphite é uma ferramenta de monitorização leve que permite ser implementado em hardwares mais baratos ou em serviços de *cloud*. Desenvolvido em 2006 por Chris Davis como um projeto secundário da empresa *Orbitz*. Após dois anos o projeto tornou-se *open-source* com licença Apache 2.0. Hoje em dia esta ferramenta é usada por diversas empresas como *Github, Booking.com, Reddit*, entre outras [37].

Desenvolvido em *Python* 2.0, efetua operações de I/O numa variedade de ficheiros para que o processamento dos dados seja o mais rápido possível. Possui mecanismos de cache opcionais, que conseguem reter alguma informação caso os discos locais não consigam aguentar com o grande volume de informação. Para tal, requer um bom sistema RAID (*Redundant Array of Inexpensive Drives*) ou um conjunto de SSDs (*Solid-state Drive*) [37], [38].

Este sistema tem o intuito de armazenar dados de séries temporais e apresentá-los de forma gráfica de acordo com o que é pretendido. Graphite não tem a função de reter os dados que recebe, no entanto demonstra-se útil para que as métricas sejam inseridas numa base de dados de séries temporais. Em termos de arquitetura este é composto por três componentes:

- *Carbon*: Serviço de alta performance que ouve e recebe métricas;
- *Whisper*: Simples biblioteca de base de dados para guardar a informação;
- *Graphite-web*: Interface com o utilizador *Danjo* que faz render (biblioteca Cairo) de gráficos e painéis.

De uma forma simples as aplicações enviam a informação para o *Carbon* do Graphite que as retêm na base de dados. À medida que a informação entra no Graphite vai se tornando disponível para apresentação gráfica (Figura 18).

Figura 18 Arquitetura do Graphite [39]

Existem três modos principais para enviar dados para o Graphite, através de *PlainText*, *Pickle* ou AMQP (*Advanced Message Queuing Protocol*). Independentemente do modo de entrada dos dados o *Carbon* atuará e controlará a informação, posteriormente a interface obterá a informação da cache ou da leitura direta dos discos [40].

O método *plaintext* é o mais simples de ser implementado, no entanto é indicado para aplicações singulares ou para testes de funcionamento. Os dados devem ser enviados num formato específico, *<metric path> <metric value> <metric timestamp>*, para que o carbon consiga traduzir a informação de forma que a *whisper* entenda. É possivel usar a ferramenta netcat para criar *sockets* que permitam o envio dos dados para o Carbon (*plaintext* usa por norma o porto 2003) [40].

Para grandes quantidades de dados a opção pickle demonstra ser a mais eficiente pois consegue atualizar vários campos da base de dados em simultâneo. A informação é enviada no formato de lista, [(*path*, (*timestamp*, *value*)), ...], para o *Carbon* através de um *socket*. A informação é encapsulada com um simples *header* mais a lista (*payload*) A lista ao ser preenchida não deve ultrapassar um determinado tamanho [40].

Para o uso de AMQP é necessário definir como *True* a configuração AMQP_METRIC_NAME_IN_BODY no ficheiro *carbon.conf*. Apresentará um aspeto semelhante ao usado no *plaintext* [40].

De uma forma geral, o Graphite reconhece a informação recebida da seguinte forma:

```
metric_path value timestamp\n
```

- *metric_path* - destina-se a identificar o campo que recebe uma nova leitura;
- *value* - corresponde ao valor lido, associado a um determinado *metric_path;*
- *timestamp* - número em segundos de acordo com o sistema temporal Unix.

2.4. GRAFANA

Com o desenvolvimento das bases de dados de séries temporais, surgiu a necessidade de aproveitar a informação obtida e criar ferramentas de suporte que organizem os dados de forma intuitiva e de fácil leitura. Assim, ferramentas de monitorização e *plugins* foram concebidos para possibilitar a visualização da informação, em fontes de podem reter dados, mais especificamente, bases de dados.

A Grafana é um projeto iniciado por Torkel Ödegaard em 2014 e rapidamente se tornou um dos maiores projetos *open-source* da plataforma GitHub. Consiste numa plataforma para executar *queries*, visualizar, alertar e analisar métricas por meio de gráficos. Tem um modelo de entrada de dados através de *plugins* e é compatível com diversas bases de dados de séries temporais (Graphite, Prometheus, ElasticSearch, OpenTSDB e InfluxDB) e bases de dados SQL (MySQL e PostgreSQL). Conta com um suporte adicional a serviços de *cloud* como a Google Stackdriver, Amazon CloudWatch e Microsoft Azure. Para além dos *plugins* de bases de dados, a plataforma permite incluir aplicações e ferramentas adicionais de forma a enriquecer a variedade de painéis e gráficos possíveis. [35], [41].

A plataforma organiza-se a partir de painéis altamente dinâmicos e customizáveis onde o utilizador decide que gráficos se adequam às métricas que pretende monitorizar (ver Figura 19). Os painéis podem ser constituídos por gráficos de barras, linhas, em escadas, área, pontos, entre outros. O número de painéis pode ser variado, estes reutilizam-se e existem *templates* da comunidade que podem ser importados [35], [41].

Figura 19 Exemplo *dashboard* Grafana [35]

A entrada de métricas pode ter diversas fontes, estas conseguem ser apresentadas e comparadas em tempo real. Parâmetros temporais, como o tempo de atualização dos dados, os intervalos de tempo para visualização, permitem ajustar os gráficos e reorganizar a informação de acordo com o desejo do utilizador. Como a informação consiste maioritariamente na análise e apresentação de valores numéricos, é possível definir alertas sobre as métricas recebidas para que haja notificações de valores indesejados [35], [41].

Com estas funcionalidades e um design agradável e de simples análise, a plataforma começou a despertar o interesse de empresas que a utilizam para analisar métricas. A sua utilização pode ser interna ou vendido como um serviço a clientes. Para tal, é possível incorporar logos de empresas e fazer uma gestão dos utilizadores, os seus privilégios e os gráficos a que tem acesso [35], [41].

Verifica-se, na última década, um aumento do uso das bases de dados de séries temporais e dos seus casos de uso, o que favorece, consequentemente, plataformas de monitorização como a Grafana. Os seus pontos relevantes são:

- Ajuste automática à resolução dos dados para facilitar a sua leitura;
- Apresentação de valores decimais com o controlo do número de casas desejadas;

- Permite legendar os gráficos como os seus eixos;
- *Shared Crosshair* ou *Tooltip* para mostrar informação/valores dos gráficos em simultâneo;
- Em termos de apresentação, é possível alterar uma grande variedade de cores, formas dos gráficos.

3. LOW-POWER WIRELESS AREA NETWORKS

As LPWAN (*Low Power Wireless Area Networks*) são uma subcategoria de WAN (*Wireless Area Network*) que têm vindo a ganhar visibilidade e sido tendência no universo industrial, especialmente em aplicações IoT e comunicação M2M (*Machine to Machine*) [8]. Caraterizam-se, essencialmente pelo baixo consumo energético, cobertura alargada e baixas taxas de transmissão com pequenos pacotes de dados (ver Figura 20). Existe um conjunto de características das LPWAN (ver Tabela 4), que surgem da necessidade de vários sistemas IoT exigirem um maior alcance de comunicação, para se tornarem aplicações eficientes e válidas. Especialmente no âmbito das *smart cities*, em que os pontos essenciais são a comunicação de longo alcance e um consumo energético baixo [8], [42].

Estas tecnologias oferecem um conjunto de funcionalidades como: controlo de utilizadores, modos de poupança de energia, gestão de interferências, técnicas de segurança, localização exata e flexibilidade no desenvolvimento e atualização de *software*. É espectável que na grande parte dos casos de uso, sejam utilizadas estas funcionalidades [8], [43].

Tabela 4 Características gerais de tecnologias LPWAN [43]

Características gerais	
Alcance	Zonas Rurais – 10 km a 40 km; Zonas urbanas – 1 km a 5 km; Máximo – 1000 km
Tamanho dos pacotes de dados	10 a 1000 bytes
Velocidade de transmissão	Até 200 Kbps
Frequências de funcionamento	Frequências licenciadas e não licenciadas
Consumo energético	Altamente eficientes. Baterias com tempo de vida de 10 anos
Custo	Componentes pouco dispendiosos

As soluções existentes, 2G, 3G ou 4G, fornecem uma maior cobertura e uma taxa transmissão de dados maior, ao custo de um consumo energético superior. São especialmente desenvolvidas para comunicação de voz e vídeo, não sendo usadas em aplicações com base em sensores. Tecnologias de rádio vastamente usadas, como IEEE 802.11 WLAN (*Wireless Local Area Networks*), IEEE 802.15.3 ZigBee e IEEE 802.15.1 Bluetooth, são adaptadas para aplicações IoT de curto alcance [8], [43].

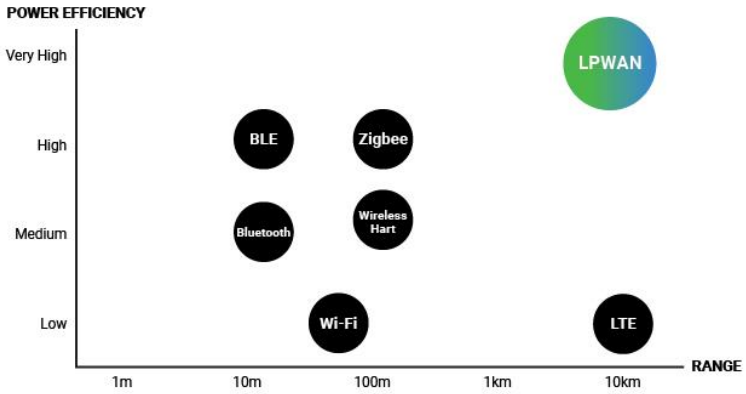


Figura 20 LPWAN: Eficiência energética vs. Cobertura [44]

Devido aos avanços tecnológicos, especialmente na miniaturização de componentes eletrónicos, comunicação, baterias e sensores, as LPWAN conseguem abranger aplicações inteligentes numa vasta gama de atividades (ver Figura 21). Os pontos principais para as mais variadas aplicações, deve-se às vantagens em termos de cobertura, custo e baixa potência que esta tecnologia oferece.

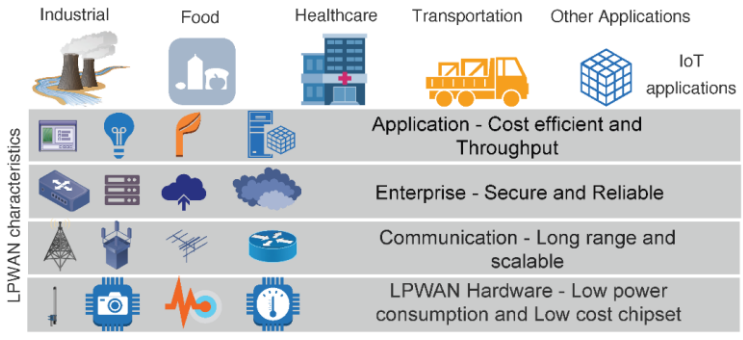


Figura 21 Exemplos de aplicações LPWAN [45]

Um sistema que use uma tecnologia LPWAN, utiliza uma topologia em estrela em que existe um dispositivo central que recebe e transmite os dados (ver Figura 22). A tecnologia LPWAN está presente na comunicação entre os *End Nodes* (por norma

dispositivos periféricos que são constituídos por sensores) e a *Base Station*. A *Base Station* (*Gateway*) é responsável por receber, interpretar e reformular os pacotes a enviar para um servidor local ou um serviço de *cloud* através de TCP/IP (*Transmission Control Protocol/Internet Protocol*). Com a informação armazenada, esta pode ser monitorizada e analisada através de API [44].

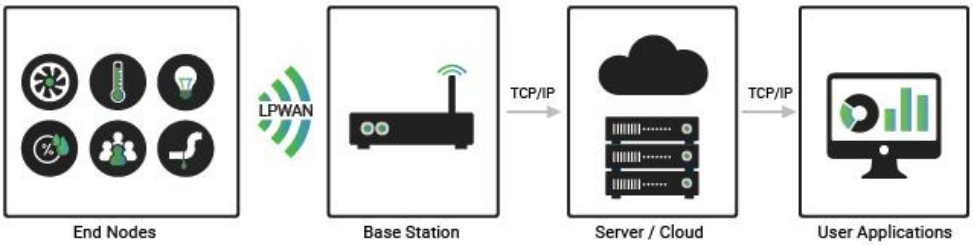


Figura 22 Topologia em estrela das LPWAN [44]

Atualmente, existe uma variedade de tecnologias emergentes que se enquadram na subcategoria LPWAN, como LoRa, Sigfox e NB-IoT. Estes transcetores, destinam-se à recolha de informação de sensores contidos em vários dispositivos, espalhados pela zona de cobertura. Estas, apesar de serem LPWAN têm características de funcionamento diferentes, pelo que requerem uma análise individual quando escolher qual utilizar.

3.1. TECNOLOGIA LORA

LoRa (*Long Range*) é uma tecnologia desenvolvida pela empresa Cycleo (adquirida pela Semtech), que usa frequências não licenciadas (sub-GHz) para efetuar comunicação de dados. Tecnologia vastamente usada, consegue trocar dados, bidireccionalmente, através de modulação com base na tecnologia CSS (*Chirp Spread Spectrum*) que usa impulsos lineares de banda estreita num canal de banda larga. Assim, envia sinais com baixo ruido que são resilientes a perturbações. Um aumento da largura de banda do canal de comunicação melhora a relação sinal-ruido (SNR – *Signal-to-noise ratio*), segundo o teorema de Shannon-Hartley. Permite, desta forma, cobrir uma área de quilómetros a um preço reduzido e aberto a qualquer individuo [42], [46].

Com base na técnica de comunicação LoRa, foi desenvolvido o protocolo LoRaWAN, que pode ser usado em conjugação com a camada de modulação/física. É

possível usar o modelo OSI (*Open System Interconnection*) para enquadrar a técnica de modulação como o protocolo LoRaWAN nas diferentes camadas que o modelo apresenta. Desta forma, a camada física (camada que inclui o *hardware* necessário para que haja troca de informação) e a camada ligação de dados (camada que permite a comunicação entre dois dispositivos da mesma rede, aplicando controlo de fluxo e erros) podem corresponder ao envio de informação por modelação de sinal, ou seja, a camada LoRa. Por sua vez, o protocolo LoRaWAN corresponderá à camada de ligação de dados e à camada de rede (camada que facilita a transferência de dados entre duas redes diferentes) (ver Figura 23).

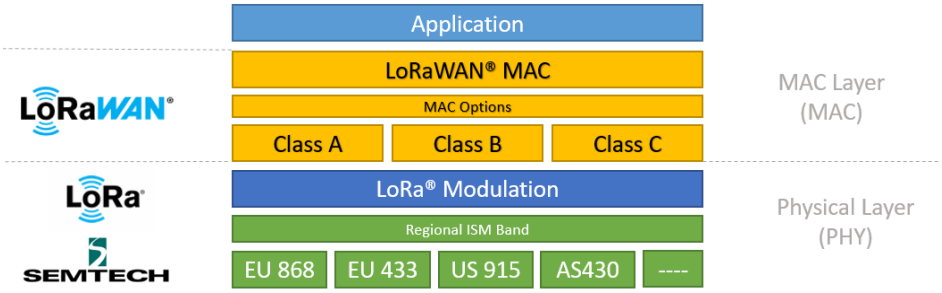


Figura 23 *Stack* da tecnologia LoRa [47]

3.2. PROTOCOLO LORA: CAMADA FÍSICA

A camada física LoRa tem como base a modulação por CSS (*Chirp Spread Spectrum*). Presente em aplicações radar desde os anos 40, é referenciada pela sua robustez e pelos mecanismos que evitam a propagação de ruído [42]. Trata-se de uma tecnologia de banda larga que usa *chirps* de frequência (aumento da frequência, *up-chirps* e diminuição da frequência, *down-chirps*, de acordo com pulsos sinusoidais, ao longo do tempo) como modulação de sinais. Usa três níveis de banda larga, 125 kHz, 250 kHz e 500 kHz e um *Spreading Factor* que determina, consequentemente a taxa de envio da informação (*Bit Rate*) [48]. Para além do *spreading factor*, existem parâmetros como a *bandwidth* (largura de banda), *carrier frequency* (frequência da onda portadora), *coding rate* (taxa de codificação) e *transmission power* (potência de transmissão).

3.2.1. SPREADING FACTOR (SF)

O *spreading factor* (SF) tem impacto na comunicação através de LoRa. Determina quantos *chirps*, são usados nos dados, por segundo. Um menor SF utiliza de mais *chirps* por

segundo, o que permite codificar mais informação com um tempo de transmissão baixo. Para um SF maior, reduz a quantidade de informação a ser codificada, no entanto amplia o tempo de transmissão [49], [50] .

A modulação LoRa conta com seis níveis de SF (SF 7 a SF 12). Em termos práticos, a seleção de um maior SF, amplia o alcance de comunicação, aumenta o consumo energético e reduz a taxa de transmissão, numa largura de banda constante (ver Figura 24). Verifica-se que este parâmetro deve ser devidamente ajustado, visto que existe uma permuta destas características, de forma a corresponder às necessidades da aplicação em concreto [49], [50], [51].

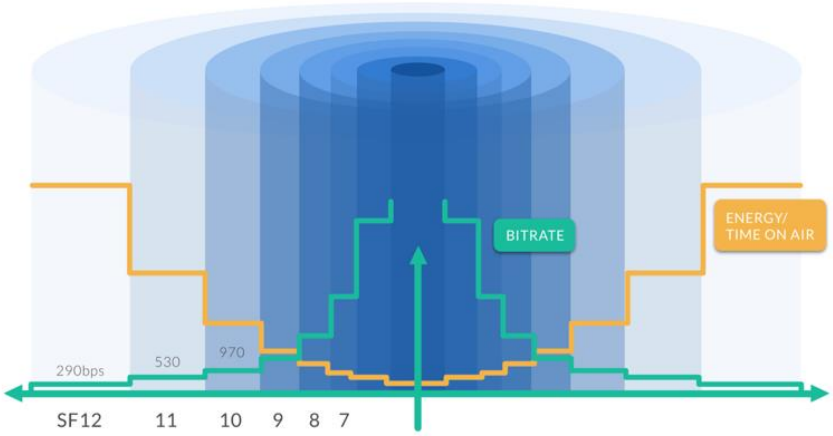


Figura 24 Efeito do *spreading factor* na modulação LoRa [49]

3.2.2. FREQUÊNCIA DA ONDA PORTADORA E RECOMENDAÇÕES EUROPEIAS

Frequências que pertencem à banda ISM (*Industrial, Scientific and Medical*), porções no espetro das frequências rádio destinadas a aplicações industriais, científicas e médicas. A frequência da onda portadora depende da região do planeta em que se pretenda desenvolver comunicações por LoRa (ver Figura 25). Assim, como são frequências não licenciadas, é possível comunicar dentro destes planos de frequência sem necessitar de licença.

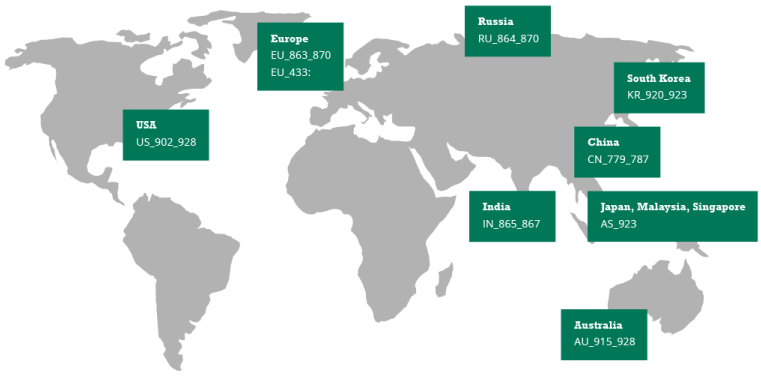


Figura 25 Planos de frequências para comunicação LoRa [52]

No continente europeu os planos são EU863_870 e EU433 que correspondem a frequências 863 MHz a 870 MHz e 433 MHz. No entanto, existem recomendações para dispositivos de baixo alcance na Europa, CEPT/ERC Rec. 70-03.

3.2.3. LARGURA DE BANDA (BW)

A largura de banda corresponde ao conjunto de frequências em que os dados são transmitidos. Valores de largura de banda elevados, implicam um aumento da taxa de transmissão (*bit rate*) e uma diminuição do alcance e da sensibilidade (devido à introdução de ruído). Em sentido inverso, frequências mais baixas invertem o sentido destas características [53].

De acordo com a onda portadora, para a banda ISM de alta frequência (868 MHz e 915 MHz) é comum o uso das larguras de banda 125 kHz, 250 kHz e 500 kHz. No caso das baixas frequências (160 MHz e 480 MHz) é possível optar por larguras da banda de 7.8 kHz, 10.4 kHz, 15.6 kHz, 20.8 kHz, 31.2 kHz, 41.7 kHz e 62.5 kHz [46].

3.2.4. *CODING RATE* (CR)

O *coding rate* previne que interferências bruscas perturbem o sinal que está a ser transmitido. Tem como base o método FEC (*Foward Error Correction*) que na transmissão de sinais digitais, acrescenta bits redundantes, em que o recetor consegue detetar erros e recuperar a informação em qualquer parte da mensagem enviada [54]. São codificados 4 bits de dados redundantes em blocos de 5,6 7 ou 8 bits [55]. Este parâmetro apesar de acrescentar uma camada de proteção ao sinal, tem a desvantagem de aumentar o TOA (*time on air*). A comunicação entre dispositivos pode ser realizada com CR diferentes, se for usado um

cabeçalho explícito, visto que é sempre usado o CR de 4/8. Estes podem ser 4/5, 4/6, 4/7 ou 4/8 e representam a razão de dados não redundantes em cada transmissão [50], [56].

3.2.5. POTÊNCIA DE TRANSMISSÃO

A potência de transmissão é a energia associada ao envio de mensagens e pode ser ajustado na gama de -4 dBm a 20 dBm, com resolução de 1dBm. É um parâmetro que influência o alcance do sinal enviado, no entanto existem limitações de *hardware* que alteram a gama de valores para 2 dBm a 20 dBm [56]. As recomendações europeias abordam este parâmetro e são considerados *uplinks* (transmissões do *end node*) com um máximo de 14 dBm (25 mW) e para *downlinks* (transmissões da *gateway* para o *end node*) com um valor máximo de 27 dBm (500 mW). O duty cycle varia de 0.1% e 1% em transmissões por dia e a antena do dispositivo não pode o ganho de +2.15 dBi [57].

3.3. PROTOCOLO LORA: LORAWAN

O protocolo LoRaWAN é patenteado e gerido pela LoRa Alliance, uma associação não lucrativa que tem como objetivo promover e efetuar colaborações no desenvolvimento de soluções IoT em rede, com base na tecnologia LoRa. É uma associação com membros de organizações de várias regiões do globo, desde empresas que desenvolvem sensores, start-ups, do ramo das telecomunicações, entre outros [58].

