TECNOLOGIA SETÚBAL
ESCOLA SUPERIOR
POLITÉCNICO SETÚBAL

Rúben Miguel da Costa André

LoRaGrid – Sistema de monitorização de consumos elétricos

Relatório de Dissertação do Mestrado em Engenharia Eletrotécnica e de Computadores

ORIENTADOR

Doutor Filipe Duarte dos Santos Cardoso, ESTSetúbal/IPS

CO-ORIENTADOR

Mestre José Garcia Costa Correia de Sousa, ESTSetúbal/IPS

dezembro/2024

Rúben Miguel da Costa André

LoRaGrid – Sistema de monitorização de consumos elétricos

JÚRI

Presidente: Doutor Armando José Pinheiro Marques Pires, ESTSetúbal/IPS
Orientador: Doutor Filipe Duarte dos Santos Cardoso, ESTSetúbal/IPS
Arguente: Doutor Jorge Manuel Martins, ESTSetúbal/IPS

dezembro/2024

Aos meus familiares, por todo o amor e inspiração, e aos meus amigos, pelo apoio e pela partilha de momentos inesquecíveis. Como disse Fernando Pessoa, "Tenho em mim todos os sonhos do mundo", e muitos deles foram impulsionados por vocês.

Agradecimentos

Agradeço, em primeiro lugar, aos meus pais, à minha namorada, e aos meus familiares, pelo apoio incondicional, pelo carinho e pela inspiração que me acompanharam ao longo deste percurso académico e pessoal. Sem eles, esta conquista não seria possível.

Ao Professor José Sousa, deixo o meu sincero agradecimento pela orientação dedicada e pela oportunidade de investigar este tema, que me permitiu expandir os meus horizontes e aprofundar os meus conhecimentos.

Ao Professor Filipe Cardoso, manifesto a minha gratidão pela gestão deste projeto e por tornar possível a realização desta investigação, oferecendo os recursos e os apoios necessários na escrita da dissertação e que contribuíram para o meu sucesso.

Ao meu colega e amigo, Bernardo Milheiro, pela troca de conhecimento que certamente impulsionou a investigação deste tema.

Aos monitores que permitiram usar o material dos laboratórios para que pudesse desenvolver o parte prática do projeto.

Por último, ao Instituto Politécnico de Setúbal, à Escola Superior de Tecnologia de Setúbal e a todos os que, de alguma forma, contribuíram para este trabalho, o meu muito obrigado por terem participado e tornado possível esta minha experiência académica.

Resumo

Este trabalho de investigação teve por objetivo o desenvolvimento de uma estrutura de Internet das Coisas (*Internet of Things IoT*) para o Instituto Politécnico de Setúbal (IPS) visando a monitorização de consumos elétricos para a Escola Superior de Tecnologia de Setúbal (ESTSetubal/IPS), com recurso a uma rede de sensores com interface *LoRa*. O principal foco da implementação foi o desenho duma rede automatizada e escalável, na medida em que se torne fácil e prático adicionar novos sensores na rede. Este projeto inclui o desenvolvimento duma aplicação *web* onde os dados recolhidos dos sensores são processados e exibidos de uma forma clara e objetiva aos utilizadores, com o uso de ferramentas para construção de gráficos, e com o devido controlo de permissões para que haja segurança no acesso aos dados. Para isso, desenhou-se um protótipo de um sensor *LoRa* que foi ligado ao servidor *The Things Network* (*TTN*), que por meio de uma ligação sem fios envia os dados para a aplicação final. A aplicação foi construída através da *framework* React e comunica com uma base dados para obter e atualizar a informação conforme a navegação do utilizador. Com a realização deste trabalho foi possível concluir que é necessário usar uma alternativa ao servidor *TTN* para que a plataforma do instituto possa ficar independente de serviços de terceiros, assim como alterar a aplicação *web* para que permita o controlo remoto dos sensores.

Palavras-chave: Internet of Things (IoT), Medidor de consumo de energia, Tecnologia LoRa, Aplicação web React

Abstract

This research aimed to develop an Internet of Things (IoT) framework for Polytechnic Institute of Setúbal (IPS), focusing on monitoring electricity consumption at the School of Technology of Setúbal (ESTSetúbal/IPS), using a network of sensors with a LoRa interface. The primary focus of the implementation was the design of an automated and scalable network, making it easy and practical to add new sensors to the network. This project included the development of a web application where the data collected from the sensors is processed and displayed clearly and objectively to users, using tools for building charts and ensuring proper access control for data security. To achieve this, a LoRa sensor prototype was designed and connected to The Things Network (TTN) server, which wirelessly sends the data to the final application. The application was built using the React framework and communicates with a database to fetch and update information according to user navigation. From this work, it was concluded that an alternative to TTN server is necessary to make the institute's platform independent of third-party services, as well as modifying the web application to enable remote control of the sensors.