A LoRaWAN é categorizada como LPWAN e corresponde à camada MAC (*Media Access Control*) que permite a comunicação de dispositivos de baixo consumo energético, a aplicações ligadas à Internet, através de comunicação sem fio de grande alcance. A sua arquitetura é baseada em quatro componentes essenciais (ver Figura 26):

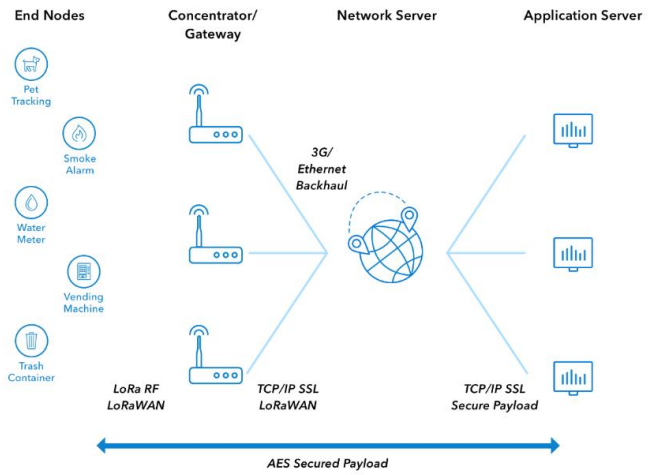


Figura 26 Arquitetura LoRaWAN [59]

- *End Node* – Correspondem aos dispositivos embebidos de baixo consumo, que por norma, adquirem dados em determinados locais, através do uso de sensores. Para que se assegure a comunicação com a *gateway*, estes dispositivos usam transcetores LoRa da empresa Semtech, que detém a patente da técnica de modulação [59], [60].

- *Gateways* – Dispositivos com antenas que recebem os sinais rádio provenientes dos *end nodes*. Apesar do sentido da informação, ser privilegiada do *end node* para a *gateway (uplinks)*, o inverso também é possível (*downlinks*). Este componente pode ser disposto no interior ou exterior de edifícios e tem como função receber e enviar os dados para o servidor de rede [59], [60].

- *Network Server* – Servidor que troca informação com a camada de aplicação e vice-versa. Elimina os dados duplicados da mensagem [56].

- *Application* – Software que permite a arrecadação dos dados e/ou visualização dos mesmos. Visto que, grande parte do interesse é a visualização dos valores obtidos pelos sensores do *end node*, a camada de aplicação pode terminar numa ferramenta de monitorização [59].

De uma forma generalizada, o seu funcionamento consiste na troca de dados entre um *end node* e uma ou mais *gateways*, que reencaminham a informação através de um *backbone* da Internet até a um servidor de rede. Este após receber a informação, elimina dados redundantes da mensagem e envia-a para a camada de aplicação. O típico utilizador

desenvolve o *end node* que pretende usar, assim como a camada de aplicação, pois existem fornecedores de servidores que asseguram a troca de informação entre os dois extremos [56].

No protocolo LoRaWAN os *end nodes* são classificados de acordo com o funcionamento suportado, as classes A, B e C.

Dispositivos em classe A são mais eficientes em termos energéticos, sendo ideal para aplicações de aquisição de dados a partir de sensores. Na maioria do tempo, encontram-se no modo *sleep* (consumo mínimo) e "acordam" para transmitir e receber mensagens. Assim, suportam comunicação bidirecional, entre si e o servidor através da *gateway* [56], [61]–[63].

O seu funcionamento tem como base um ciclo de *uplinks* e *downlinks*. As transmissões (*uplinks*) podem acontecer em tempos aleatórios ou em intervalos constantes. Após cada transmissão, os dispositivos escutam, por duas ocasiões, por possíveis mensagens do servidor (*downlink*). Estas janelas têm um intervalo de tempo específico e acontecem no primeiro e segundo segundos após cada envio (intervalo variável e configurável) (ver Figura 26) [56], [61]–[63].

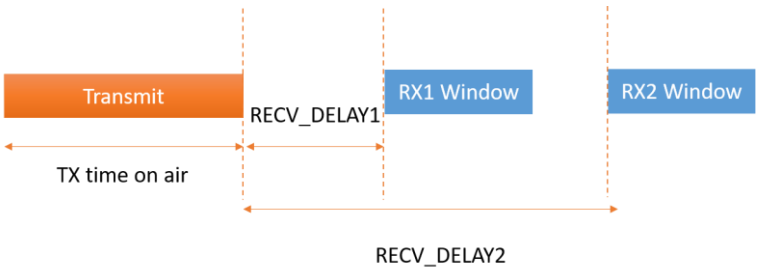


Figura 27 Funcionamento do *end node* em classe A [61]

O servidor caso pretenda comunicar com um *end node*, deve enviar a mensagem logo após receber dados do mesmo dispositivo, ou no decorrer da segunda janela de receção. No caso de o *downlink* não ter sucesso, o servidor só terá oportunidade na próxima transmissão do *end node* [62].

O *end nodes* em classe B comportam-se se forma semelhante aos *end nodes* em classe A, no entanto nestes dispositivos são agendadas as mensagens de *downlink* por parte do servidor. Este processo acontece através da sincronização entre a *gateway* e o *end node* através de *beacons*, em que o dispositivo abre janelas periódicas para a receção de

mensagens do servidor (Figura 28). Assim, o servidor tem conhecimento que o dispositivo em concreto está à escuta e o *downlink* acontece durante uma dessas janelas [63].

Na classe C os dispositivos estão constantemente à escuta, excluindo os períodos de transmissão (ver Figura 28). Com este funcionamento, os *downlinks* não necessitam de ser agendados ou enviados em períodos críticos, como nas classes B e A, respetivamente. É a classe menos eficiente e não é recomendada para *end nodes* alimentados por bateria [63].

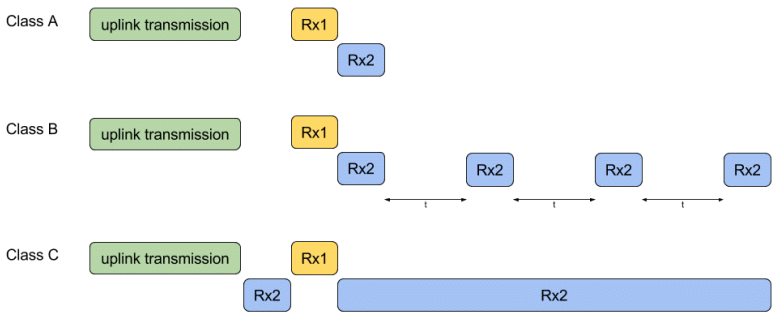


Figura 28 Classes de funcionamento do protocolo LoRaWAN [64]

Neste protocolo a segurança é uma questão importante, visto que as mensagens dos *end nodes* podem passar por várias *gateways* de terceiros, até chegar ao servidor. Sem qualquer proteção estas mensagens podem estar visíveis para os proprietários das *gateways*. Assim, é usada a encriptação AES (*Advanced Encryption Standard*) de 128 bits, em duas camadas de segurança, na camada de rede e na camada de aplicação. Na rede é usada a chave *NwkSKey* (*Network Session Key*) que autentifica o *end node* na rede e valida a mensagem enviada. A chave *AppSKey* (*Application Session Key*) é usada para assegurar que o fornecedor de rede não tem acesso à mensagem, sendo desencriptada na aplicação do utilizador final (Figura 29). Ambas as encriptações têm início no *end node*, o que garante a segurança das mensagens até às aplicações correspondentes [63].

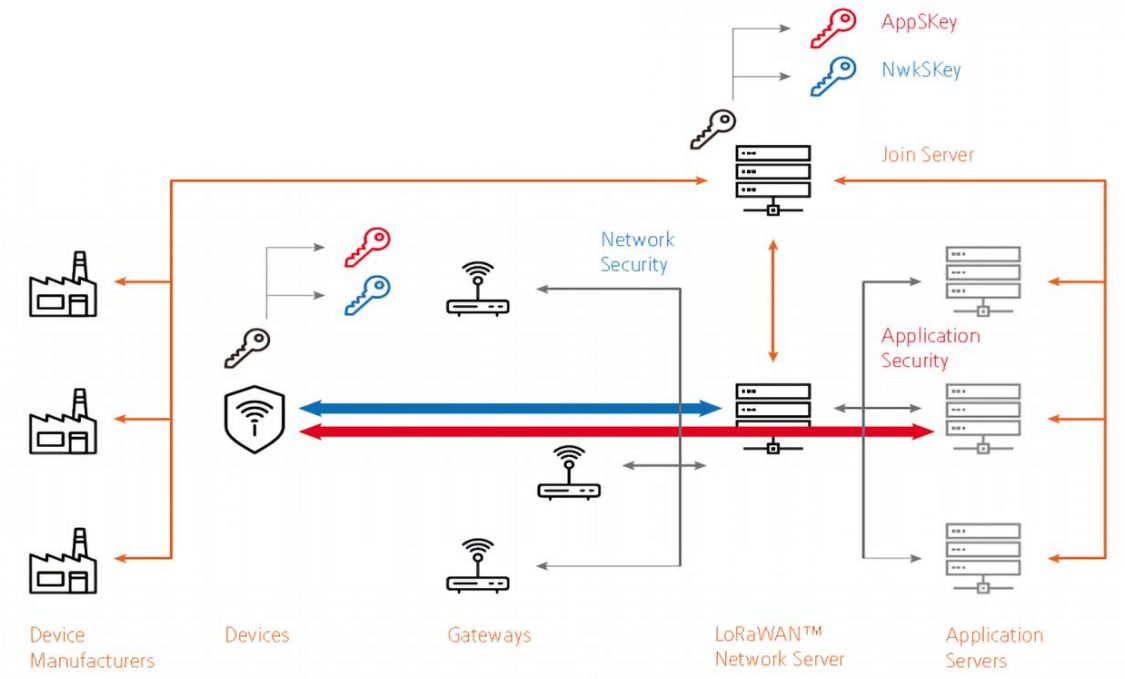


Figura 29 Chaves de encriptação AES [65]

Os fabricantes de dispositivos destinado a comunicações por LoRa atribuem no ato de fabrico, um DevEUI de 64 bits (EUI-64) único, que identifica o equipamento globalmente (semelhante a endereços MAC). Este é usado para identificar o dispositivo antes de ser aceite pela rede. Após a ativação ser bem-sucedida, é atribuído um endereço dinâmico DevAddr de 32 bits (não único) que identifica o *end node* dentro da rede (semelhante a endereços IP) em todas as trocas de informação [66], [67] .

Um dispositivo para fazer parte da rede deve estar autenticado e ativado. A ativação é realizada através de um de dois métodos possíveis, OTAA (*Over-The-Air Activation*) ou ABP (*Activation By Personalization*), diretamente relacionadas com a segurança das transmissões.

A ativação por OTAA é a mais comum e segura. O *end node* envia um pedido de ativação à rede, em que é respondido com um *DevAddr* (*Device Address*) dinâmico e informações adicionais. O *end node* retém o *DevAddr* e as chaves *AppSKey* e *NwkSKey* de forma a conseguir efetuar transmissões. Caso o dispositivo perda as *chaves* ou a rede as expire, o ciclo de ativação terá de ocorrer novamente, em que são geradas chaves de sessão diferentes (ver Figura 30) [63], [68]. Este método permite programar o dispositivo uma única vez, visto que as chaves de sessão são automaticamente geradas [69].

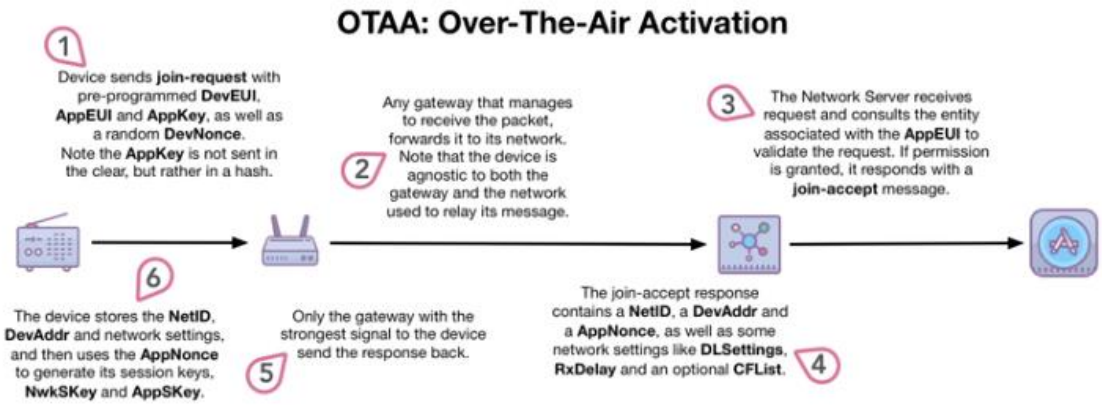


Figura 30 Ativação *Over-The-Air* (OTAA) [69]

Na ativação por ABP, o *DevAddr* é fixo e as chaves de sessão são previamente selecionadas e programadas no *end node*. Estas chaves não sofrem alterações durante o seu tempo de funcionamento. Assim, este método não requer que seja realizado o procedimento de ativação como no método OTAA (ver Figura 31) [63], [68]. É o método de ativação mais rápido a conectar com o servidor e é usado em fases de testes. No entanto, o comprometimento das chaves pode ocorrer caso haja acesso ao dispositivo, tornando este método menos seguro [69].

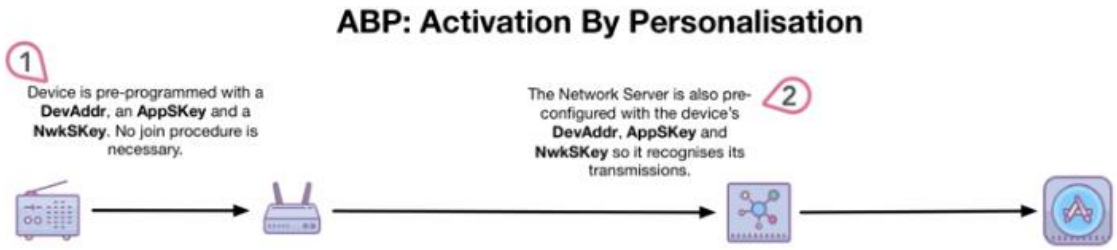


Figura 31 Ativação por Personalização (ABP) [69]

Além das funcionalidades referidas, o protocolo LoRaWAN apresenta vantagens e desvantagens, de acordo com a aplicação desejada. A Tabela 5 descreve um conjunto de características de relevância, para o uso deste protocolo.

Tabela 5 Vantagens e desvantagens do protocolo LoRaWAN [70], [71]

Vantagens	▪ Pode ser implementado sem custos de serviço, só custos associados ao hardware utilizado; ▪ Grande eficiência energética; ▪ Opera com dispositivos em movimento; ▪ Suporte e parceiros como a IBM Microchip Technologies e Cisco; ▪ Boa opção para comunicação bidirecional; ▪ Possível configurar e controlar a própria rede;
Desvantagens	▪ Limites nos *uplinks* e *downlinks*, de acordo com a limitação do *duty cycle*; ▪ Requer a utilização de chips da Semtech; ▪ Requer sempre uma *gateway*; ▪ Não a melhor opção para aplicações em tempo-real;

3.4. SIGFOX

Sigfox é uma tecnologia concebida em França no ano de 2010. Ludovic Le Moan e Christophe Fourtet criaram a empresa com o intuito de criar e comercializar um serviço LPWAN. Neste momento, Sigfox está presente em 75 países, disponibilizando soluções IoT a empresas, com base numa rede LPWAN própria. É considerado um dos maiores ecossistemas IoT da atualidade, com 19 milhões de dispositivos, que enviam 76.2 milhões de mensagens por dia. Neste momento, conta com 6 milhões de quilómetros quadrados de cobertura, abrangendo 1,3 mil milhões de pessoas [72], [73] .

A empresa é detentora da técnica de modulação *Differential Binary Phase Shift Keying* (DBPSK), tecnologia *Ultra-Narrow Bandwidth* (UNB), que permite a interligação de dispositivos e a sua comunicação bidirecional, até 50 km. Esta técnica modifica a fase da onda portadora para codificar a informação, numa banda do espectro muito reduzido, o que proporciona a mitigação de interferências ou ruídos no sinal [70], [72].

Geralmente, um dispositivo emite uma mensagem que é recebida por uma ou diversas *base stations* (*gateways*). Esta é reencaminhada para a *cloud* Sigfox, que consequentemente, é enviada para a *backend* do cliente (ver Figura 32). Os *end nodes* enviam pacotes em três canais de diferentes frequências, como forma de garantir que uma *base station* receba os pacotes. Assim, no caso de vários dispositivos estarem a emitir mensagens, as *base stations* conseguem, simultaneamente, recebê-las com sucesso. Com esta

redundância de envio, as *base stations* não usam um sistema de *acknowledge* (ACK) para confirmar os *uplinks* recebidos [72], [74], [75].

Figura 32 Rede de serviços e mensagens Sigfox [76]

A utilização da tecnologia UNB limita a taxa de transmissão dos *uplinks* para 100 bps, no continente europeu, e 600 bps na América do Norte/Ásia. As comunicações usam frequências sub-GHz não licenciadas como 868 MHz na Europa, 433 MHz na Ásia e 915 MHz na América do Norte. Devido a restrições europeias, mais concretamente, sobre o *duty cycles* de 1%, implica que possam ser enviados, por dia, 140 *uplinks* com um máximo de 12 bytes e 4 *downlinks* até 8 bytes (ver Figura 33) [72], [74], [75].

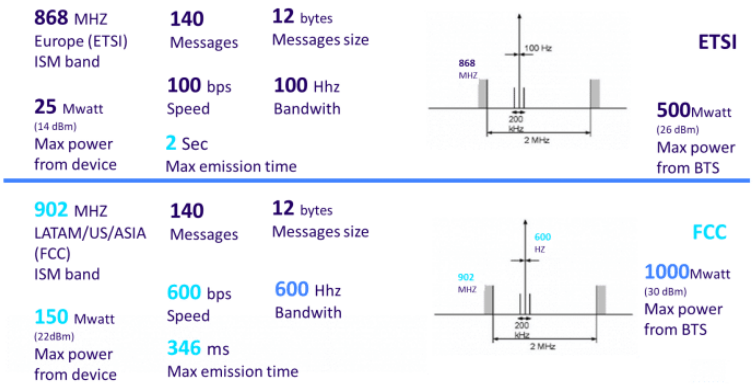


Figura 33 Transmissões Sigfox [76]

Sigfox funciona com pequenas mensagens, não sendo ideal em casos em que a mensagem necessite de ser maior que 12 bytes. O baixo número de transmissões e o seu tamanho reduzido, torna os dispositivos altamente eficientes energeticamente, possibilitando que as suas baterias durem décadas. Devido ao a estas caraterísticas, a solução Sigfox pode não ser indicada para aplicações que envolvam conjuntos de dados maiores, assim como aplicações em tempo real. A Tabela 6 resume algumas características desta tecnologia/serviço IoT.

Tabela 6 Vantagens e desvantagens do protocolo Sigfox [70], [71]

Vantagens	▪ Cobre uma grande área, principalmente na Europa; ▪ Consome pouca energia nas transmissões; ▪ Otimizado para pequenos dispositivos que emitam irregularmente, com intervalos de tempo significativos;
Desvantagens	▪ Existe custo de serviço, para além do custo do hardware; ▪ Possui 16 bits de encriptação; ▪ Pequeno tamanho das mensagens; ▪ Potenciais interferências com frequências rádio mais elevadas; ▪ Depende da área de cobertura, impedindo a implementação em alguns casos de uso; ▪ Não indicado a aplicações em tempo-real;

3.5. NARROWBAND-IOT (NB-IOT)

Narrowband-IoT é uma tecnologia LPWAN de banda estreita, apresentada em 2016 no consórcio 3GPP (*3rd* Generation Partnership *Project*), que reutiliza o protocolo móvel LTE, otimizando-o de forma, a conceber uma solução para comunicações IoT entre dispositivos. Assim, as suas grandes valências são, a compatibilidade com redes móveis, a reutilização de hardware e a ocupação do mesmo espectro sem haver interferências. Esta tecnologia usa os protocolos LTE/4G (*Long Term Evolution*) ou GMS (*Global System for Mobile Communication*) em frequências licenciadas [72], [74].

O protocolo NB-IoT pode ser implementado de acordo com três modos de funcionamento: *Stand alone operation*, em que é usada uma portadora própria GMS (200 kHz de banda larga) para a comunicação; *Guardband operation*, onde ocupa a largura de banda de uma portadora LTE; *In-band operation*, que ocupa a *guardband* (banda frequências estreita que separa duas larguras de banda, de forma a evitar interferências) de uma portadora LTE (ver Figura 34) [74].

O protocolo LTE usa um *backend* para fazer *broadcast* da informação a todos os *end nodes*, o que provoca um desperdício energético nos dispositivos. O protocolo NB-IoT reduz as funcionalidades LTE, como a verificação da qualidade de transmissão, com o objetivo de otimizar as transmissões para aplicações IoT. É concebido para efetuar trocas de pequenos pacotes de mensagens e suporta para mais de 100 mil dispositivos [74], [78], [79].

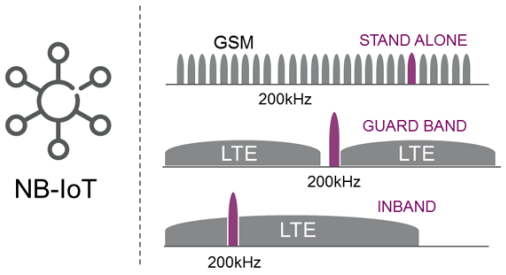


Figura 34 Modos de operação de transmissão NB-IoT [77]

Usa modulação QPSK (*Quadrature Phase Shift Keying*), com a técnica FDMA (*Frequency Division Multiple Access*) em *uplinks* e OFDMA (*Orthogonal Frequency Division Multiple Access*) para *downlinks.* As taxas de transmissão são significativas, com 20 kbps de *uplink* e 200 kbps em *downlinks*, em mensagens que podem chegar a 1600 bytes. De acordo com os tamanhos das mensagens, as baterias que alimentam os *end nodes*, podem chegar a 1 década de funcionamento [74], [78], [79].

Apesar da semelhança entre o protocolo LTE/4G e o protocolo NB-IoT, este possui caraterísticas importantes na implementação de soluções IoT. Assim, a Tabela 7 pretende enumerar algumas vantagens e desvantagem, da sua utilização.

Tabela 7 Vantagens e desvantagens de protocolo NB-IoT [70], [80], [81]

Vantagens	▪ Cobertura extensa, com funcionamento estável dentro de edifícios; ▪ Tempos de resposta rápidos, garantindo estabilidade e qualidade de serviço; ▪ Autenticação e conexão segura; ▪ Envio de pacotes intermitentes até 1600 bytes; ▪ Encriptação de 256 bits 3GPP;
Desvantagens	▪ Dificilmente funciona em dispositivos em movimento; ▪ Complexidade no desenvolvimento de *firmware*, para envio de mensagens de maiores dimensões; ▪ Custo de serviço das operadoras;

Com o crescimento do conceito IoT, as operadoras de comunicações, observam este protocolo como uma forma de rendimento futuro. Em 2019, a empresa Altice foi a primeira a investir nesta tecnologia em Portugal, com um modelo de conectividade semelhante ao 3G e 4G, cobrindo a totalidade do país. Atualmente, comercializa um conjunto de soluções com parceiros e clientes, a nível industrial, na utilização da sua rede NB-IoT, de forma a impulsionar o futuro das comunicações IoT [82].