Keywords: Internet of Things (IoT), Energy consumption meter, LoRa Technology, React web application

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

ABP	*Activation By Personalization*
ACID	Atomicidade-Consistência-Isolamento-Durabilidade
ADC	*Analog-to-Digital-Converter*
API	*Application Programming Interface*
BLE	*Bluetooth Low Energy*
BOD	*Brown-Out Detection*
CoAP	*Constrained Application Protocol*
CRC	*Cyclic Redundancy Check*
ESTSetúbal/IPS	Escola Superior de Tecnologia de Setúbal
ETL	*Extract-Transform-Load*
HTML	*HyperText Markup Languange*
IoT	*Internet of Things*
IPS	Instituto Politécnico de Setúbal
JSON	*JavaScript Object Notation*
LoRa	*LowRange Radio*
LoRaWAN	*Low Range Wide Area Network*
LPWAN	*Low Power Wide Area Network*
LTE-M	*Long Term Evolution M*
MAC	*Media Access Control*
MQTT	*MQ Telemetry Transport*
NB-IoT	*NarrowBand-Internet of Things*
OTAA	*Over-The-Air-Activation*
PHDR	*Physical Header*
QoE	*Quality of Experience*
QoL	*Quality of Life*
RSSF	Rede de Sensores Sem Fios
RSSI	*Received Signal Strength Indicator*

SNR	*Signal-to-Noise Ratio*
TTN	*The Things Network*
URL	*Uniform Resource Locator*
Wi-Fi ah	*Wi-Fi HaLow*
WoT	*Web of Things*
W3C	*World Wide Web Consortium*
WWW	*World Wide Web*
XML	*Extensible Markup Language*

Lista de Símbolos

ϕ	Diferença de fase (Desfasamento)
BW	Largura de banda
CR	*Coding Rate*
N	Número de amostras
Rb	Taxa de transmissão
Rs	Taxa de bits
SF	Fator de dispersão (*Spreading Factor*)

1.Introdução

Neste capítulo é feito a introdução do tema desta investigação, assim como da motivação para a realização desta dissertação. É, por último, apresentado a estrutura deste documento quanto aos capítulos e o assunto abordado em cada.

1.1. IoT e a monitorização de consumo elétrico

A *Internet of Things* (IoT) é um conceito que começou a ganhar destaque no final da década de 1990, quando Kevin Ashton introduziu o termo [1]. A IoT refere-se a uma rede de dispositivos físicos que trocam dados entre si de forma automatizada. Esses dispositivos, comumente com sensores integrados, podem coletar, processar e compartilhar informações, criando um ambiente que possibilita a automatização e otimização de várias funções, desde casas até indústrias automatizadas. Com o aumento da conectividade, redes sem fios e a expansão da internet, a IoT tem-se tornado fundamental em diversos setores, permitindo a automatização da recolha de dados e do seu processamento contribuindo, assim, para uma maior eficiência e precisão na gestão de processos e recursos.

Um dos setores em que se beneficia desta tecnologia é a monitorização de consumos energéticos. À medida que a preocupação com o meio ambiente cresce e a necessidade de reduzir o impacto ambiental das atividades humanas torna-se uma prioridade, os sistemas de monitorização baseados em IoT estão a ganhar força para monitorizar o uso de energia, tanto em ambientes residenciais quanto industriais. A capacidade de medir, monitorizar e controlar o consumo de eletricidade em tempo real, usando sensores e sistemas "inteligentes", permite otimizar o uso dos recursos energéticos e reduzir os desperdícios [2].

Os sistemas de monitorização de consumo elétrico IoT podem recolher dados sobre o uso de aparelhos elétricos, permitindo que os utilizadores identifiquem picos de consumo, equipamentos ineficientes ou possíveis anomalias. Essas informações são cruciais para melhorar a eficiência energética e ajudar a reduzir os custos operacionais. Além disso, podem ser integrados em sistemas de gestão de energia que ajudam a ajustar automaticamente o uso da energia elétrica com base nas necessidades e nos horários de pico.