3.6. ANÁLISE ENTRE LORAWAN, SIGFOX E NB-IOT

As LPWAN referidas, têm diferenças assinaláveis, que podem ter particularidades importantes, na consideração de qual utilizar. A aplicação que se pretende desenvolver, assim como os custos associados *vs*. aproveitamento, têm um papel determinante. No sentido de fazer esta análise, podem ser comparadas características como a qualidade de transmissão da informação ou confiabilidade, capacidade de envio, área de cobertura, eficiência energética, segurança e custos. A Tabela 8 apresenta uma comparação direta entre as três tecnologias, de acordo com alguns destes fatores.

Tabela 8 Comparação entre LoRWAN, Sigfox e NB-IoT [43]

Características	**LoRaWAN**	**Sigfox**	**NB-IoT**
Frequências	Sem licença ISM - 433 MHz (Ásia), 868 MHz (Europa) e 915 MHz (USA)	Sem licença ISM - 433 MHz (Ásia), 868 MHz (Europa) e 915 MHz (USA)	Frequência licenciada LTE
Taxa de Transmissão	50 kbps	100 bps	20/200 kbps
Taxa de transmissão adaptativa	Disponível	Não disponível	Não disponível
Cobertura	5 km (urbano) / 20 km (rural)	10 km (urbano) / 40 km (rural)	1 km (urbano) / 10 km (rural)
Mensagens por dia	De acordo com os limites do *duty cycle*	140 *uplink* / 4 *downlink*	Ilimitado
Tamanho das mensagens	51 Bytes (Europa) ou 11 Bytes (USA)	12 Bytes	1600 Bytes
Conectividade	*End nodes* conectam a várias *base stations*	*End nodes* conectam a várias *base stations*	*End nodes* conectam a uma *base station*
Autenticação/Encriptação	AES 128 bits	Não suportado	3GPP 256 bit
Resistência a interferências	Baixa	Baixa	Alta
Modulação	BPSK	CSS	QPSK

3.6.1. QUALIDADE DE TRANSMISSÃO DE INFORMAÇÃO

Devido à utilização de frequências não licenciadas, os protocolos LoRaWAN e Sigfox estão sujeitas a diversas interferências do meio em que se insere. Estas acontecem devido à quantidade de materiais que possam obstruir o sinal, ou a colisões de mensagens, de acordo com o número de dispositivos a transmitir. À medida que a distância de transmissão aumenta, a percentagem de mensagens perdidas também aumenta (ver Figura 35).

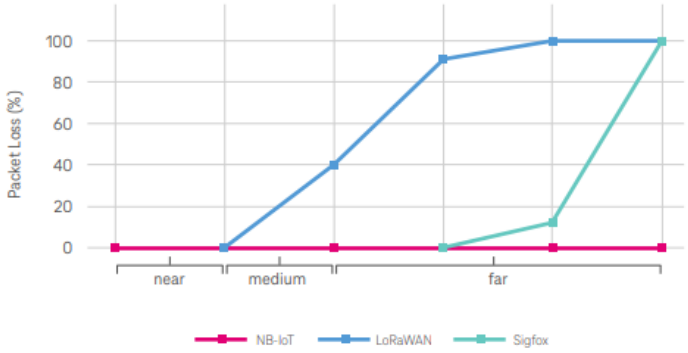


Figura 35 Percentagem de mensagens perdidas pela distância de transmissão [83]

A NB-IoT funciona em frequências licenciadas, com sistemas que previnem colisões de forma a fornecer a solução mais consistente e confiável. A distância de transmissão neste protocolo tem um impacto mínimo, pelo que, menos de 5% das mensagens podem ser perdidas. Em termos de confiabilidade de envio da informação, a NB-IoT apresenta ser a opção mais eficiente [83], [84].

3.6.2. CAPACIDADE DE ENVIO

LoRaWAN e Sigfox têm limitações no número de pacotes que podem ser enviados por dia. Este limite provém da imposição do *duty cycle* de 1% para frequências não licenciadas.

Os dispositivos Sigfox têm de dividir as mensagens maiores, aumentando o consumo energético. O número de *uplinks* é fixo, para qualquer situação e não é permitido atualizar o *firmware* dos dispositivos. Verifica-se ainda, que a taxa de transmissão é a menor das três tecnologias [83].

O protocolo LoRaWAN possui o *spreading factor*, anteriormente referido, que permite ajustar a transmissão numa gama de frequências maior. Um maior *spreading factor* reduz significativamente o número de mensagens que podem ser enviadas. Com um SF 7/8 (pequenas distâncias), podem ser enviadas, em teoria, 242 mensagens, já com o SF 10, podem ser emitidas 51 mensagens. [83].

NB-IoT não define limite máximo de *uplinks*, o que torna a troca de comunicação mais fluida e consistente. A taxa de transmissão neste protocolo é significativamente superior aos anteriores, atingindo os 27 kbps em conexões boas a excelentes. Em termos de capacidade e rapidez de transmissão, NB-IoT é claramente o protocolo mais vantajoso para aplicações que envolvam uma grande quantidade de dados e frequência de envio [83].

3.6.3. ÁREA DE COBERTURA

O protocolo LoRaWAN ao pertencer à LoRa Alliance consegue estar presente em mais de 160 países, apesar de existirem diferentes frequências de funcionamento, de acordo com a região do mundo. Hoje em dia, os dispositivos produzidos e comercializados já possuem suporte para as principais regiões: Europa, Ásia e América do Norte. Sigfox, apesar de estar presente em mais de 70 países, possui um número pequeno de *base stations* distribuídas pelos países, o que diminui a confiabilidade da área de cobertura. As tecnologias que usem frequências não licenciadas têm dificuldade em manter conexões estáveis, principalmente no interior de edifícios [83]..

A NB-IoT ao utilizar rede móvel, está presente em vários países, permitindo que mais de 100 operadores (em 54 países) comercializem os seus serviços com base NB-IoT [83].

3.6.4. INVESTIMENTO

A implementação destas tecnologias, tem um custo/investimento associado de acordo com a aplicação em causa. Assim, existe a opção de desenvolver e controlar a própria rede privada ou investir num serviço pago, fornecido pelos operadores de rede. O hardware tem sempre um custo e é independente da tecnologia a implementar. Devido à vasta produção destes dispositivos, estes estão a tornar-se cada vez mais baratos. A Figura 36 demonstra o investimento expectável para a execução destes protocolos [83].

	NB-IoT	LoRaWAN		Sigfox
		Own local network	Subscription with operator	
Devices incl. radio module	€	€	€	€
SIM cards	€	–	–	–
Subscription fees*	€	–	€	€
Own network infrastructure	–	€	–	–
Network ops & maintenance	–	€	–	–
Application (server)	€	€	€	€

Figura 36 Investimento de implementação LPWAN [83]

4. ARQUITETURA DO SISTEMA

De acordo com os objetivos descritos no capítulo 1 e do estudo efetuado, nos capítulos 2 e 3, foram tomadas opções para a melhor implementação do projeto. Desta forma, a recolha e apresentação de informação passa pelos seguintes componentes: um microcontrolador; sensores; transcetores; servidor; base de dados e ferramenta de monitorização. Neste capítulo, apresentam-se os elementos do sistema utilizados assim como a sua interligação. Como introdução ao sistema desenvolvido, a Figura 37 apresenta um diagrama do projeto, com o *hardware* e o *software* utilizados no decorrer do desenvolvimento.

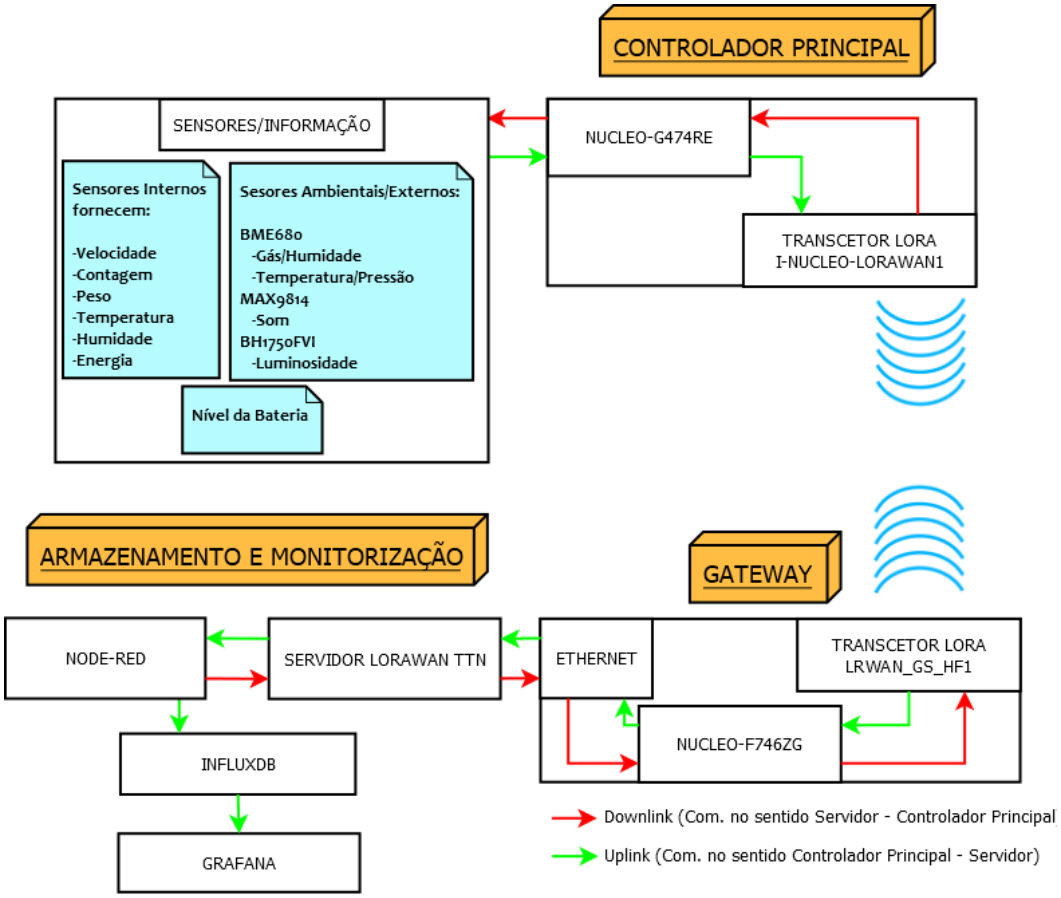


Figura 37 Diagrama do projeto

4.1. PARÂMETROS A MONITORIZAR

Numa primeira fase, são considerados os dados pretendidos, de forma a analisar e optar por sensores que forneçam com precisão os valores corretos ou próximo da realidade. Estes são distinguidos em dois grupos, os sensores internos, referentes ao sistema NEXT-Road e os sensores ambientais/externos, que quantificam o meio envolvente em valores digitais ou analógicos. Frisar que os sensores internos não fazem parte dos objetivos deste projeto, apenas sendo referidos, os valores que são enviados.

4.1.1. INFORMAÇÃO A ENVIAR

O sistema NEXT-Road destina-se a ser implementado em locais críticos das vias rodoviárias de forma a prevenir acidentes e controlar a velocidade dos veículos em circulação. Assim, cada um destes sistemas precisará de um conjunto de sensores internos e ambientais, que indiquem o estado daquele local e do próprio sistema. Esta informação é relevante para se definirem ciclos e padrões de comportamento dos condutores, assim como deteção de erros ou alertas de valores excessivos.

Os parâmetros ambientais são os dados obtidos através dos sensores ambientais/externos e que se situam ligados diretamente ao controlador principal. São efetuadas medições de temperatura, humidade, pressão atmosférica, indicadores da presença de gases, luminosidade e intensidade sonora. Com este conjunto de parâmetros é possível indicar a visibilidade dos condutores, o estado do tempo e a presença/concentração gases nocivos.

Os parâmetros internos dizem respeito aos dados de sensores inseridos no sistema NEXT-Road. Neste é ambicionado obter valores de energia, temperatura, humidade, aceleração e peso. Com esta informação, podem ser retidos dados indiretos, como:

- Velocidades do veículo durante a travessia pelo pavimento NEXT-Road, através das variações dos valores dos acelerómetros instalados. São enviadas velocidades de entrada e saída do pavimento;
- Classe dos veículos pelo peso registado durante a sua passagem;
- Contagem de veículos que atravessam o pavimento, pelo número de ativações dos acelerómetros.

Além dos dados provenientes dos sensores, os dados indiretos são recebidos e reencaminhados pelo controlador principal. Esta informação deve, posteriormente, ser analisada e cruzada, de maneira a possuir a maior informação possível em casos de acidente ou emergência. A Tabela 9 resume os parâmetros descritos, de acordo com a sua origem.

Tabela 9 Parâmetros pretendidos do sistema

Parâmetros Internos	**Parâmetros Ambientais/Externos**
▪ Velocidade de entrada ▪ Velocidade de saída ▪ Peso ▪ Contagem ▪ Energia gerada ▪ Temperatura ▪ Humidade	▪ Temperatura ▪ Humidade ▪ Pressão atmosférica ▪ Luminosidade ▪ Intensidade sonora ▪ Nível da bateria

A alimentação dos sistemas envolvidos, depende de duas baterias de 12 V, que são devidamente carregadas durante a passagem de veículos. Como parâmetro de monitorização, o nível das baterias permite detetar/prever os momentos de troca ou de falha de alimentação do sistema. Assim, através de um conversor analógico-digital (ADC) é possível fazer essa análise e reencaminhar pelo sistema, um valor aproximado do nível de carregamento da bateria. É ainda expectável que, em períodos de grande movimento, a bateria carregue e vice-versa.

4.1.2. CADÊNCIA DE ENVIO

A cadência de envio é um fator importante nesta aplicação e na escolha do protocolo LPWAN a implementar. Depende da importância dos dados e da taxa de atualização dos mesmos. Desta forma, pode-se considerar que,

- a informação relativa à passagem dos veículos é a mais relevante. Assim, dados como a contagem, velocidade e peso, devem ser enviados com uma cadência maior;

- seguidamente, os parâmetros internos do sistema NEXT-Road, como a temperatura, humidade e energia gerada pelo pavimento. Esta informação deve acompanhar, a uma frequência menor, as mensagens dos veículos;

- no sentido de acompanhar o estado do sistema NEXT-Road, são enviadas mensagens de erros. Estas consistem na deteção de hardware com defeito ou perda de ligação, e na deteção de valores de sensores, acima ou abaixo do esperado;

- por fim, as mensagens com menor prioridade, são as referentes aos parâmetros ambientais. Estes podem ser atualizados entre vários minutos ou horas. Têm o propósito de enquadrar os dados com o meio que rodeia o sistema.

Tendo em consideração esta abordagem, as mensagens têm uma organização semelhante ao referido. Estas são posteriormente, enviadas pelo controlador principal de acordo com o protocolo LPWAN escolhido.

4.1.3. MÉTODO DE RECEÇÃO

A informação chega ao controlador principal de formas diferentes, de acordo com os protocolos implementados no sistema NEXT-Road. No caso dos sensores ambientais, estes estão diretamente conectados ao controlador principal e comunicam através do protocolo série I²C (*Inter-Integrated Circuit*) ou por conversor analógico-digital, em sensores analógicos. Os restantes parâmetros são enviados por protocolo CAN FD ou RS485. Assim, o controlador principal está preparado para receber parâmetros internos, através destes dois protocolos e efetuar a devida organização da informação, consoante a própria estrutura das mensagens recebidas.

4.1.4. MÉTODO DE ENVIO (LORA)

Após a definição da informação a receber, é necessário optar por uma tecnologia LPWAN, que permita reencaminhar os dados para um servidor/base de dados. Da análise efetuada no capítulo 3, a opção para o desenvolvimento deste projeto, recai no protocolo LoRaWAN. Devido às suas características, pode ser implementado com flexibilidade e de acordo às necessidades do projeto, numa rede comunitária aberta, sem custos adicionais. Em

fase de teste, o investimento é mínimo comparado com as restantes soluções abordadas no capítulo 3. Salientar que a empresa Pavnext, tem um historial de testes com transcetores LoRa, assim como hardware necessário para a implementação deste projeto.

Durante o uso do protocolo LoRaWAN, ao usufruir da rede aberta, a quantidade de dispositivos a transmitir tem influência na qualidade de transmissão das mensagens, para além de existir um limite restrito de *uplinks*. Estes inconvenientes, podem ser ultrapassados pela gestão e organização da informação e pela configuração do transcetor LoRa, respetivamente.

A utilização deste protocolo obriga a que exista uma *gateway*, no alcance dos controladores principais, que consiga receber a mensagem LoRa emitida. A *gateway* tem o objetivo de desmodular o pacote e reencaminhá-lo para um servidor LoRaWAN.

4.1.5. HARDWARE (SENSORES/MICROCONTROLADOR)

Definidos os modos de receção e envio da informação é fundamental optar por hardware que possibilite a melhor solução para o projeto. Desta forma, a componente de hardware é composta pelos sensores ambientais, controlador principal, *gateway* e transcetores LoRaWAN.

4.1.5.1. SENSOR BME680

O BME680 é o um sensor digital de alta precisão e linearização para medições de gás, pressão, temperatura e humidade (ver Figura 38). Desenvolvido para dispositivos móveis ou de uso quotidiano, apresenta um tamanho reduzido e um baixo consumo energético. Permite ser configurado por modos de funcionamento em que estes influenciam diretamente o seu consumo. Desta forma, este sensor pode ser ajustado a vários casos de uso, como [85], [86]:

- Monitorização pessoal da qualidade do ar;
- Monitorização da qualidade do ar em transportes;
- Indicador da qualidade do ar, IAQ (*index for air quality*);
- Monitorização do clima e possível previsão;
- Mapeamento de zonas de acordo com os valores IAQ;

- Automação e controlo de equipamentos domésticos.

Figura 38 Sensor Bosch BME680

O sensor é capaz de fazer medições de VOCs (*Volatile Organic Compounds*) do ar que o rodeia. Gases como etano, etanol, acetona, monóxido de carbono e isopreno. Alguns dos exemplos mais comuns são de gases provenientes de tintas, lixo, respiração e cozinhados. Estes valores são obtidos através de medições do valor de resistência, de acordo com as concentrações de VOC (para concentrações elevadas, a resistência diminui e vice-versa) [85].

Baseado num algoritmo inteligente, o sistema de IAQ, usa os valores VOC, permitindo quantificar a qualidade do ar num local. Consegue retornar um intervalo de valores entre 0 e 500, possui uma resolução de 1 e apresenta possíveis desvios de ±15% [85].

A Tabela 10 apresenta intervalos de valores e a sua respetiva classificação, em termos de impacto para a saúde, assim como ações recomendadas para contrariar esses efeitos.

Tabela 10 Classificação do IAQ [85]

IAQ Index	Air Quality	Impact (long-term exposure)	Suggested action
0 – 50	Excellent	Pure air; best for well-being	No measures needed
51 – 100	Good	No irritation or impact on well-being	No measures needed
101 – 150	Lightly polluted	Reduction of well-being possible	Ventilation suggested
151 – 200	Moderately polluted	More significant irritation possible	Increase ventilation with clean air
201 – 250[9]	Heavily polluted	Exposition might lead to effects like headache depending on type of VOCs	optimize ventilation
251 – 350	Severely polluted	More severe health issue possible if harmful VOC present	Contamination should be identified if level is reached even w/o presence of people; maximize ventilation & reduce attendance
> 351	Extremely polluted	Headaches, additional neurotoxic effects possible	Contamination needs to be identified; avoid presence in room and maximize ventilation

O sensor suporta dois modos de funcionamento de baixo consumo, *Sleep* e *Forced* (Tabela 11). Estes podem ser alterados com a escrita num registo definido para o efeito. De forma automática, o sensor ao ser iniciado encontra-se em modo *Sleep*. Se uma leitura estiver a decorrer, não é possível escrever em registos até que o ciclo de medições termine, garantindo a qualidade e integridade da informação [85].

Tabela 11 Modos de utilização do sensor BME680

Modo *Sleep*	Não são realizadas quaisquer medições, pelo que o sensor apresenta o consumo mínimo de energia neste modo.
Modo *Forced*	É realizada uma medição por cada sensor, em sequência. Durante as medições é efetuado o aquecimento do sensor de gás, antes da sua leitura. Após o ciclo terminar, o modo é alterado para *Sleep*, automaticamente, de maneira a poupar energia.

No ciclo de leitura as medições têm uma determinada ordem: Temperatura, Pressão. Humidade e Gás (ciclo TPHG). Na Figura 39 é demonstrado o ciclo de medições, quando o sensor entra no modo *Forced.*

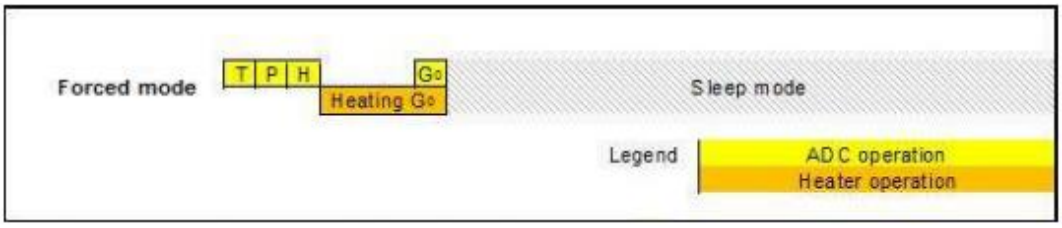


Figura 39 Modo *Forced* do sensor BME680

Existem ainda, 4 modos energéticos para o sensor BME680 que influenciam diretamente a frequência de aquisição dos dados. A Tabela 12 apresenta as suas diferenças, assim como aplicações de uso geral adequadas a cada modo [85].

Tabela 12 Modos energéticos do sensor BME680

Power Mode	***Update Rate***	**Consumo médio de corrente**	**Aplicações**
***Ultra-Low Power* (ULP)**	3.3 mHz	0.09 mA	Dispositivos alimentados a bateria
***Quick Ultra-Low Power* (q-ULP)**	0.33 Hz	0.1 mA	Dispositivos alimentados a bateria
***Low Power* (LP)**	0.33 Hz	0.9 mA	Aplicações interativas onde se deseja que a frequência de leituras seja mais elevada
Contínuo (somente para testes)	1 Hz	12 mA	Usado em casos de testes ou exceções

Como o sensor BME680 integra 4 sensores, é necessária uma análise de especificações de cada um, para que uma possível utilização e integração no sistema sejam adequadas. Assim, a Tabela 13 apresenta, para cada sensor, parâmetros de maior relevo [85].

Tabela 13 Parâmetros sensores TPHG

Sensor	**Condições de Funcionamento**	**Gama de valores**	**Resolução**	**Precisão**	**Tempo de resposta**
Humidade	-40 ºC a 85 ºC	0% -100% HR	0.008% HR	±3%	8 s
Temperatura	-40 ºC a 85 ºC	0ºC a 65ºC	0.01 ºC	±0.5 ºC (25 ºC)	N/A
Pressão	-40 ºC a 85 ºC	300 a 1100 hPa	0.18 Pascal	±0.6 hPa	N/A
Gás	-40 ºC a 85 ºC / 10% HR a 95% HR	0 - 500	0.05 - 0.11	±3	ULP 92 s LP 1.4 s Contínuo 0.75 s

A comunicação com o sensor pode ser realizada através de I²C ou SPI (*Serial Peripheral Interface*). Neste projeto foi priorizado o protocolo série I²C por ser largamente usado e robusto. O sensor possui registos específicos de 8 bits, em que são efetuadas escritas e leituras, de forma a configurar e obter os valores do sensor [85].