Isso é feito através de uma combinação de sensores, atuadores, e microcontroladores como o *Raspberry Pi* [3], que medem o fluxo de eletricidade e transmitem esses dados para uma interface de utilizador, via internet, permitindo o acesso remoto a partir de qualquer dispositivo terminal [4]. A utilização desses sistemas não se limita ao controlo de custos. Em termos de sustentabilidade, a capacidade de monitorizar o uso de energia também contribui significativamente para a redução do impacto ambiental, uma vez que permite a integração com fontes de energias renováveis, como a solar e a eólica. As empresas, por exemplo, podem utilizar dados em tempo real para ajustar as suas operações de acordo com os horários de menor custo ou maior eficiência energética, o que resulta em redução de custos e menores emissões de carbono.

Nos últimos anos, surgiram diversos exemplos de monitorização do consumo através da IoT, como o uso de sensores que podem medir o consumo em intervalos regulares e fornecer informação imediata ao consumidor e às empresas de distribuição [5].

1.2. Motivação

Nos dias de hoje, a sustentabilidade e a eficiência energética tornaram-se temas cruciais para qualquer instituição que procura por um futuro mais responsável e ambientalmente consciente. A Escola Superior de Tecnologia de Setúbal (ESTSetúbal/IPS), no Instituto Politécnico de Setúbal (IPS), comprometida em reduzir a sua pegada ecológica, enfrenta o desafio de controlar os custos energéticos sem comprometer a qualidade dos serviços oferecidos. Este controlo é essencial para a projeção financeira e ambiental de projetos futuros, como a instalação de painéis solares, que podem contribuir significativamente para a geração de energia de fonte renovável no campus.

Ao longo dos últimos anos, o aumento do consumo energético tornou-se uma preocupação crescente, refletindo-se nos custos operacionais da instituição. A redução desses custos, aliada à preservação dos recursos naturais, é uma prioridade para garantir um desenvolvimento sustentável. Além disso, a escola pretende servir de exemplo para outras instituições, mostrando como é possível implementar soluções inteligentes e eficazes na gestão energética. A motivação para a realização deste trabalho de investigação surge, portanto, da necessidade de encontrar soluções práticas e inovadoras para otimizar o consumo da energia na escola.

Através de uma análise detalhada dos padrões de uso energético e da aplicação de tecnologias adequadas, será possível não só reduzir os custos, mas também promover uma cultura mais sustentável junto da comunidade. Para além disso, o instituto adquiriu um novo *gateway LowRange Radio Wide Area Network* (*LoRaWAN*) que abre portas para novos temas de projeto que utilizam esta tecnologia. Assim, este projeto é pioneiro no uso da modulação *LowRange Radio* (*LoRa*) no IPS e é, também, inovador no sentido em que representa um primeiro passo para mudanças que impactam positivamente o Instituto, abrindo portas para futuros projetos como a instalação de painéis solares. Tais projetos não só reduzirão os custos energéticos dos edifícios como contribuirão diretamente para o uso de energias de fonte renovável com impacto direto na eficiência energética dos mesmos.

Deste modo, espera-se que o trabalho desenvolvido nesta investigação possa ser aplicado na prática, criando um impacto positivo e duradouro tanto para a escola como para o meio ambiente. Esta oportunidade permitirá que a instituição avance na direção de uma gestão mais sustentável e inovadora, preparando o caminho para um futuro mais verde.

O projeto proposto consiste no desenvolvimento de um sistema com base na arquitetura da Figura 1, implicando o desenvolvimento de um sensor para a leitura de várias métricas de uma carga e realizar a sua transmissão sob a comunicação sem fios LoRa. Pretende-se implementar o sensor com base na integração de módulos existentes no mercado visando uma solução económica. Os dados gerados pelo sensor serão armazenados num servidor local que vai alimentar uma aplicação *web dashboard*, onde todos os dados vão ser apresentados em gráficos. O projeto contempla um sistema de alarmes, que comunica através de emails para os utilizadores sempre que forem encontradas situações anómalas nos consumos energéticos.