4.1.5.2. AMPLIFICADOR MAX9814

Os microfones de eletreto são vulgarmente utilizados em circuitos eletrónicos com o objetivo de capturar e gravar o ambiente. O MAX9814 é um amplificador de um microfone de eletreto com uma resposta em frequências de 20 Hz a 20 kHz (ver Figura 40). Este amplificador possui *Automatic Gain Control* (AGC) que ajusta o ganho de acordo com a tensão de saída, prevenindo o corte da onda de saída quando o ganho de entrada é elevado. Permite ainda, escolher o nível de ganho pretendido, 40 dB, 50 dB e 60 dB, num único pino.

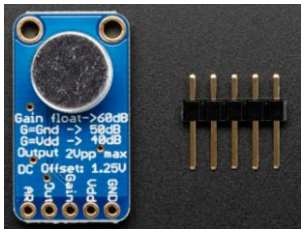


Figura 40 Sensor MAX9814

Numa primeira fase é aplicada uma pré-amplificação LNA (*Low Noise Amplifier*) de 12 dB, posteriormente é usado um segundo amplificador que depende da configuração do AGC (0 dB a 20 dB) e por fim, a seleção de ganho do utilizador (8 dB, 18 dB ou 28dB). Assim, possui um ganho total de 40 dB, 50 dB ou 60 dB. A Figura 41 demonstra os componentes do MAX9814 assim como a cascata de ganho, até à tensão de saída [87].

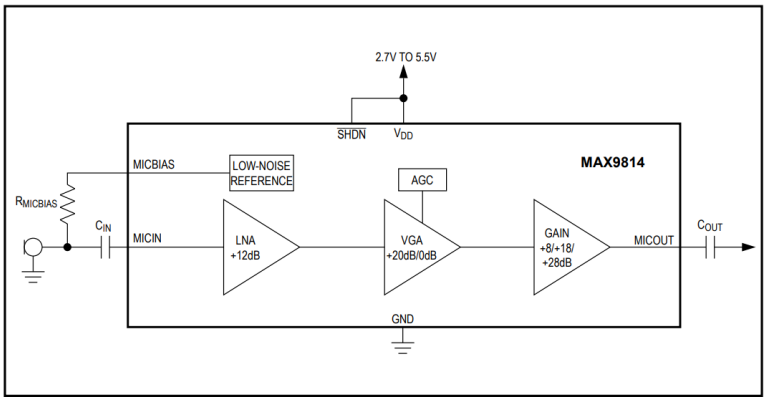


Figura 41 Diagrama MAX9814 [87]

O MAX9814 é utilizado neste projeto pela simples implementação, por ser compacto e pelas características como o controlo do ganho. No sentido de detetar a intensidade sonora da passagem dos veículos e do meio ambiente, constatou-se através de pequenos testes, que se pretende uma resposta em tensão mais linear, o que o AGC não permite. Desta forma, foi desativado o amplificador VGA.

4.1.5.3. SENSOR BH1750FVI

O BH1750FVI é um sensor de luminosidade digital, de 16 bits (1 lx a 65535 lx), com o objetivo de detetar a luminosidade ambiente (ver Figura 42). Este não depende do tipo de fonte luminosa para as medições, como por exemplo lâmpadas ou o sol. Devido ao seu tamanho, destina-se a ser aplicado em *smartphones*, câmaras digitais e LCD (*Liquid Crystal Display*), onde se pretende ajustar a visibilidade dos ecrãs de acordo com a luminosidade ambiente.

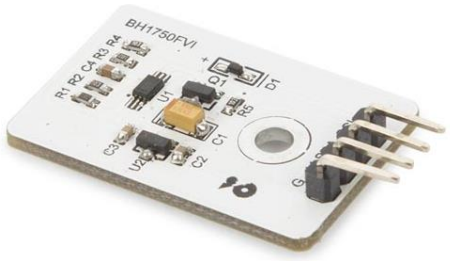


Figura 42 Sensor de luminosidade BH1750FVI [88]

Suporta o protocolo de comunicação série I²C com registos de 8 bits. Estes consistem na ativação do sensor, assim como nos modos de funcionamento disponíveis. Para efetuar as medições podem ser usados três modos, dois de alta resolução (H-Resolution Mode2 e H-Resolution Mode) e um de baixa resolução (L-Resolution Mode) (ver Tabela 14). De acordo com estes modos de funcionamento, existem registos específicos para serem executadas leituras únicas ou leituras contínuas [89].

Tabela 14 Modos de funcionamento do sensor BH1750FVI [89]

Modo de medição	Tempo de medição	Resolução
H-Resolution Mode2	120 ms	0.5 lx
H-Resolution Mode	120 ms	1 lx
L-Resolution Mode	16 ms	4 lx

É recomendada a utilização do modo H-Resolution Mode ou o modo H-Resolution Mode2, devido ao tempo de medição de 120 ms, que permite eliminar ruido de 50 Hz e 60 Hz. Ambos os modos têm resolução suficiente para produzir boas medições em ambientes escuros [89].

O sensor é formado por um fotodíodo que capta a intensidade luminosa do ambiente e transforma-a em corrente elétrica. A corrente gerada pelo fotodíodo, percorre um amplificador de integração, de forma a converter a corrente em tensão. Posteriormente, o sinal atravessa um conversor analógico-digital de 16 bits, que alimenta o bloco lógico onde são realizados os cálculos necessários, de forma a obter um valor de luminosidade congruente. A luminosidade é escrita num registo, de possível leitura, assim como o tempo de medição [89].

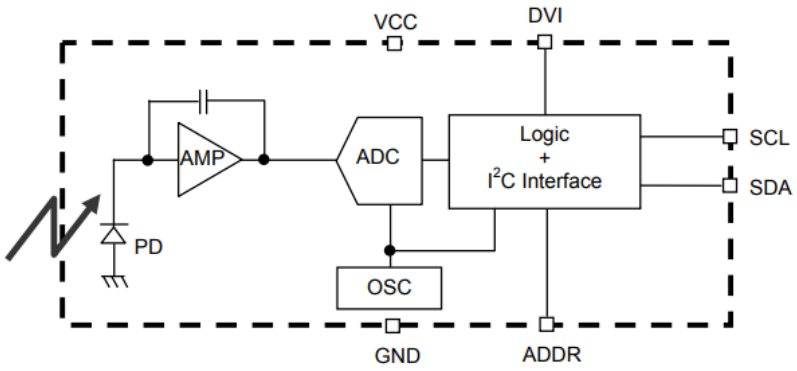


Figura 43 Diagrama do sensor BH1750FVI [89]

4.1.5.4. CONTROLADOR PRINCIPAL E GATEWAY

O controlador principal é o dispositivo que se destina a receber e enviar a informação conforme o protocolo LPWAN selecionado. Como já referido, LoRaWAN é o protocolo que vai ser utilizado e como tal, é necessário optar por hardware que garanta que as comunicações por LoRa sejam bem-sucedidas.

Inicialmente foram analisados, em conjunto com os restantes sujeitos envolvidos no sistema NEXT-Road e a Pavnext, microcontroladores no sentido de optar por um único fabricante a interligar os dois sistemas. Assim, a opção recaiu nos dispositivos da empresa STMicroelectronics que detém uma gama de opções para comunicação LoRa. De acordo com uma análise dos dispositivos, foi optado pelo conjunto P-NUCLEO-LRWAN2 (ver

Figura 44) que possui um *end-node*, g*ateway* e as respetivas antenas, para implementar soluções LoRaWAN.

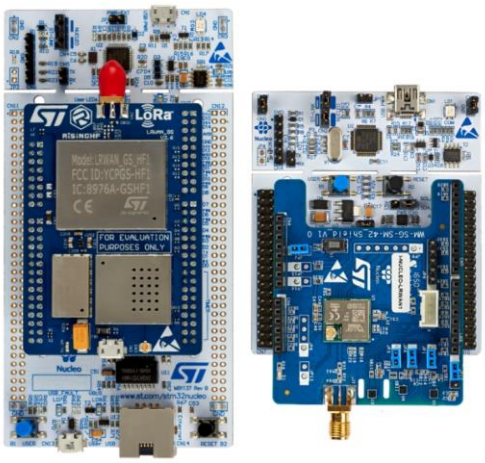


Figura 44 Pack de desenvolvimento P-NUCLEO-LRWAN2 [90]

Este *pack* é uma das opções disponíveis na empresa Pavnext e como tal, foi sujeito a uma análise inicial, de acordo com os seus componentes (ver Tabela 15). Como referido no capítulo 3, a troca de informação por LoRa, em espaço europeu, tem um plano de frequência de 868 MHz (alta frequência) e 433 MHz (baixa frequência). Assim, os transcetores LoRa incluídos no *pack*, permitem comunicações a altas frequências.

Tabela 15 Componentes do pack P-NUCLEO-LRWAN2 [90]

End Node	**Gateway**
• Placa de desenvolvimento STM32 Nucleo-L073RZ que contém o microcontrolador Arm STM32L073RZT6 de baixa potência. • Placa de expansão I-NUCLEO-LRWAN1 de altas frequências (868/915/923 MHz) com um transcetor Semtech SX1272.	• Placa de desenvolvimento STM32 Nucleo-F746ZG que contém o microcontrolador Arm STM32F746ZGT6. • Placa de expansão LRWAN_GS_HF1 de altas frequências (868/915/923 MHz) com um concentrador e transcetor SX1301/SX1257 HF da Semtech. Suporta aplicações de classe A e C do protocolo LoRaWAN. • Possui ainda um encaminhador de pacotes Semtech. Suporta DNS (*Domain Name System*) e NTP (*Network Time Protocol*).

A placa I-NUCLEO-LRWAN1 (ver Figura 45) é fabricada pela empresa USI, em parceria com a empresa STMicroelectronics, para criar um *shield* LPWAN com suporte à tecnologia LoRa. A sua arquitetura consiste num microcontrolador STM32L052T8Y6, que comunica com o transcetor SX1272 da Semtech, onde ocorre a modulação do sinal LoRa até ser emitido por uma antena. Este *shield* pode ser controlado a partir de outro microcontrolador que, ocorrendo troca de informação por USART (*Universal Synchronous Asynchronous Receiver Transmitter*), consegue controlar o seu comportamento e a informação enviada [91].

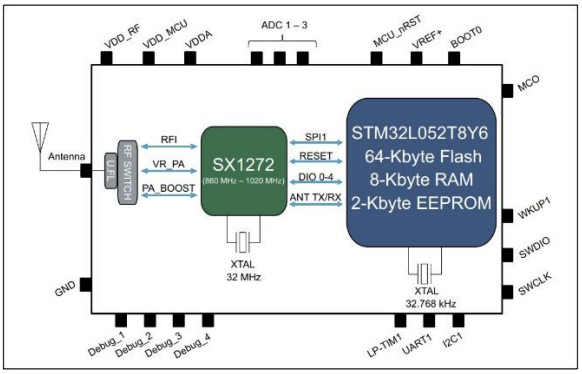


Figura 45 Diagrama da placa I-NUCLEO-LRWAN1 [91]

A *gateway* é constituída por uma placa de expansão LRWAN_GS_HF1 (ver Figura 46), desenvolvida pela empresa RisingHF, que é controlada pelo microcontrolador STM32F746ZGT6, através do protocolo série SPI. É composta por: um concentrador LoRa SX1301, destinado a processar os sinais digitais na banda de frequências ISM; um transcetor SX1257 de alta frequência e dois filtros SAW (*Surface Acoustic Wave*) de forma a obter uma largura de banda de 868 MHz ou 915 MHz [92], [93].

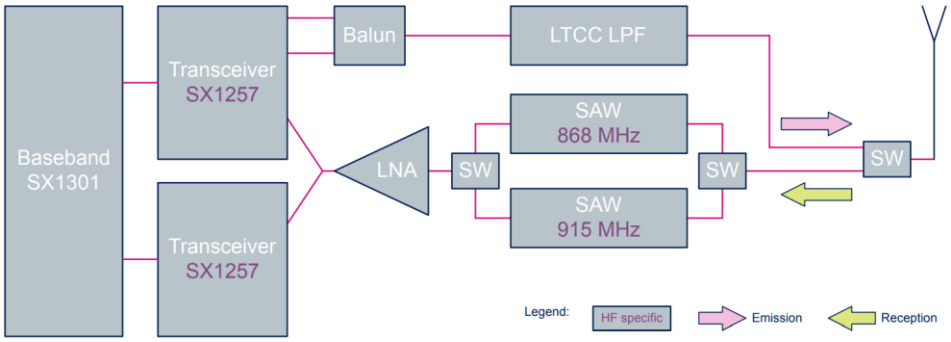


Figura 46 Diagrama da placa LRWAN_GS_HF1 [93]

No início da etapa de desenvolvimento de *firmware*, este era o hardware a ser utilizado e testado. No entanto, devido à opção de ser implementada comunicação pelo protocolo CAN FD, a partir do sistema NEXT-Road, houve a necessidade de alterar o microcontrolador do controlador principal. O STM32L073RZT6 não possui funcionalidade para comunicação por CAN FD, devido ao recente desenvolvimento deste protocolo. Assim, optou-se pela utilização da placa de desenvolvimento Nucleo-G474RE, com o microcontrolador STM32G474RE que, para além de conter um maior número de interfaces de comunicação, permite configurar e realizar trocas de informação por CAN FD.

Ambas as placas de desenvolvimento pertencem à gama STM32 Nucleo-64 da STMicroelectronics, possuindo em termos físicos, as mesmas dimensões e configuração dos pinos. Portanto, não existiu qualquer problema de compatibilidade do Nucleo-G474RE com a placa I-NUCLEO-LRWAN1.

4.2. ORGANIZAÇÃO DA INFORMAÇÃO

Com o objetivo de apresentar informação de forma útil e intuitiva, são efetuadas filtragens aos dados que são adquiridos, desde o controlador principal até à sua apresentação. Assim, o controlador principal é a primeira instância de filtragem em que existe um cuidado na organização e na quantidade de bytes que são enviados. A *gateway* ao receber uma mensagem do controlador principal, desmodula o sinal recebido e encaminha a sua informação para um servidor LoRaWAN, através de uma ligação *ethernet* a um *router*.

O servidor LoRaWAN tem a capacidade de descodificar e exportar a informação que recebe através de integrações. As integrações fazem parte da aplicação do servidor e servem para gerir e direcionar a informação, normalmente para bases de dados ou serviços em *cloud*. A informação extraída do servidor LoRaWAN é sujeita a uma nova filtragem, seguindo a organização pretendida para a base de dados. A base de dados retém a informação, no formato de séries temporais, onde os dados estão prontos a ser cruzados e visualizados.

4.2.1. SERVIDOR LORAWAN

Atualmente existem diversos ecossistemas, criados por empresas que fornecem ferramentas e servidores especializados no protocolo LoRaWAN. Funcionam de forma semelhante, em que os pacotes enviados pelo controlador principal são recebidos por uma *gateway* (ponte entre os protocolos rádio e a internet). LoRaWAN é um protocolo *non-IP*, o

que obriga a existir um processamento antes de chegar ao utilizador. Assim, estes servidores LoRaWAN, situam-se entre as *gateways* e as aplicações, atuando como *routers*.

Do vasto número de operadores, membros da LoRa Alliance, foi escolhida a utilização dos servidores da empresa The Things Network (TTN), que disponibiliza um conjunto de ferramentas para a gestão e reencaminhamento dos dados.

A empresa TTN possui a The Things Stack (TTS) que corresponde ao conjunto de servidores com base em código *open-source*. São responsáveis por gerir a rede LoRaWAN e receber o tráfico IP das *gateways*. Atualmente encontra-se na versão 3 (ver Figura 47) e onde é possível gerir as aplicações via hardware ou *cloud*.

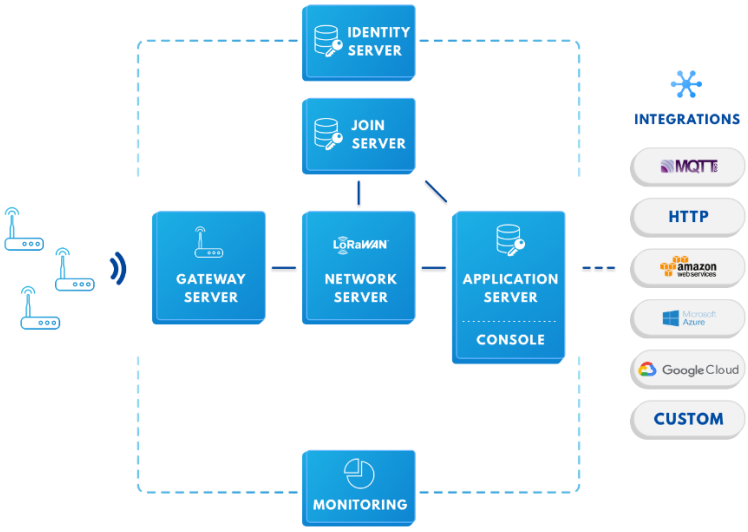


Figura 47 Arquitetura da terceira versão da TTS

É necessário a criação de aplicações (*Application ID*) em que são registados os *end nodes* e as *gateways*, para permitir que o servidor possa identificar a aplicação e comunicar com os dispositivos que pertencem a essa aplicação. Após a identificação dos dispositivos no servidor, é possível enviar mensagens LoRa do controlador principal até à *gateway* e estas são apresentadas no servidor através da *Console*, aplicação web que permite a gestão dos dispositivos e do tráfego dos dados.

A TTN não impõe aos utilizadores qualquer tipo de hardware ou de integrações. Pretende ser um servidor intermédio para aplicações LoRaWAN, possuindo planos empresariais, com suporte e instalação da melhor solução para o cliente.

4.2.2. NODE-RED, INFLUXDB E GRAFANA

A informação que chega no servidor LoRaWAN é reencaminhada para o Node-Red, através da integração MQTT (*Message Queuing Telemetry Transport*). Node-Red é uma ferramenta de programação em blocos (*nodes*), para interligar diversos dispositivos, serviços web e API. Foi criado em 2013 como projeto *open-source*, com o objetivo de simplificar processos, atualmente faz parte da OpenJS Foundation. Consiste num *runtime* Nodejs, onde é possível abrir uma interface com o utilizador, a partir do *browser*, para fácil edição e configuração de *nodes* em linguagem JavaScript.

Cada *node* tem uma função específica, podendo ser consultados e instalados através da biblioteca Node-Red. Assim, os *uplinks* detetados no servidor são recebidos através do *node* MQTT, para posteriormente inicializar a organização dos dados.

Das diversas opções possíveis, descritas no capítulo 2, optou-se pelo servidor de base de dados InfluxDB. Este preenche os requisitos para a retenção de dados de séries temporais. Portanto, a introdução das séries temporais na base de dados é realizada através do Node-Red, através de nodes específicos. A Figura 48 apresenta a divisão dos dados em *measurements*, considerando a sua origem e importância.

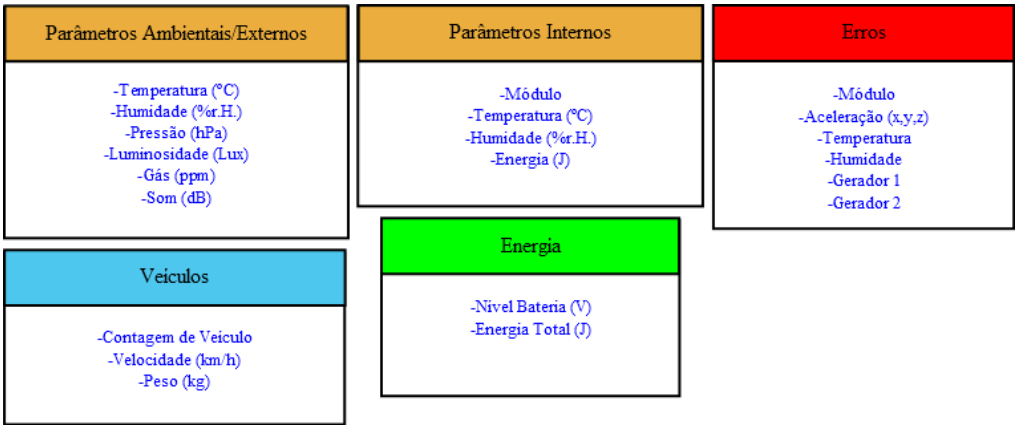


Figura 48 Organização da base de dados InfluxDB

Após considerar toda a informação que se pretende armazenar, foi decidido que a base de dados InfluxDB, teria os seguintes *measurements*:

- Parâmetros ambientais/externos – valores de sensores destinados a captar o estado ambiental, onde o sistema NEXT-Road está inserido;

- Parâmetros Internos – valores de sensores que pertencem ao sistema NEXT-Road;

- Veículos – informações adicionais sobre os veículos, obtidas através de sensores;

- Energia – valores de energia gerada e estado de alimentação do sistema;

- Erros – tabela destinada a receber possíveis sinais de erros do sistema NEXT-Road, sensores ambientais e do controlador principal.

No sentido de conseguir analisar e monitorizar os dados introduzidos na base de dados, foi escolhida a ferramenta de monitorização Grafana (abordada no capítulo 2). Através de um *plugin*, é possível optar por uma fonte de informação InfluxDB, onde a Grafana lê os campos da base de dados e executa *queries*, para apresentá-los de forma gráfica.

5. IMPLEMENTAÇÃO

Durante a implementação do projeto, foram considerados os objetivos iniciais, assim como a arquitetura do sistema, descritos nos capítulos 1 e 4, respetivamente. Portanto, este capítulo aborda as etapas percorridas, desde a receção da informação, até à sua visualização. Durante o desenvolvimento e teste do sistema, as etapas percorridas sofreram pequenas alterações de acordo com as possibilidades. Estas apresentam a seguinte ordem:

- Descoberta das funcionalidades do hardware a ser utilizado, assim como o respetivo ambiente de desenvolvimento;
- Comunicação com os sensores ambientais em *breadboard*;
- Teste de comunicação LoRa entre o controlador principal e a *gateway*, através do envio de valores obtidos dos sensores ambientais;
- Integração do Node-Red e divisão dos dados, de acordo com a organização pretendida para a base de dados;
- Introdução da informação nos respetivos *measurements;*
- Acesso à informação a partir da Grafana para o desenvolvimento de *dashboards*.
- Retificação e melhoria do sistema, de forma a receber mensagens através dos protocolos CAN FD e RS485;
- Teste em laboratório e no terreno.

Numa primeira fase, a atenção estava em ser totalmente independente do sistema NEXT-Road. Pretendia-se criar uma base de ligação entre a receção da informação e a sua apresentação, antes de serem adicionados os protocolos usados no resto do sistema, que ainda estariam em fase de desenvolvimento. Desta forma, a maioria da implementação decorreu com dados dos sensores ambientais, assim como valores fictícios. Estes serviram com o propósito de prever os dados recebidos e a sua organização, durante o seu percurso

no sistema. A referida abordagem permitiu, que a implementação destes protocolos se tornasse uma tarefa rápida e sem demais complicações.

5.1. HARDWARE

O *hardware* usado na implementação do projeto, segue os componentes descritos no capítulo 4. Aqui são abordados os desenvolvimentos e configurações, para os sensores, protocolos de comunicação e controlador principal. Após a devida validação em *breadboard*, foi realizada uma PCB (*Printed Circuit Board*), de forma a tornar o controlador principal mais robusto, aglomerando todos os componentes.

5.1.1. CONTROLADOR PRINCIPAL

O *hardware* que compõem o controlador principal, tem como base os componentes descritos na arquitetura do sistema, nomeadamente: placa de desenvolvimento Nucleo-G474RE; sensores ambientais e placa de expansão I-NUCLEO-LRWAN1. Foi desenvolvida uma PCB com o objetivo de fornecer suporte aos sensores, transcetores dos protocolos CAN FD e RS485, assim como alimentação para todos os componentes.

Após um estudo da placa de desenvolvimento Nucleo-G474RE, esta não sofreu qualquer tipo de modificação, pelo que a configuração de fábrica permite, sem qualquer obstáculo, o desenvolvimento deste projeto. O único componente que desperta atenção é o *jumper* (JP5), que seleciona a entrada de alimentação na placa. Neste sentido, existem duas configurações que são usadas: momentos de programação e *debug*, com JP5 em 5VSTLK; alimentação com bateria de 9 V a 12 V, com JP5 em 5V_VIN.

A placa de expansão I-NUCLEO-LRWAN1, de igual forma, não necessitou de qualquer alteração física. Está preparado com conectores Arduino, de forma a acentar nos conectores CN8 e CN9 do Nucleo-G474RE, fornecendo a alimentação necessária. Possui um conector SMA (*SubMiniature version A*) para a instalação da respetiva antena de 50 ohm. A comunicação com o microcontrolador ocorre através de USART nos pinos 35 (USART1-RX) e 37 (USART1-TX) do conector CN10 (ver Figura 49) .

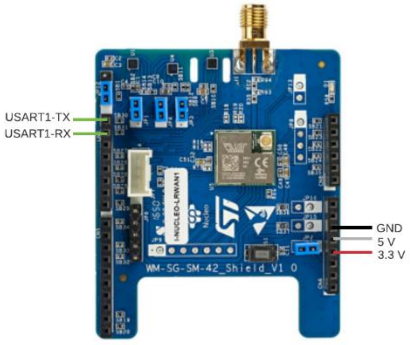


Figura 49 Alimentação e comunicação da placa I-NUCLEO-LRWAN1. Adaptado de [91]

O acréscimo dos sensores ambientais, implica uma pesquisa dos protocolos de comunicação série que podem ser utilizados. O sensor BME680 pode ser incluido no sistema a partir do protocolo I²C ou SPI, enquanto o sensor BH1750FVI, só suporta trocas de informação por I²C. Optou-se por usar o protocolo I²C, sempre que possível, independentemente do sensor, pelas suas vantagens face a outros protocolos série. Assim, a Figura 50 demonstra as ligações efetuadas entre o microcontrolador e os sensores BME680 e BH1750FVI.

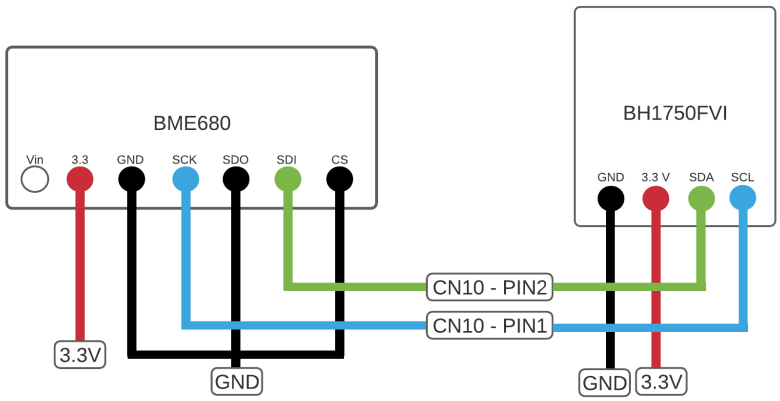


Figura 50 Ligações dos sensores BME680 e BH1750FVI ao controlador principal

O amplificador MAX9814 produz à saída, uma diferença de potencial de acordo com a intensidade sonora que atravessa o microfone. No sentido de detetar esta tensão é usado um conversor analógico-digital, assim como eletrónica à saída do amplificador. De acordo com a documentação, a saída do amplificador apresenta um *offset* DC de 1,23 V, para eliminar este *offset* é usado um condensador (C_{out}) e uma resistência(R_L), formando um filtro passa-alto. Este permite a passagem de frequências acima de 1 kHz, frequência na qual a

influência do ganho estabiliza. O cálculo da capacitância e da resistência para uma frequência de 1 kHz, segue a equação,

$$F_{3dB} = \frac{1}{2\pi * R_L * C_{out}}. \quad (1)$$

Após o condensador, a tensão nesse ponto é a tensão de entrada no conversor, assim conseguindo utilizar os valores como leituras. Está incluido um *jumper* que serve para selecionar o nível de ganho desejado (3,3 V, GND ou em aberto). Referir ainda, que o AGC foi desativado, de forma não ter efeito nas medições de tensão obtidas. A Figura 51 demonstra as ligações efetuadas para a inclusão do amplificador no sistema.

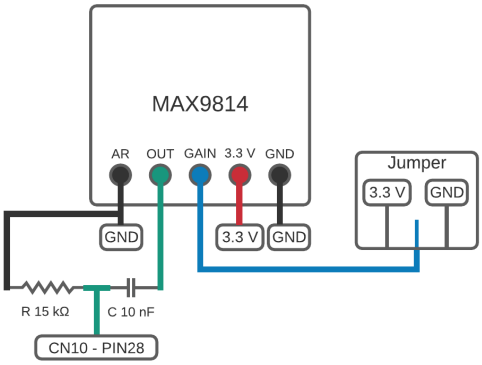


Figura 51 Ligações do amplificador MAX9814 ao controlador principal

5.1.2. DESENVOLVIMENTO DA PCB

O desenvolvimento da PCB decorre numa altura em que já estava validada a informação dos sensores, assim como a comunicação LoRa e consequente envio de informação até à base de dados e Grafana. No entanto, testes com os protocolos do sistema NEXT-Road, estariam ainda por realizar, apesar de se conhecer aqueles que iriam ser utilizados. Assim, de acordo com as informações e *hardware* necessário para o sucesso das comunicações, a PCB foi realizada e testada, não necessitando de qualquer ajuste.

Para o desenvolvimento da PCB foi utilizado o software Altium Designer (versão 17) e este teve os seguintes passos:

1. Reconhecimento e planeamento de todo o hardware e das ligações necessárias;

2. Criação de um esquema com todos os componentes;

3. Criação do formato da PCB e das ligações entre os elementos;

4. Adição de bibliotecas e corpos 3D que correspondem aos componentes usados;

5. Aplicação de uma camada de GND na parte superior da PCB.

Além das ligações dos sensores (descritas no ponto 5.1.1.), a PCB está preparada para trocar informações através de CAN FD e RS485. Para tal, são necessários os respetivos transcetores assim como algumas resistências. A Figura 52 demonstra as conexões entre os diferentes componentes, desde os conetores, para o sistema NEXT-Road, até aos pinos correspondentes.

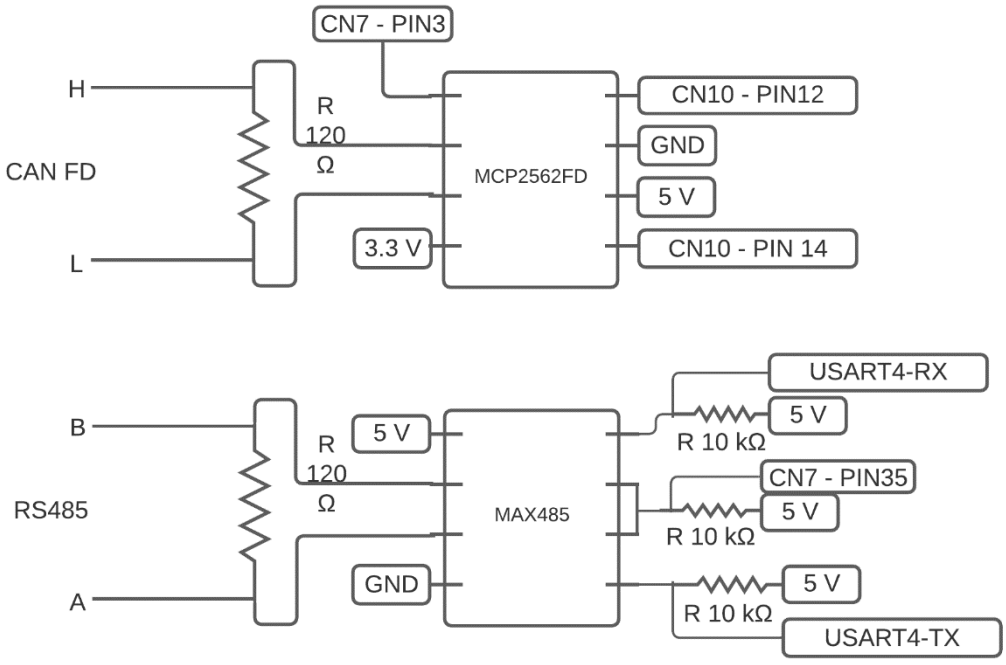


Figura 52 Ligações elétricas para os protocolos CAN FD e RS485

Com o objetivo de alimentar o controlador principal com uma bateria de 12 V, foi ainda acrescentado um conetor ligado ao pino V_{in}, assim como ligação a um ADC, que atribui uma percentagem à tensão lida no intervalo de 0 V a 3,3V. Considerando as ligações dos sensores, protocolos de comunicação e alimentação, a PCB do controlador principal tem o aspeto apresentado na Figura 53.

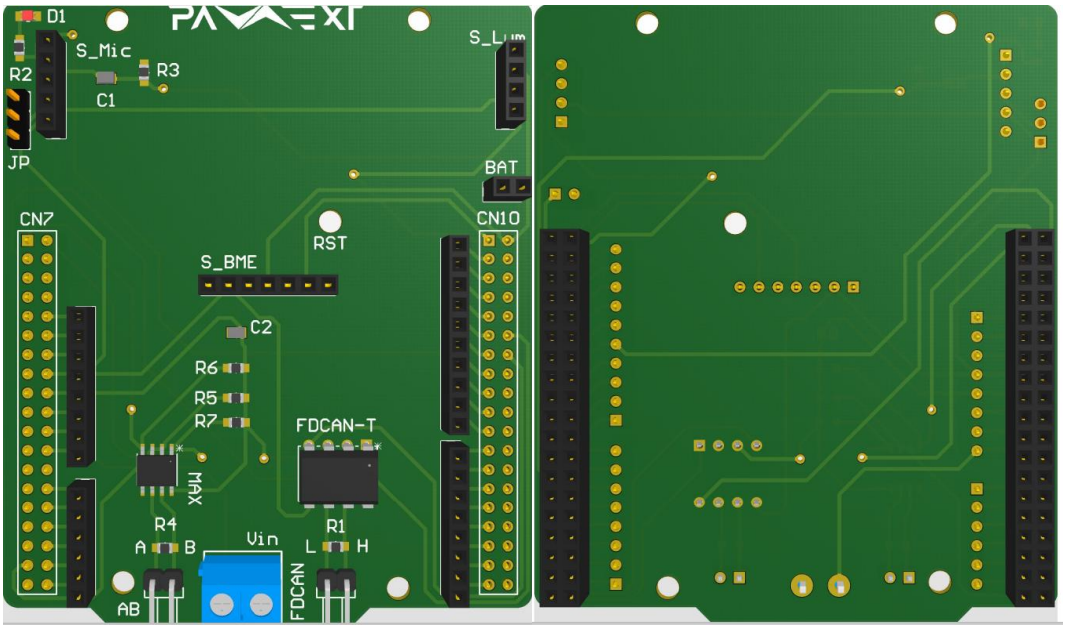


Figura 53 *Layout* final da PCB do controlador principal

Após receber a PCB do fabricante, foram soldados os conetores, os transcetores e os componentes eletrónicos. Posteriormente, foi instalada no Nucleo-G474RE para testar possíveis erros sou defeitos nas ligações. Todo o sistema do controlador principal funciona com a PCB, pelo que esta versão não foi alterada, sendo usada nos testes que decorreram. Assim, a sua instalação consiste no Nucleo-G474RE na parte inferior, com a PCB e a placa de expansão LoRa, em cascata, juntamente com os sensores. A Figura 54 mostra o controlador principal com todos os seus elementos.

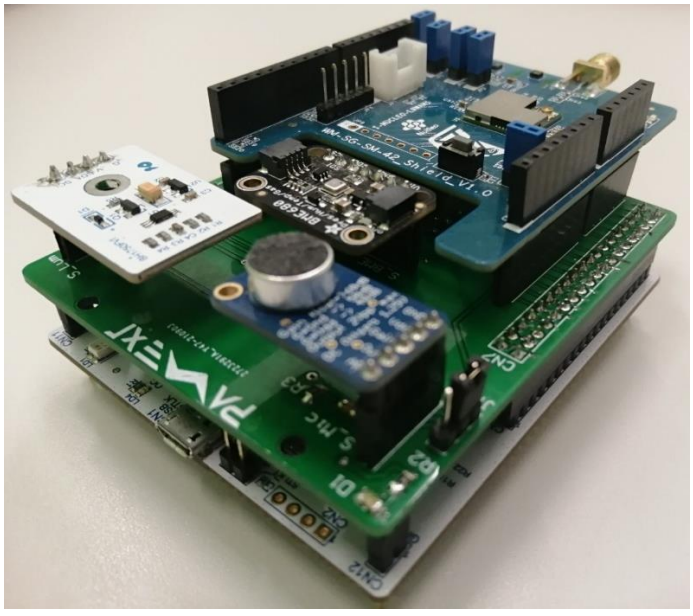


Figura 54 Controlador principal

5.1.3. GATEWAY

A *gateway* é composta por um Nucleo-F746ZG e a placa de expansão LRWAN_GS_HF1. Este conjunto é instalado a alguma distância do controlador principal, dentro dos limites das comunicações LoRa, particularmente em instalações de interesse, com acesso a um router ligado à internet. Possibilita a configuração de parâmetros necessários para o encaminhamento dos pacotes, como: o endereço DNS; endereço IP da *gateway*; endereço MAC e ID da *gateway*; servidor privado ou público; plano de frequência; *baud rate*; entre outros. Estas configurações são conseguidas a partir de uma interface de comandos AT através da porta virtual do ST-LINK (programador da placa de desenvolvimento).

A placa de expansão possui um circuito interno que alimenta o conjunto que constitui a *gateway*, com uma tensão de 5 V. O conetor CN1 alimenta a placa de expansão e através do pino V_{in}, fornece energia para o Nucleo-G474RE. Portanto, é recomendado que a alimentação da *gateway* seja realizada através de um transformador ligado a uma tomada e um cabo micro USB (*Universal Serial Bus*), ligado ao conetor CN1 (ver Figura 55).

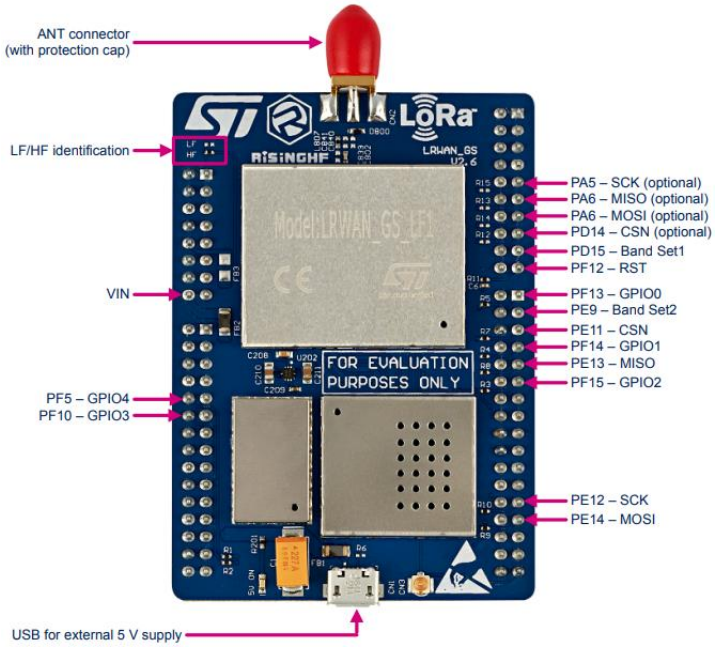


Figura 55 Placa de expansão LRWAN_GS_HF1 [90]

5.2. FIRMWARE

O desenvolvimento do *firmware* está presente durante todo o progresso do projeto. Essencialmente contido no controlador principal, na receção de informação, organização e envio de pacotes por LoRa. Na *gateway*, a empresa STMicroelectronics fornece o *firmware* baseado no encaminhador de pacotes Semtech, que permite a conexão com servidores LoRaWAN através do protocolo Semtech UDP (*User Datagram Protocol*). Este protocolo possibilita a troca de informação entre a *gateway* e o servidor, através de pseudo-JSON. O método de reconfiguração da *gateway*, segue o referido no ponto 5.1.3.

5.2.1. FIRMWARE DO CONTROLADOR PRINCIPAL

Numa primeira instância é imperativa a análise de requisitos que o controlador principal deve possuir, em termos de *firmware*, de forma que o seu funcionamento relacione todos os periféricos descritos no ponto 5.1. Portanto, o *firmware* desenvolvido para o controlador principal conta com um conjunto de bibliotecas realizadas e organizadas por função/objetivo. As intenções para o *firmware* têm a seguinte ordem:

- Inicialização de interfaces de comunicação, periféricos e GPIOs (*General Purpose Input/Output*) do microcontrolador;
- Definição de estruturas para a retenção dos dados;
- Início de conexão com o servidor LoRaWAN através da *gateway*;
- Leitura de dados dos sensores ambientais;
- Verificação de receção de mensagens por CAN FD ou RS485;
- Envio de pacotes LoRa ciclicamente, de acordo com a sua relevância;
- Verificação de possíveis erros de *firmware* do sistema.

No sentido de programar o microcontrolador com o *firmware*, foi optada pela utilização da plataforma STM32CubeIDE da STMicroelectronics. Esta possibilita o desenvolvimento de código em C e C++, a sua compilação (GCC) e opções de *debug* para dispositivos STM32 (GDB) (ver Figura 56). Baseia-se no *framework* Eclipse e possui

funcionalidades como geração de código automático e configuração intuitiva do microcontrolador.

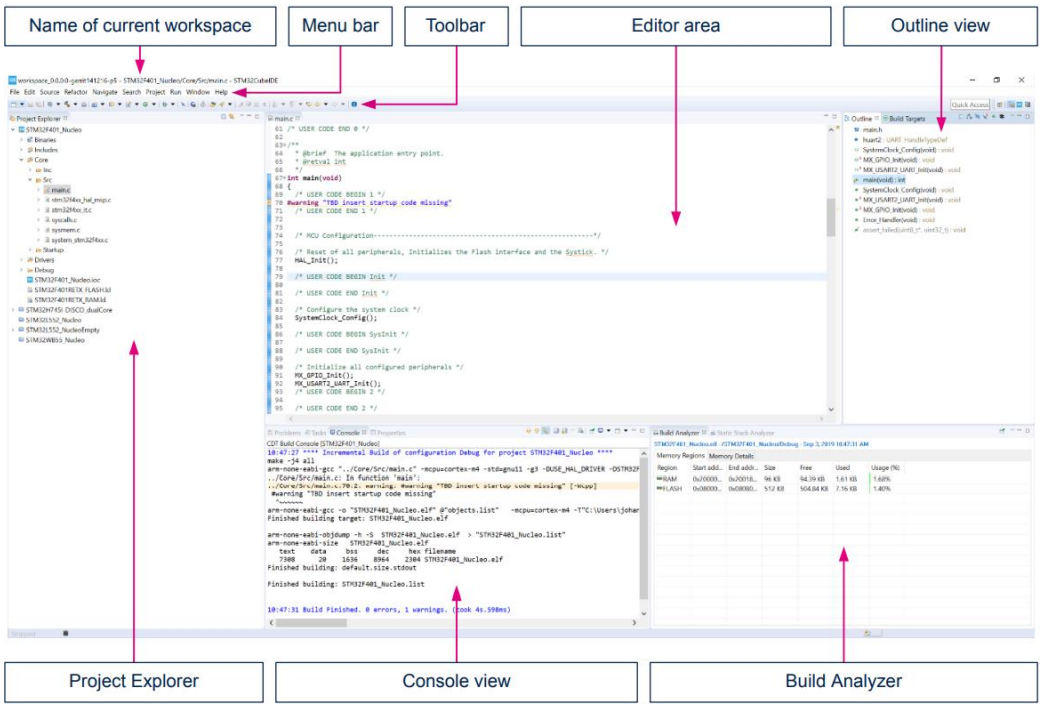


Figura 56 Exemplo da organização do STM32CubeIDE [94]

No intuito de conseguir cumprir os objetivos descritos para o controlador principal, um conjunto de periféricos são inicializados no início deste *firmware*. A Tabela 16 apresenta os diferentes periféricos assim como a sua função.

Com o sentido de organizar o *firmware*, as variáveis globais, assim como as estruturas, funções e *defines*, foram declarados no ficheiro main.h. O ato de incluir este ficheiro nas diversas bibliotecas (ficheiros do tipo .c e .h), permite que o compilador reconheça as variáveis e que as possa alterar, se for o caso. As bibliotecas presentes no *firmware* consistem essencialmente, em aplicações de sensores, configurações LoRa, formação de pacotes LoRa e comunicação CAN FD e RS485.

Tabela 16 Inicialização de periféricos e a sua função

Periféricos	Função
I²C3	Utilização do protocolo I²C para implementar os sensores BME680 e BH1750FVI.
FDCAN1	Protocolo CAN FD para a troca de mensagens entre o sistema NEXT-Road e o controlador principal.
USART1	Troca de informação entre o microcontrolador e a placa de expansão I-NUCLEO-LRWAN1.
USART4	Receção das mensagens com o protocolo RS485 entre o sistema NEXT-Road e o controlador principal.
ADC1	Conversão dos valores de tensão do amplificador MAX9814.
ADC3	Conversão dos valores de tensão da bateria para percentagem.
TIM16	Temporizador que define o ciclo de envio de pacotes LoRa.
TIM20	Temporizador que define a verificação da receção de mensagens por RS485.

As estruturas são conjuntos de variáveis, de acordo com a divisão que é pretendida para o envio dos pacotes, assim como na posterior organização da base de dados. Desta forma, existem cinco estruturas definidas que são atualizadas de acordo com a receção da informação. O seu conteúdo é posteriormente usado para a formação das mensagens LoRa.

```
typedef  struct sensores_amb
             //PARÂMETROS AMBIENTAIS
{
  uint8_t id[1];              //VALOR: 0x20
  uint8_t luminosidade[2];    //LUZ
  uint8_t temperatura[1];     //TEMPERATURA
  uint8_t humidade[1];        //HUMIDADE
  uint8_t iaq [1];            //GAS
```

```
  uint8_t voc [2];            //GAS
  uint8_t co2 [2];            //GAS
  uint8_t som [1];            //SOM
  uint8_t pressao[1];         //PRESSAO
} sensores_amb;
struct sensores_amb sensor_amb;
```

5.2.1.1. FIRMWARE DOS SENSORES

Os *firmwares* desenvolvidos para a troca de comunicação entre os sensores e o microcontrolador, têm uma sequência de configurações (escritas e leituras de registos) até a informação ser obtida. Desta forma, considerando o sensor BH1750FVI, que converte a intensidade luminosa num valor digital, a suas etapas estão apresentadas no fluxograma da Figura 57 .

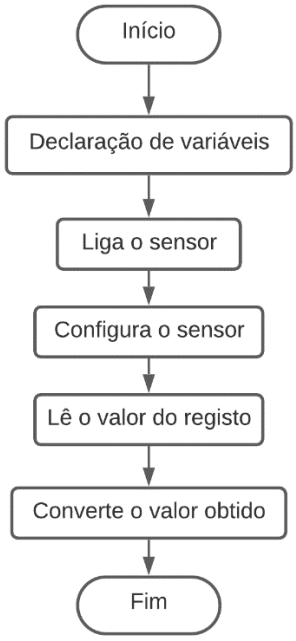


Figura 57 Fluxograma da biblioteca do sensor BH1750FVI

Após as necessárias conversões recomendadas na documentação do sensor, é possível obter um valor aproximado da intensidade luminosa do presente local. Usando o mesmo periférico, o sensor BME680 possui etapas mais complexas e conversões extensas, pelo que o *firmware* deste sensor teve um período de desenvolvimento bastante superior.

Como descrito no capítulo 4, este sensor mede uma grande variedade de parâmetros e como tal existem dependências de valores entre as medições. A Figura 58 apresenta a

sequência de configurações até à obtenção dos principais valores e o processo cíclico do algoritmo BSEC (*Bosch Sensortec Environmental Cluster*).

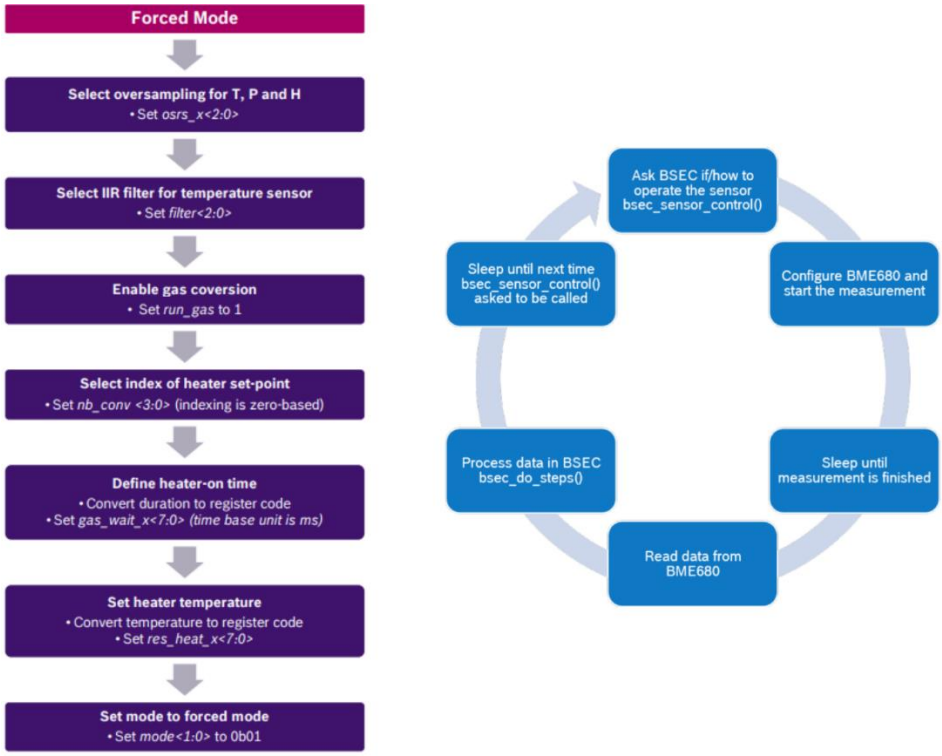


Figura 58 Processo de medição e ciclo do algoritmo BSEC [85], [95]

A Bosch, empresa que fabrica o BME680, contém o algoritmo BSEC para a determinação de IAQ, VOC e B-CO2, assim como calibrações dos restantes sensores, com o passar do tempo. Sem a inclusão deste algoritmo, só é possível obter valores de temperatura, humidade, pressão e um valor de gás (sem qualquer significado explicito). Assim, de forma a aplicar este algoritmo a Bosch fornece um conjunto de bibliotecas (arquivo de bibliotecas estáticas) e documentação para efetuar as conversões necessárias.

```
void mybsec_loop()
{
   time_stamp = HAL_GetTick();
   time_stamp= time_stamp*1000000;
//TimeStamp em ns
  bsec_library_return_t bsec_status = BSEC_OK;

   if ((time_stamp >= sensor_settings.next_call))
   //Quando chegar ao valor de next_call
{
   bsec_sensor_control(time_stamp,&sensor_settings
);
//Preenche a estrutura sensor_settings com as
configurações a escritas nos registos do sensor
   bme_trigger();
```

```
//Aplica a configuração e ativa uma medição
   bme_read_all_sensors();
//Lê os registos
   num_bsec_inputs=0;
   fill_inputs(time_stamp, &bme_values, inputs,
&num_bsec_inputs);
//Preenche a estrutura inputs que vão ser
processados pelo algoritmo
   process_data(inputs, num_bsec_inputs);
//Preenche a estrutura outputs com os resultados
   }
}
```

5.2.1.2. FIRMWARE DOS PROTOCOLOS DE COMUNICAÇÃO CAN FD E RS485

No caso das bibliotecas que permitem a receção da informação do sistema NEXT-Road, estas foram desenvolvidas pelos sujeitos que estiveram com esse projeto. Assim, sendo essenciais para a receção de informação, estão presentes no controlador principal. Em ambas, após a deteção da mensagem, é preenchida uma estrutura da biblioteca, a informação assim fica retida temporariamente até surgir uma nova mensagem.

Em cada biblioteca, foram adicionadas funções permitindo o preenchimento das estruturas do controlador principal que, posteriormente formam os pacotes LoRa. Estas funções adequam e organizam a informação de acordo com os pacotes que se pretende enviar. Portanto, os dados ficam retidos nas estruturas até serem enviadas pelo controlador principal, ou até que existam novas mensagens.

5.2.1.3. FIMWARE DE CONFIGURAÇÃO DE COMUNICAÇÃO LORA E ENVIO DE MENSAGENS

A comunicação entre a placa de expansão e o microcontrolador, acontece através de comandos AT enviados por USART. A documentação destes comandos é extensa e apresenta exemplos para cada comando AT enviado. Após uma análise cuidada dos comandos necessários, foram selecionados os mais importantes para que a conexão com o servidor ocorra sem qualquer problema. Desta forma a Tabela 17 exibe os comandos AT usados no decorrer da configuração da placa de extensão, assim como no resto do *firmware*.

Tabela 17 Comandos AT usados no *firmware*

Comando AT	Função
AT	Comando que verifica o estado de conexão com o microcontrolador. A resposta a este comando indica que a placa de expansão está pronta a receber comandos AT.
AT+BAND=0	Seleciona a banda de frequências a que as mensagens vão ser enviadas. Neste caso o valor 0 corresponde à banda EU868.
At+DR=3	Define o *data rate*/*spreading factor* da placa, assim o valor 3 corresponde a uma largura de banda de 125 kHz e um *spreading factor* de 9 (BW=125 kHz / SF=9).
AT+RX2DR=3	Configura o *data rate* da segunda janela de receção (EU868- BW=125 kHz / SF=9).
AT+RX1DT=5000	Configura o *delay* da primeira janela, a partir do envio de uma mensagem. Neste caso, 5000 milissegundos o que corresponde a 5 segundos.
AT+RX2Dt=6000	Configura o *delay* da segunda janela, pelo que 6000 milissegundos correspondem a um *delay* de 6 segundos.
AT+JOIN=1	Envia um pedido de ativação (*JOIN*) ao servidor através de LoRa. Existe a opção 1 para ativação por OTAA e 0 por ABP.
AT+JSTA	Verifica a conexão entre o dispositivo e o servidor. A resposta "1" significa que ainda se encontra conectado, enquanto "0" significa que perdeu conexão ou não foi conectado.
AT+SEND	Comando que envia a mensagem LoRa para a placa de expansão.

Para uma prática mais acessível e clara, estes comandos são definidos em macros e usados no decorrer do *firmware*, contendo uma *string* do comando completo.

```
#define AT              "AT\r\n"
#define RESET_          "ATZ\r\n"
#define DEVEUI          "AT+EUI\r\n"
#define APPEUI           "AT+APPEUI\r\n"
#define DR              "AT+DR=3\r\n"
#define BAND            "AT+BAND=0\r\n"
#define CLASS           "AT+CLASS=0\r\n"
#define DC              "AT+DC=0\r\n"
#define NTYP            "AT+NTYP=0\r\n"
#define ADR             "AT+ADR=0\r\n"
#define RX1DT           "AT+RX1DT=5000\r"
#define RX2DT           "AT+RX2DT=6000\r\n"
#define JRX1DT          "AT+JRX1DT\r\n"
```

```
#define JRX2DT            "AT+JRX2DT\r\n”
#define RX2DR             "AT+RX2DR=3\r\n"
#define JOIN              "AT+JOIN=1\r\n"
#define SEND              "AT+SEND\r\n"
#define JSTA              "AT+JSTA\r\n"
```

Na biblioteca de configuração de comunicação LoRa, estão presentes funções que utilizam estes comandos (envio e receção por USART), assim como a rotina de conexão ao servidor LoRaWAN. Estas funções, exceto a de escrita e leitura, são executadas sequencialmente assim que o microcontrolador é alimentado.

Considerando que a função mais importante do controlador principal é o envio de informação por LoRa, o microcontrolador não executará outras tarefas se não estiver conectado ao servidor. Desta forma, o controlador principal está preparado para enviar pedidos de ativação assim que é ligado ou se perder conexão no decorrer do *firmware*. O seguinte fluxograma (ver Figura 59) identifica as etapas de configuração e conexão.

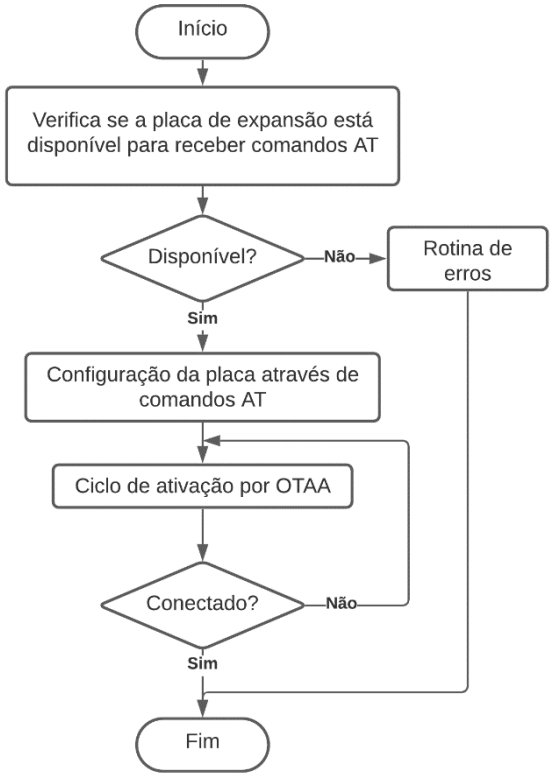


Figura 59 Fluxograma da configuração para comunicação LoRa

A biblioteca de formação de pacotes LoRa contém cinco funções principais, que correspondem à organização das estruturas do controlador principal. Os pacotes para serem

enviados pela placa de expansão têm de possuir três elementos: a porta a qual a informação vai ser enviada (de 1 a 223), o *payload* em formato hexadecimal e uma opção de confirmação da receção do pacote (*acknowledge*). Portanto, as mensagens enviadas por USART para a placa de expansão têm que corresponder a este formato.

```
AT+SEND=2,DE23912F4929194119A4,0
```

De forma mais intuitiva, o exemplo anterior indica o envio de dez bytes em formato hexadecimal como *payload*. Seguindo o formato descrito, o número "2" representa o porto para qual a mensagem é enviada, enquanto que o número "0" indica a não confirmação por parte do servidor. É possível pedir a confirmação de mensagens trocando o valor "0" para "1". Cada uma das funções nesta biblioteca segue passos semelhantes para a formação do respetivo pacote:

- Define o tamanho total do *payload* a ser enviado através das variáveis presentes nas estruturas;
- Copia o conteúdo das variáveis da estrutura para um *buffer*, segundo uma ordem anteriormente ponderada (*payload*).
- Entra num ciclo para formar a mensagem com o formato referido. Um novo *buffer* é preenchido com o cabeçalho "AT+SEND=2,", com o *payload* e com a terminação ",0".
- É enviado o conteúdo do *buffer* para a placa de expansão. Esta prossegue para a modulação da mensagem e o respetivo envio para o servidor LoRaWAN.

```
for(int i=0;i<sizeof(pacote);i++)
{
        if (i==0)
        {     sprintf((char*)buf_send,
        "AT+SEND=2,%02x",pacote[i]);
        }else if (i<(sizeof(pacote)-1))
        {
sprintf((char*)buf_send,"%s%02x",buf_send,pacote[
i]);
        }else if (i==(sizeof(pacote)-1))
        {
sprintf((char*)buf_send,"%s%02x,%u\r\n",buf_send,
pacote[i],conf_msg);
        }
}
```

5.2.1.4. CICLO PRINCIPAL

O ciclo principal aglomera todas as funcionalidades do *firmware*. Portanto, este percorre continuamente, caso não existam erros nas comunicações com a *gateway*, protocolos de comunicação ou placa de expansão. A Figura 60 exibe, em fluxograma, o funcionamento do ciclo principal.

O ciclo principal pode ser decomposto em etapas de acordo com os objetivos pretendidos. Assim, podem ser consideradas as seguintes etapas:

- Inicializações necessárias – periféricos, configurações LoRa da placa de expansão e sensores;
- Conexão com o servidor LoRaWAN;
- Receção de informação – tanto de sensores como do sistema NEXT-Road a partir de CAN FD ou RS485;
- Envio de informação – de acordo com um temporizador.

A partir do momento que o controlador se conecta ao servidor LoRaWAN, um temporizador inicia a contagem do tempo. A contagem chega a um valor de comparação indicado, que inicia uma interrupção. Esta inclui variáveis que contam o número de vezes que o microcontrolador percorre a interrupção, assim como um ciclo *switch-case* que verifica o valor dos contadores. Desta forma, é possível conseguir controlar o envio das mensagens assim como a sua frequência. Em ambiente de teste, o temporizador sofreu alterações, alternando entre minutos e segundos, no sentido de validar diferentes períodos de envio.

Pretende-se que as informações enviadas pelo controlador principal, cheguem em períodos constantes, dependendo da importância da mensagem. Portanto, com uma base estabelecida, é possível modificar o temporizador sem alterar, significativamente o *firmware* da interrupção.

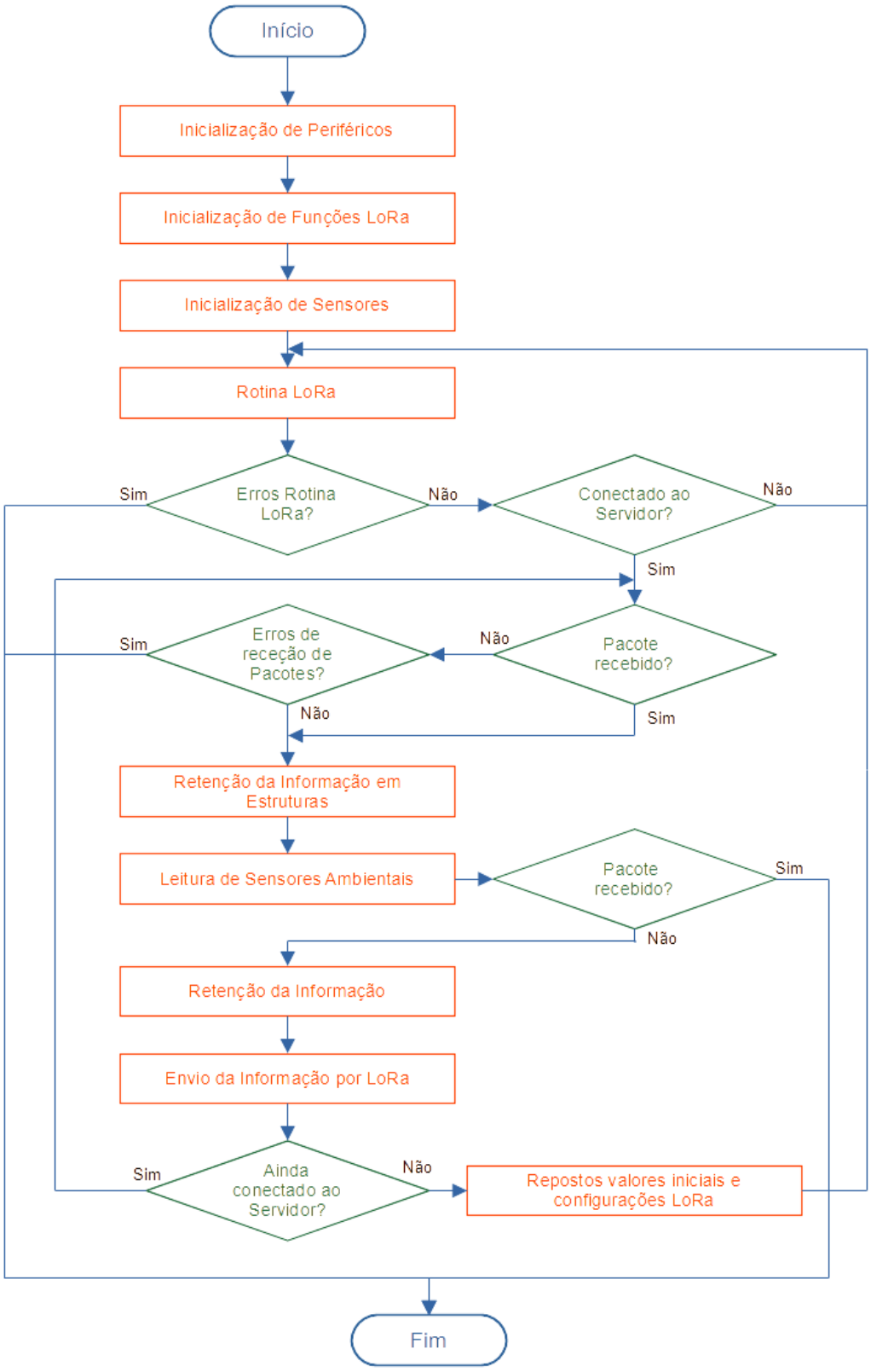


Figura 60 Fluxograma do ciclo principal do *firmware*

5.2.2. CONFIGURAÇÃO DA GATEWAY

De acordo com o referido no ponto 5.2, o *firmware* disponibilizado pela STMicroelectronics é programado no microcontrolador STM32F746ZG, através de um ficheiro binário. Este permite que possam ser efetuadas alterações no encaminhador de pacotes, de acordo com comandos AT. Após a sua instalação, apresenta na porta virtual do Nucleo-F746ZG, todas as informações observadas na Figura 61.

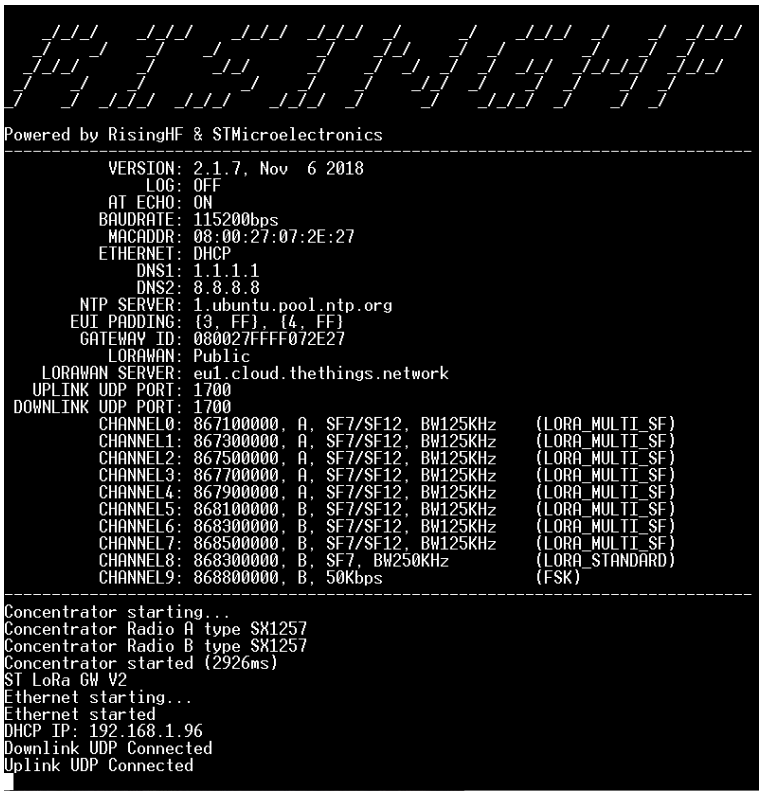


Figura 61 Configuração de parâmetros da *gateway*

As configurações, por omissão, estão preparadas para uma banda EU868, de qualquer forma foram enviados todos os comandos necessários para a comunicação da *gateway* com o servidor. Esta *gateway* só pode ser configurada para servidores que suportem o encaminhador da Semtech. Assim, foi necessário alterar o endereço do servidor LoRaWAN, para o servidor *The Things Stack*, este corresponde ao endereço, eu1.cloud.thethings.network, com os portos 1700, de *uplink* e *downlink*.

```
AT+PKTFWD=eu1.cloud.thethings.network,1700,1700
```

Após a verificação de todas as configurações apresentadas, a *gateway* necessita de ser reiniciada de forma a aplicar as alterações. A ligação da *gateway* a um *router* ou *switch*, com acesso à Internet, permite que esta esteja pronta para receber e reencaminhar os pacotes.

5.3. THE THINGS STACK

The Things Stack ou TTS, permite que a informação seja temporariamente retida num servidor/*cloud*. Esta necessita que aplicações sejam criadas e que dispositivos sejam registados nessas aplicações. Através da *console*, o processo de registo é aligeirado, de forma a serem apresentados os campos necessários para preenchimento. Assim, após o registo de equipamentos (*end nodes* e *gateways*), a *console* disponibiliza um *status* do equipamento em relação ao servidor (conectado ou desconectado), assim como todo o tráfego que percorre os dispositivos.

Numa primeira instância, é possível visualizar os pacotes enviados pelo controlador principal até ao servidor e o respetivo *payload*. O *payload* é apresentado no formato hexadecimal, pelo que a *console* permite aplicar um *decoder* aos pacotes que chegam daquele dispositivo, transformando de formato hexadecimal para informação contida em variáveis, que são reencaminhadas pelo integrador escolhido.

5.3.1. CONFIGURAÇÕES GERAIS

As configurações necessárias no servidor, dizem respeito ao registo dos dispositivos na aplicação. No caso do *end node* é preciso a introdução da AppEUI, DevEUI e AppKey, indicados nos dispositivos, assim como um identificador (DevID) para o dispositivo, que pode ser atribuído pelo utilizador.

5.3.2. DECODER DAS MENSAGENS

O *decoder* disponível no *console* permite utilizar JavaScript para tornar o *payload* num formato familiar. Assim, o *decoder* segue a organização dos bytes que foram enviados pelo controlador principal.

O primeiro byte do *payload* identifica sempre o *measurement* da base de dados a que a mensagem corresponde, pelo que a primeira análise é a verificação do valor do primeiro byte. A partir deste valor, os bytes são atribuídos a variáveis que correspondem à

informação pretendida. Existem mensagens em que o primeiro byte contém ainda a identificação do módulo, a que o pacote diz respeito. A abordagem é semelhante, para as outras mensagens do controlador principal.

```
if (bytes[0] == 0x20)              //Parâmetros
Ambientais
    {
    tabela = 0x01;
    var lum = ((bytes[2] << 8) | bytes[1])/10;
    var hum_amb = bytes[3];
    var temp_amb = bytes[4] / 2;
    var iaq = (bytes[5])*2;
    var voc = (bytes[7] << 8) | bytes[6];
    var co2 = (bytes[9] << 8) | bytes[8];
    var som = bytes[10];
    var pressao = offset_pressao+bytes[11];

    return {
    Tabela: tabela,     //ID Base de Dados
    Luminosidade: lum,
    Humidade: hum_amb,
    Temperatura: temp_amb,
    IAQ: iaq,
    VOC: voc,
    CO2EQ: co2,
    Som: som,
    Pressao: pressao
    }
  }
```

5.4. ORGANIZAÇÃO E RETENÇÃO DOS DADOS

A organização e retenção da informação é conseguida através do Node-Red e da base de dados InfluxDB. Numa primeira fase são utilizados os *nodes* apresentados na Figura 62, no sentido de reencaminhar a informação para a base de dados. É usado um node de MQTT que, através das credenciais disponíveis na *console* da TTS, recebe os *uplinks*, em formato JSON, que chegam ao servidor LoRaWAN. O formato JSON é convertido para JavaScript de forma que a sua organização, seja mais eficiente e de simples compreensão.

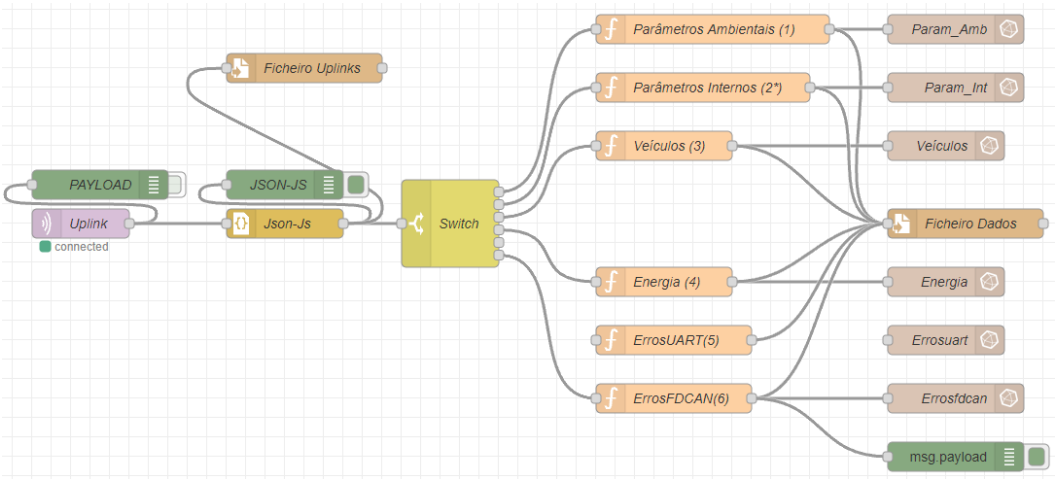


Figura 62 Fluxo de *nodes*

Do *payload* recebido é analisado o valor da variável "Tabela", este define o *measurement* a que a mensagem pertence. Antes da introdução dos valores na base de dados, estes necessitam de ser alterados e organizados de maneira que a base de dados os reconheça como *fields*. Alguns destes *nodes* fazem o retorno dos valores com variáveis novas, que irão ser identificadas na base de dados (*field keys*). Outros necessitam de alterações como as mensagens de erros, que através de um byte identificam 7 tipos de erros do sistema NEXT-Road.

```
var data = msg.payload.uplink_message;
return{
    payload:{
          Luminosidade:
data.decoded_payload.Luminosidade,
          Humidade: data.decoded_payload.Humidade,
          Temperatura:
data.decoded_payload.Temperatura,
          CO2EQ: data.decoded_payload.CO2EQ,
          IAQ: data.decoded_payload.IAQ,
          VOC: data.decoded_payload.VOC,
          Pressao_Atmosferica:
data.decoded_payload.Pressao,
          Som: data.decoded_payload.Som
    }
}
```

Após efetuar a respetiva organização, a informação é inserida nos *measurements* através dos *nodes*. Estes identificam o servidor de base de dados em execução no sistema, assim como as bases de dados criadas. Assim, estes *nodes* recebem o nome que identifica a

base de dados e o *measurement* a qual a mensagem se destina. A Figura 63 apresenta um exemplo de um *measurement* após os dados serem inseridos através do Node-Red.

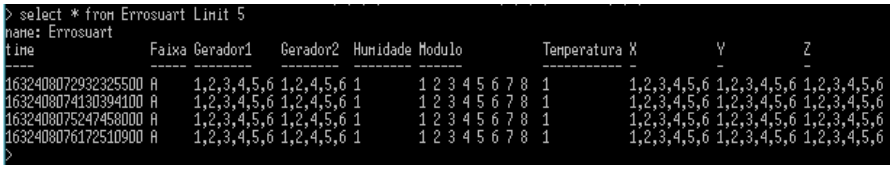


Figura 63 Exemplo de disposição de um *measurement*

5.5. DASHBOARDS GRAFANA

A Grafana é uma ferramenta que permite que a informação retida na base de dados seja apresentada. A pensar num possível cliente para a implementação do sistema, este consiste num conjunto de *dashboards* de acordo com a organização da base de dados. Cada *dashboard* possui duas abas, uma com os valores apresentados denominadas de "Resumo" (ver Figura 64) e uma aba onde estão presentes os gráficos de acordo com a linha temporal, com o nome "Gráficos" (ver Figura 65). Assim, foram criadas sete *dashboards* com as seguintes funções:

- Capa – onde está presente o logotipo da empresa com uma imagem de fundo, referente às *smart cities;*
- Parâmetros ambientais – Valores obtidos pelos sensores ambientais;
- Parâmetros internos – Valores de temperatura, humidade e energia gerada de dez conjuntos de dez módulos;
- Veículos – Valores de velocidades, contagem e peso;
- Energia – Valores de energia total gerada e nível da bateria;
- Erros – Informação dos erros do sistema NEXT-Road assim como do controlador principal;
- Analise de dados – Informação cruzada, como por exemplo, a geração de energia de acordo com a velocidade do veículo.

Após a instalação do sistema, as *dashboards* podem ser percorridas em *playlist*, o que convida a serem apresentadas num televisor, tornando o ato de monitorizar mais eficiente. Os gráficos apresentados possuem uma opção de definição de alertas, o que possibilita a deteção de erros futuros do sistema, como temperatura e humidades elevadas. Os alertas são sempre definidos a partir de um valor limite ou intervalo de valores, assim que os dados correspondam a essas condições, o alerta é analisado e verificado durante um período de tempo que pode ser definido pelo utilizador.

Figura 64 Exemplificação de valores obtidos dos sensores ambientais

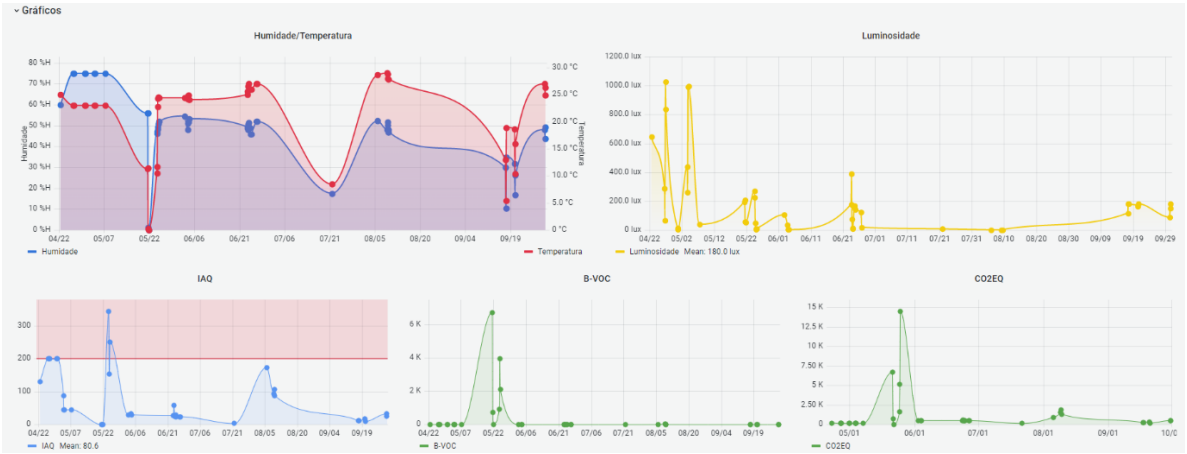


Figura 65 Exemplificação de gráficos com os dados dos sensores ambientais

6. TESTES E RESULTADOS

Durante a fase inicial de implementação e teste dos sensores, recorreu-se a uma *breadboard*, de maneira a conseguir efetuar as ligações elétricas eficientemente. Nesta etapa, a comunicação por LoRa ainda estava a ser desenvolvida. A Figura 66 apresenta os sensores em *breadboard*, assim como o Nucleo-G474RE. Com a informação recebida, já era possivel organizá-la nas estruturas, para posteriormente, ser enviada por LoRa.

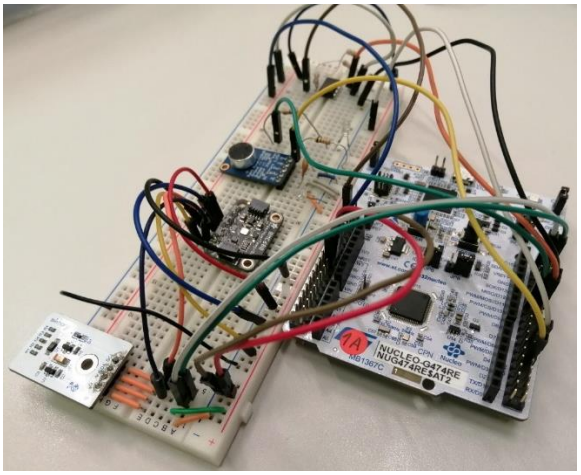


Figura 66 Implementação e testes dos sensores em *breadboard*

Com o envio de dados para o servidor validada, foram efetuados dois testes que permitiram verificar a continuidade do envio de pacotes. Os testes foram efetuados em períodos de tempo diferentes em que o primeiro teste teve uma duração de 24 horas e o segundo 68 horas (ver Figura 67), de mensagens imitidas continuamente.

Figura 67 Resultados de temperatura, humidade e pressão após 68 horas

Numa fase avançada de desenvolvimento, foram registados testes de envio de pacotes de cada uma das *dashboards*. Ocorreram testes de validação em ambiente de laboratório, dias antes de os componentes serem preparados para serem testados no terreno. Estes testes foram bem sucedidos e todos os componetes dos sistemas funcionavam de acordo com o esperado.

Além dos valores dos sensores ambientais, os restantes dados são fixos e fictícios, sendo a transmissão de pacotes, independente do sistema NEXT-Road (ver Figura 68, Figura 69, Figura 70 e Figura 71). Todos os componentes dos sistemas funcionavam de acordo com o esperado, os pacotes estavam a ser enviados a uma frequência de dez segundos por mensagem. Verificou-se que todos os pacotes eram recebidos pelo servidor e devidamente apresentados na Grafana.

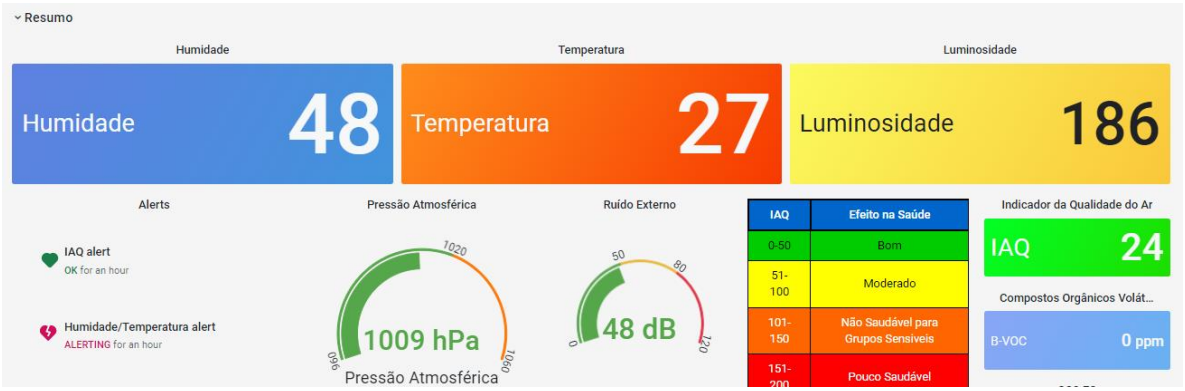


Figura 68 Dados de sensores ambientais de teste em laboratório

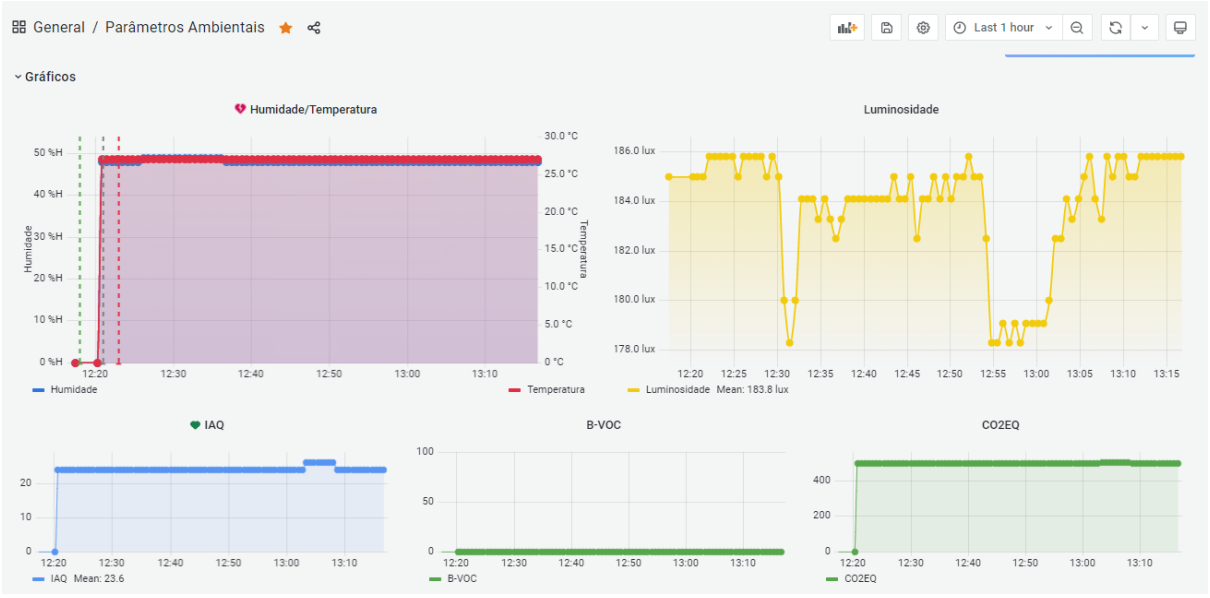


Figura 69 Gráficos de dados de sensores ambientais de teste em laboratório

Figura 70 Dados de sensores internos em teste de laboratório

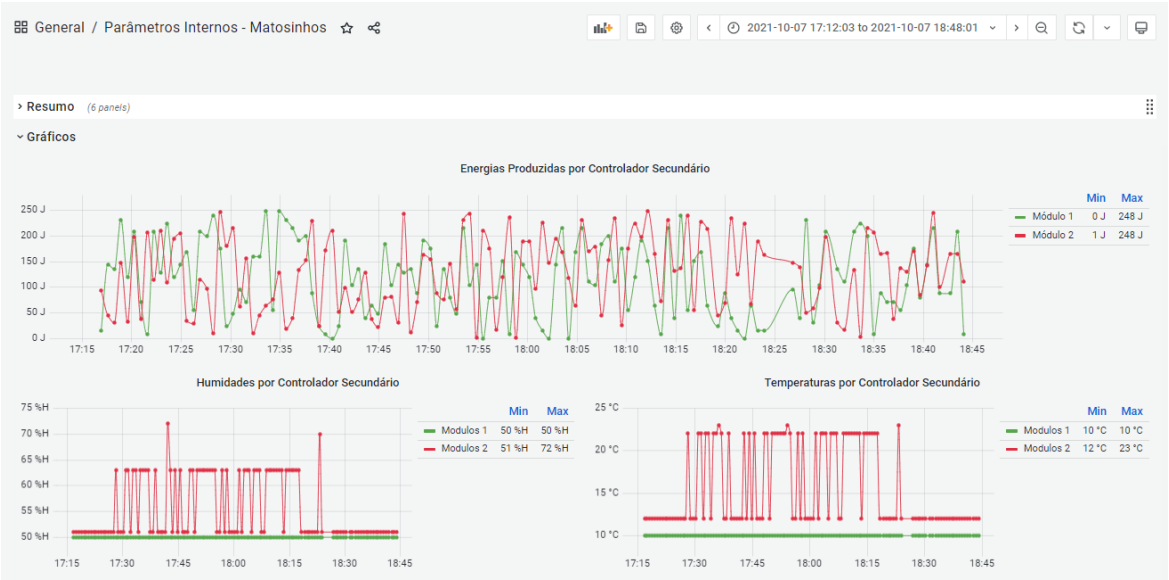


Figura 71 Gráficos dos dados de sensores internos em teste de laboratório

Os testes finais decorreram em Matosinhos, onde estavam a ser realizados testes pilotos do sistema NEXT-Road. A preparação para os testes constituiu nos seguintes passos: efetuar ligações elétricas no quadro junto à via rodoviária, alimentação do controlador principal com 12 V, provenientes das baterias (ver Figura 72), implementação da *gateway* nas instalações da Câmara Municipal de Matosinhos (a cerca de cento e quarenta metros do controlador principal) (ver Figura 73), observação dos dados obtidos desde o servidor até à Grafana.

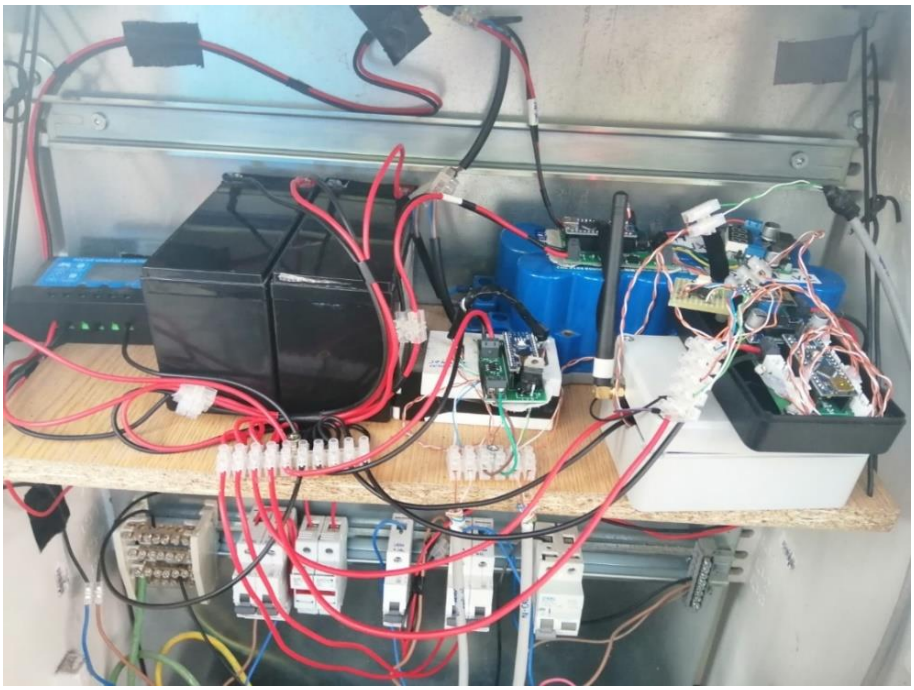


Figura 72 Instalação do controlador principal no terreno

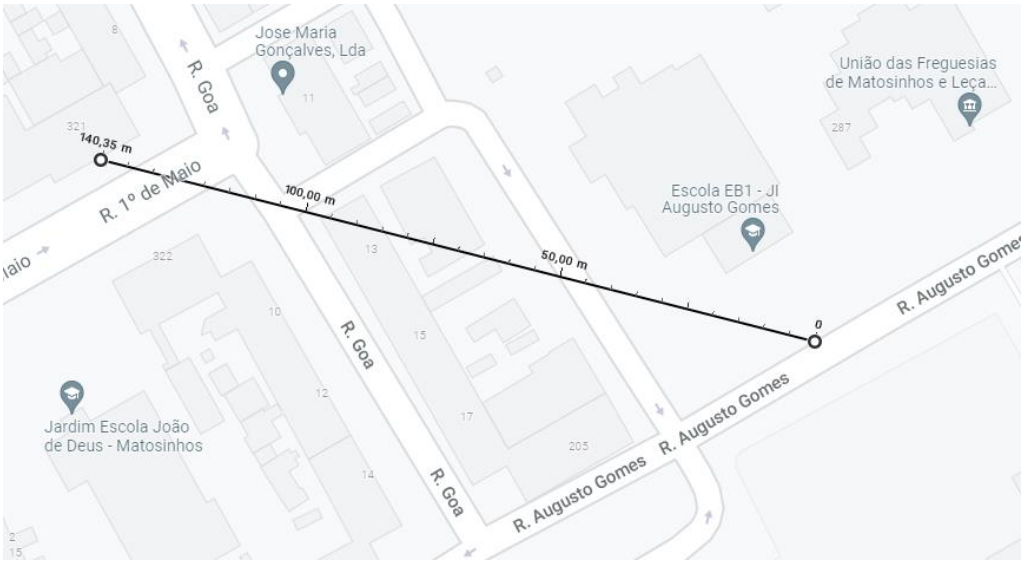


Figura 73 Distância de comunicação

Devido às condições de instalação do controlador principal, as medições de alguns dos parâmetros ambientais, sofreram alterações. O facto de estar fechado no quadro elétrico impossibilita a medição da intensidade luminosa assim, como alterar as medições de temperatura e humidade do ambiente. O sistema NEXT-Road, gradualmente começou a demonstrar erros (ligações entre componentes), desde que foi permitida a passagem de veículos pelo pavimento. Ao mesmo tempo, o controlador principal necessitou de ser atualizado, devido a um comportamento inconsistente e à repetição de valores de velocidade.

Existiu um pequeno período em que o sistema conseguiu recolher dados fiáveis com todos os seus componentes operacionais (ver Figura 74). Sem o sistema NEXT-Road totalmente operacional, o controlador principal não recebe os pacotes com os dados para serem devidamente apresentados. Demonstra-se assim que o sistema de organização e apresentação de dados funciona corretamente, assim que este seja alimentado com dados.

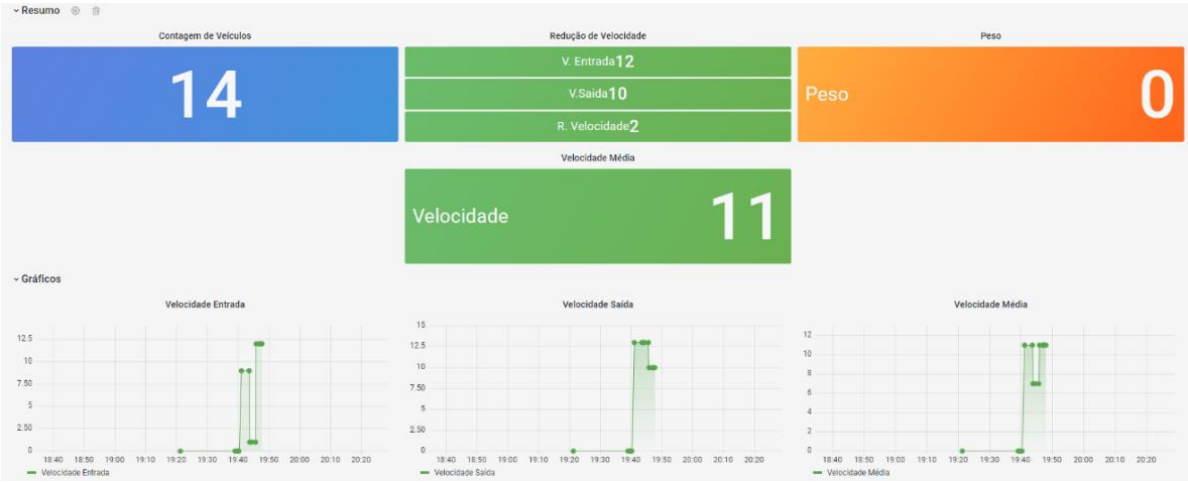


Figura 74 Valores de velocidade testados no terreno

As restrições impostas na transmissão através de LoRa não foram consideradas, visto que o período de operação dos dispositivos foi reduzido e os testes realizados, não podem ser considerados como uma implementação definitiva. Durante todo o desenvolvimento, as comunicações efetuadas permitiram definir os períodos de envio de mensagens para o servidor, assim como o respetivo ajuste de *firmware*.

7. CONCLUSÕES

O presente documento provém do desenvolvimento de um projeto proposto pela empresa Pavnext, assim como interesse pessoal em desenvolver soluções IoT e aprofundar o conhecimento, nas diversas tecnologias utilizadas atualmente. Desta forma, os capítulos 4 e 5 descrevem a solução implementada, de acordo com as possibilidade e objetivos do projeto. Os objetivos referidos no capítulo 1 foram cumpridos, com períodos de algumas dificuldades no *firmware* e *hardware*, decorrentes do desenvolvimento de soluções nunca realizadas. No entanto, o sistema revela-se, claramente, uma solução para o problema apresentado, com a demonstração de que é capaz de funcionar em aplicações reais, junto à via rodoviária.

De acordo com a organização do documento, este apresenta uma componente de *firmware*, *hardware* e *software.* A componente de *firmware* foi alterada durante todo o desenvolvimento do projeto e revelou-se essencial para a implementação do sistema. Desta forma, este foi otimizado para se tornar uma solução robusta e fiável.

Em termos de *hardware*, não existiram dificuldades relevantes. Os sistemas utilizados provêm de um kit de desenvolvimento da STMicroelectronics, e como tal, destinado a ser utilizado em ambiente de teste. Este apresentou claras valências de forma a ser utilizado neste projeto, assim é possível utilizar o *hardware* referido, como uma implementação real do sistema desenvolvido.

Num aspeto geral, o sistema teve os resultados esperados em ambiente de testes e no terreno. Este é escalável, ou seja, existe a possibilidade de acrescentar mais controladores principais, a sistemas NEXT-Road, e aprofundar a organização e visualização da informação. No entanto são reconhecidas diferentes abordagens que melhorariam o projeto. A utilização da tecnologia NB-IoT, impulsionaria a capacidade e a confiabilidade das transmissões, o investimento em sensores robustos e a instalação dos mesmos, permitiria uma capacidade de análise consistente. A implementação dos referidos melhoramentos, implicaria um conhecimento aprofundado nos sistemas IoT, assim como um investimento constante nestes sistemas.

Referências Documentais

[1] V. E. Balas, V. K. Solanki, R. Kumar, and M. Khari, *Internet of Things and Big Data Analytics for Smart Generation*. Springer, 2020.

[2] T. Segal, “Big Data Definition,” Jan. 01, 2021. https://www.investopedia.com/terms/b/big-data.asp (accessed Sep. 21, 2021).

[3] A. S. Gillis, “What is IoT (Internet of Things) and How Does it Work?,” Aug. 2021. https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT (accessed Oct. 01, 2021).

[4] United Nations Department of Economic and Social Affairs, “68% of the world population projected to live in urban areas by 2050, says UN ,” New York, May 16, 2018.

[5] H. Kharas and J. Remes, “Smart cities have an opportunity to become far more inclusive | World Economic Forum,” Jun. 08, 2018. https://www.weforum.org/agenda/2018/06/can-smart-cities-be-equitable (accessed Oct. 01, 2021).

[6] P. Wegner, “The top 10 Smart City use cases that are being prioritized now,” Sep. 01, 2020. https://iot-analytics.com/top-10-smart-city-use-cases-prioritized-now/ (accessed Oct. 03, 2021).

[7] M. Sarrab, S. Pulparambil, and M. Awadalla, “Development of an IoT based real-time traffic monitoring system for city governance,” *Glob. Transitions*, vol. 2, pp. 230–245, 2020, doi: 10.1016/j.glt.2020.09.004.

[8] B. S. Chaudhari, M. Zennaro, and S. Borkar, “LPWAN technologies: Emerging application characteristics, requirements, and design considerations,” *Futur. Internet*, vol. 12, no. 3, 2020, doi: 10.3390/fi12030046.

[9] P. Joshi, “10 Best IoT Applications (Internet of Things: Applications),” Nov. 01, 2020. https://www.techgeekbuzz.com/iot-applications/ (accessed Mar. 03, 2021).

[10] M. Ma, P. Wang, and C. H. Chu, “Data management for internet of things:

Challenges, approaches and opportunities," *Proc. - 2013 IEEE Int. Conf. Green Comput. Commun. IEEE Internet Things IEEE Cyber, Phys. Soc. Comput. GreenCom-iThings-CPSCom 2013*, pp. 1144–1151, 2013, doi: 10.1109/GreenCom-iThings-CPSCom.2013.199.

[11] C. T. Meadow, "History of databases," *J. Am. Soc. Inf. Sci.*, vol. 38, no. 4, pp. 309–309, 1987, doi: 10.1002/(SICI)1097-4571(198707)38:4<309::AID-ASI16>3.0.CO;2-#.

[12] C. J. Date, *An Introduction to Database Systems*, 8th ed. 2003.

[13] "IBM100 - Information Management System." https://www.ibm.com/ibm/history/ibm100/us/en/icons/ibmims/ (accessed Feb. 01, 2021).

[14] "A Timeline of Database History | Quickbase." https://www.quickbase.com/articles/timeline-of-database-history (accessed Feb. 01, 2021).

[15] S. Zhang, W. Zeng, I. L. Yen, and F. B. Bastani, "Semantically enhanced time series databases in IoT-edge-cloud infrastructure," *Proc. IEEE Int. Symp. High Assur. Syst. Eng.*, vol. 2019-Janua, pp. 25–32, 2019, doi: 10.1109/HASE.2019.00014.

[16] "ScyllaDB | NoSQL vs SQL." https://www.scylladb.com/resources/nosql-vs-sql/ (accessed Feb. 04, 2021).

[17] S. Rautmare and D. M. Bhalerao, "MySQL and NoSQL database comparison for IoT application," *2016 IEEE Int. Conf. Adv. Comput. Appl. ICACA 2016*, pp. 235–238, 2017, doi: 10.1109/ICACA.2016.7887957.

[18] Amazon, "O que é um banco de dados gráfico?" https://aws.amazon.com/pt/nosql/graph/ (accessed Feb. 04, 2021).

[19] "2019 Database Trends - SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use," Mar. 04, 2019. https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/ (accessed Dec. 06, 2020).

[20] S. Noor, Z. Naqvi, S. Yfantidou, and E. Zi, "Advanced Databases Time Series

Databases and InfluxDB," *Univ. Libr. Bruxelles*, no. 000455274, 2017, [Online]. Available: http://cs.ulb.ac.be/public/_media/teaching/influxdb_2017.pdf.

[21] "DB-Engines Ranking per database model category." https://db-engines.com/en/ranking_categories (accessed Feb. 08, 2021).

[22] "DB-Engines Ranking - popularity ranking of database management systems." https://db-engines.com/en/ranking (accessed Feb. 01, 2021).

[23] "Relational Vs. Time-Series Databases: Become an Expert in 20 Minutes - Sudokrew Main Website." https://www.sudokrew.com/insights/relational-vs-time-series-databases (accessed Feb. 21, 2021).

[24] InfluxData, "Get started with Flux | InfluxDB Cloud Documentation." https://docs.influxdata.com/influxdb/cloud/query-data/get-started/ (accessed Oct. 14, 2021).

[25] InfluxData, "InfluxDB key concepts | InfluxDB OSS 1.7 Documentation." https://docs.influxdata.com/influxdb/v1.7/concepts/key_concepts/ (accessed Oct. 14, 2021).

[26] InfluxData, "InfluxDB glossary | InfluxDB OSS 1.7 Documentation." https://docs.influxdata.com/influxdb/v1.7/concepts/glossary/#replication-factor (accessed Oct. 14, 2021).

[27] InfluxData, "Dashboards." https://www.influxdata.com/dashboards/ (accessed Oct. 14, 2021).

[28] "Intro to TimescaleDB New Users Guide 2019."

[29] "Overview - Comparison: PostgreSQL | latest | TimescaleDB Docs." https://docs.timescale.com/latest/introduction/timescaledb-vs-postgres (accessed Jan. 31, 2021).

[30] "Overview - Architecture | latest | TimescaleDB Docs." https://docs.timescale.com/latest/introduction/architecture (accessed Jan. 31, 2021).

[31] "Overview - Comparison: NoSQL | latest | TimescaleDB Docs."

https://docs.timescale.com/latest/introduction/timescaledb-vs-nosql (accessed Jan. 31, 2021).

[32] “Overview | Prometheus.” https://prometheus.io/docs/introduction/overview/ (accessed Dec. 13, 2020).

[33] J. Turnbull, “The Prometheus Backstory,” in *Monitoring with Prometheus*, 2018, pp. 47–48.

[34] J. Turnbull, “Prometheus Arquitecture,” in *Monitoring with Prometheusn*, 2018, pp. 48–54.

[35] “Grafana Features | Grafana Labs.” https://grafana.com/grafana/ (accessed Dec. 06, 2020).

[36] J. Turnbull, “Metrics,” in *Monitoring with Prometheus*, 2018, pp. 18–32.

[37] “Graphite.” https://graphiteapp.org/ (accessed Dec. 13, 2020).

[38] “Graphite Documentation — Graphite 1.2.0 documentation.” https://graphite.readthedocs.io/en/latest/ (accessed Dec. 13, 2020).

[39] “FAQ — Graphite 1.2.0 documentation.” https://graphite.readthedocs.io/en/latest/faq.html (accessed Dec. 13, 2020).

[40] “Feeding In Your Data — Graphite 1.2.0 documentation.” https://graphite.readthedocs.io/en/latest/feeding-carbon.html (accessed Dec. 13, 2020).

[41] “Graph panel | Grafana Labs.” https://grafana.com/docs/grafana/latest/panels/visualizations/graph-panel/ (accessed Dec. 06, 2020).

[42] Y. Song, J. Lin, M. Tang, and S. Dong, “An Internet of Energy Things Based on Wireless LPWAN,” *Engineering*, vol. 3, no. 4, pp. 460–466, 2017, doi: 10.1016/J.ENG.2017.04.011.

[43] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, “A comparative study of LPWAN technologies for large-scale IoT deployment,” *ICT Express*, vol. 5, no. 1, pp. 1–7,

2019, doi: 10.1016/j.icte.2017.12.005.

[44] Behr Technologies, “What is LPWAN | A Deep Dive into Low Power Wide Area Networks.” https://behrtech.com/blog/what-is-lpwan-a-deep-dive-into-low-power-wide-area-networks/ (accessed Feb. 23, 2021).

[45] R. K. Singh, P. P. Puluckul, R. Berkvens, and M. Weyn, “Energy consumption analysis of LPWAN technologies and lifetime estimation for IoT application,” *Sensors (Switzerland)*, vol. 20, no. 17, pp. 1–22, 2020, doi: 10.3390/s20174794.

[46] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, “Understanding the Limits of LoRaWAN,” *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, 2017, doi: 10.1109/MCOM.2017.1600613.

[47] “LoRa and LoRaWAN: Technical overview | DEVELOPER PORTAL.” https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/ (accessed Mar. 20, 2021).

[48] B. Reynders, W. Meert, and S. Pollin, “Range and coexistence analysis of long range unlicensed communication,” *2016 23rd Int. Conf. Telecommun. ICT 2016*, 2016, doi: 10.1109/ICT.2016.7500415.

[49] Qoitech, “How Spreading Factor affects LoRaWAN device battery life,” Nov. 18, 2019. https://www.thethingsnetwork.org/article/how-spreading-factor-affects-lorawan-device-battery-life (accessed Mar. 21, 2021).

[50] S. Kim, H. Lee, and S. Jeon, “An Adaptive Spreading Factor Selection Scheme for a Single Channel LoRa Modem,” doi: 10.3390/s20041008.

[51] H. Ruotsalainen and S. Grebeniuk, “Towards wireless secret key agreement with LoRa physical layer,” *ACM Int. Conf. Proceeding Ser.*, no. August, 2018, doi: 10.1145/3230833.3232803.

[52] “µMETOS Technical notes - METOS by Pessl instruments.” https://metos.at/pb/micrometos-technical-notes/ (accessed Mar. 21, 2021).

[53] D. Zorbas and B. O’Flynn, “Autonomous collision-free scheduling for lora-based industrial internet of things,” *20th IEEE Int. Symp. A World Wireless, Mob.*

Multimed. Networks, WoWMoM 2019, 2019, doi: 10.1109/WoWMoM.2019.8792975.

[54] A. Davis, “Forward Error Correction | EE Times,” Jun. 12, 1998. https://www.eetimes.com/forward-error-correction/ (accessed Mar. 23, 2021).

[55] J. Matondang, “Spreading Factor, Bandwidth, Coding Rate and Bit Rate in LoRa (English) ,” Aug. 14, 2018. https://josefmtd.com/2018/08/14/spreading-factor-bandwidth-coding-rate-and-bit-rate-in-lora-english/ (accessed Mar. 24, 2021).

[56] M. Bor and U. Roedig, “LoRa transmission parameter selection,” *Proc. - 2017 13th Int. Conf. Distrib. Comput. Sens. Syst. DCOSS 2017*, vol. 2018-Janua, pp. 27–34, 2018, doi: 10.1109/DCOSS.2017.10.

[57] “LoRa — LoRa documentation.” https://lora.readthedocs.io/en/latest/#range-vs-power (accessed Mar. 24, 2021).

[58] “About LoRa Alliance® - LoRa Alliance®.” https://lora-alliance.org/about-lora-alliance/ (accessed Mar. 24, 2021).

[59] “LoRaWAN Architecture | The Things Network.” https://www.thethingsnetwork.org/docs/lorawan/architecture.html (accessed Mar. 24, 2021).

[60] Semtech, “Long Range, Low Power RF Transceiver with LoRa Technology.” https://www.semtech.com/products/wireless-rf/lora-transceivers (accessed Mar. 24, 2021).

[61] arm, “LoRaWAN - API references and tutorials | Mbed OS 6 Documentation.” https://os.mbed.com/docs/mbed-os/v6.8/apis/lora-tech.html (accessed Mar. 24, 2021).

[62] The Things Network, “Classes .” https://www.thethingsnetwork.org/docs/lorawan/classes.html (accessed Mar. 24, 2021).

[63] S. Devalal and K. Iot, “LoRa technology-an overview,” *2018 Second Int. Conf. Electron. Commun. Aerosp. Technol.*, no. Iceca, pp. 284–290, 2018.

[64] J. Quere, “LoRaWan, a dedicated IoT network ,” Jan. 17, 2018. https://witekio.com/blog/lorawan-a-dedicated-iot-network/ (accessed Mar. 24, 2021).

[65] MultiTech, “LoRaWAN ® Security,” 2017.

[66] MachineQ, “What are the DevEUI and DevAddr?” https://support.machineq.com/s/article/What-is-the-DevEUI-and-DevAddr (accessed Apr. 02, 2021).

[67] The Things Stack for LoRaWAN, “ABP vs OTAA .” https://www.thethingsindustries.com/docs/devices/abp-vs-otaa/ (accessed Apr. 02, 2021).

[68] E. Aras, G. S. Ramachandran, P. Lawrence, and D. Hughes, “Exploring the security vulnerabilities of LoRa,” *2017 3rd IEEE Int. Conf. Cybern. CYBCONF 2017 - Proc.*, 2017, doi: 10.1109/CYBConf.2017.7985777.

[69] NewieVentures, “LoRaWAN: OTAA or ABP? .” https://www.newieventures.com.au/blogtext/2018/2/26/lorawan-otaa-or-abp (accessed Apr. 02, 2021).

[70] Ray Brian, “SigFox Vs. LoRa: A Comparison Between Technologies & Business Models,” May 31, 2018. https://www.link-labs.com/blog/sigfox-vs-lora (accessed Oct. 18, 2021).

[71] Altium, “The Pros and Cons of Different LPWAN Networks for Your IoT Application ,” Dec. 21, 2020. https://resources.altium.com/p/the-pros-and-cons-of-different-lpwan-networks-for-your-iot-application (accessed Oct. 19, 2021).

[72] M. Aernouts, R. Berkvens, K. Van Vlaenderen, and M. Weyn, “Sigfox and LoRaWAN datasets for fingerprint localization in large urban and rural areas,” *Data*, vol. 3, no. 2, pp. 1–15, 2018, doi: 10.3390/data3020013.

[73] Sigfox, “Our story.” https://www.sigfox.com/en/sigfox-story (accessed Oct. 18, 2021).

[74] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, “Overview of Cellular LPWAN

Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT," *2018 IEEE Int. Conf. Pervasive Comput. Commun. Work. PerCom Work. 2018*, pp. 197–202, 2018, doi: 10.1109/PERCOMW.2018.8480255.

[75] A. Lavric, A. I. Petrariu, and V. Popa, "Long Range SigFox Communication Protocol Scalability Analysis under Large-Scale, High-Density Conditions," *IEEE Access*, vol. 7, pp. 35816–35825, 2019, doi: 10.1109/ACCESS.2019.2903157.

[76] Sigfox, "What is Sigfox?" https://build.sigfox.com/sigfox#understand-the-sigfox-technology (accessed Oct. 17, 2021).

[77] M. Kazmi and A. Behravan, "Narrowband IoT standardization soon finalized," May 12, 2016. https://www.ericsson.com/en/blog/2016/5/narrowband-iot-standardization-soon-finalized (accessed Oct. 18, 2021).

[78] Y. P. E. Wang *et al.*, "A Primer on 3GPP Narrowband Internet of Things," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 117–123, 2017, doi: 10.1109/MCOM.2017.1600510CM.

[79] A. Adhikary, X. Lin, Y. E. Wang, H. Way, and S. Jose, "Performance Evaluation of NB-IoT Coverage," 2016.

[80] A. Palaez, "LoRaWAN vs NB-IoT: A Comparison Between IoT Trend-Setters," 2020. https://ubidots.com/blog/lorawan-vs-nb-iot/ (accessed Oct. 20, 2021).

[81] G. Strait, "LTE-M vs NB-IoT: An Overview of Narrowband IoT (NB-IoT) [2021 UPDATE]," Apr. 13, 2021. https://www.link-labs.com/blog/overview-of-narrowband-iot (accessed Oct. 20, 2021).

[82] Altice Portugal, "NarrowBand-IoT." https://www.telecom.pt/pt-pt/media/noticias/paginas/2018/novembro/narrowband-iot-.aspx (accessed Oct. 18, 2021).

[83] D. T. Iot, "NB-IoT, LoRaWAN, Sigfox: An up-to-date comparison."

[84] E. Jubin Sebastian, A. Sikora, M. Schappacher, and Z. Amjad, "Test and Measurement of LPWAN and Cellular IoT Networks in a Unified Testbed," *IEEE Int. Conf. Ind. Informatics*, vol. 2019-July, pp. 1521–1527, 2019, doi:

10.1109/INDIN41052.2019.8972256.

[85] “BME680-Datasheet,” Jun. 2020. https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme680-ds001.pdf (accessed Dec. 06, 2020).

[86] “Gas Sensor BME680 | Bosch Sensortec.” https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors-bme680/ (accessed Dec. 06, 2020).

[87] Maxim Integrated, “MAX9814 Microphone Amplifier with AGC and Low-Noise Microphone Bias,” *Datasheet*, p. 14, 2009, [Online]. Available: http://datasheets.maximintegrated.com/en/ds/MAX9814.pdf.

[88] Velleman, “BH1750 Digital Light-Intensity Sensor Module User Manual,” Accessed: Oct. 23, 2021. [Online]. Available: www.arduino.org.

[89] ROHM semiconductor, “Digital 16bit Serial Output Type Ambient Light Sensor IC,” no. 11046, p. 21, 2011, [Online]. Available: www.rohm.com.

[90] STMicroelectronics, “P-NUCLEO-LRWAN2 - STM32 Nucleo pack LoRa™ HF band sensor and gateway .” https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/stm32-nucleo-expansion-boards/p-nucleo-lrwan2.html (accessed Oct. 23, 2021).

[91] STMicroelectronics, “I-NUCLEO-LRWAN1,” no. March, pp. 1–4, 2018.

[92] RisingHF, “ST Nucleo LoRa GW HW User Manual.”

[93] Stmicroelectronics, “Getting started with the P-NUCLEO-LRWAN2 and P-NUCLEO-LRWAN3 starter packs - User manual;,” no. April, 2021, [Online]. Available: www.st.com.

[94] ST-Microelectronics, “User manual STM32CubeIDE quick start guide,” no. April, pp. 1–11, 2019.

[95] B. S. Gmbh, “Integration Guide Bosch Software Environmental Cluster (BSEC),”

2020.