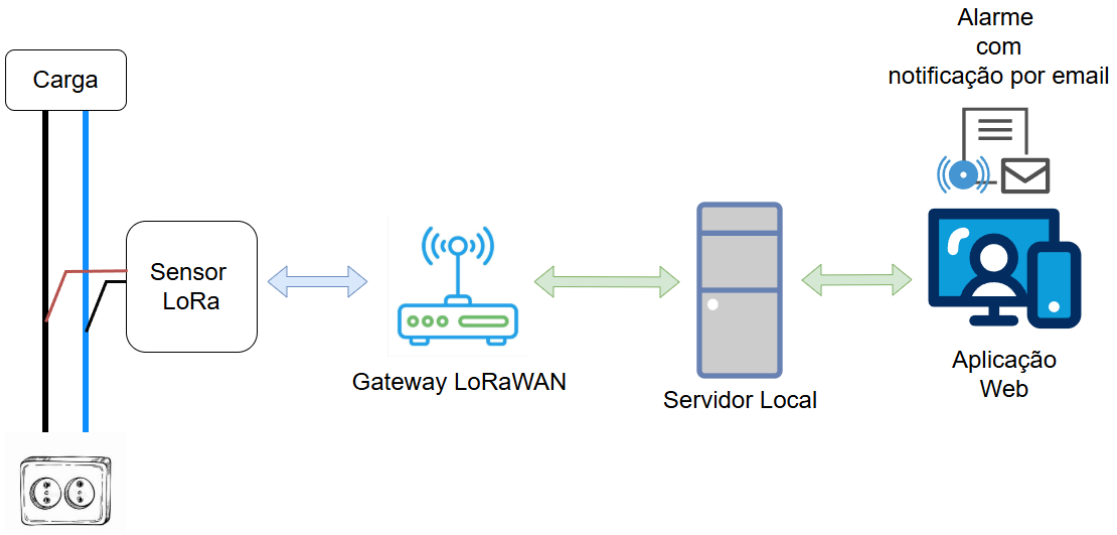


Figura 1 - Desenho da arquitetura do protótipo.

1.3. Estrutura da dissertação

Este documento encontra-se estruturado em seis capítulos, incluindo este. No capítulo dois é feita uma descrição dos temas em estudo neste trabalho de investigação, nomeadamente do conceito de *IoT*, e os seus benefícios para a sociedade e qualidade de vida. Faz-se uma introdução às arquiteturas IoT e aos sensores, tal como as topologias das arquiteturas de sensores, os protocolos de comunicação sem fios e o *Edge Computing*. Faz-se, ainda, a apresentação da tecnologia de modulação *LoRa* e de outras características como as classes dos dispositivos, noção dos servidores de rede *LoRaWAN*, identificação dos equipamentos numa rede *LoRaWAN* e métodos de autenticação. Por último, são mostrados os tipos de medidores de energia existentes e as tecnologias utilizadas para a implementação da camada de aplicação, *frontend* e *backend*.

No capítulo três são apresentados alguns casos de estudo onde se implementaram arquiteturas para a monitorização de consumos energéticos LoRa. Nos três casos apresentados descreve-se a construção dos respetivos sistemas de monitorização LoRa, fazendo-se um levantamento de vantagens e desvantagens de cada solução e a descrição das metodologias usadas na construção dos sensores e da rede propriamente dita. No capítulo quatro apresenta-se a construção da arquitetura proposta, passando pelos pontos mais relevantes da fase da implementação. Apresenta-se a escolha justificada do *hardware* utilizado para o sensor e também se demonstra o processo de desenho e montagem do medidor.

Também se explica a construção da aplicação final, das tecnologias utilizadas para *frontend* e *backend*, bem como, os passos realizados para a integração da rede como um todo. Neste capítulo é possível acompanhar a concretização da arquitetura de monitorização dos consumos de energia elétrica.

No capítulo cinco apresentam-se resultados de alguns testes de cobertura realizados no edifício da ESTSetúbal/IPS e algumas outras configurações como, por exemplo, os alarmes. Os resultados anotados são analisados e discutidos por comparação com a expectativa resultante da leitura realizada no capítulo dois e com os casos de estudo analisados no capítulo três.

O capítulo seis apresenta a conclusão deste trabalho, resumindo todo o progresso alcançado e as contribuições do projeto para a Instituição. Também se apresenta uma sugestão para de novos tópicos de investigação futura assente no trabalho já desenvolvido.

2. Estado da arte

Neste capítulo é introduzido várias noções, como a *IoT* e a sua importância e benefícios para saúde e qualidade de vida humana. São referenciadas algumas topologias de arquiteturas de redes de sensores, assim como uma análise comparativa entre elas. É feito uma abordagem à noção das redes de sensores, do *Cloud Computing* e *Edge Computing* seguindo para a revisão dos vários protocolos de comunicação sem fios, como o Wi-Fi, Bluetooth, Zigbee e 5G, com especial atenção ao *LoRa* e aos conceitos técnicos de modulação e do funcionamento de uma rede *LoRaWAN*. De modo a entender como os equipamentos e aplicações *LoRaWAN* são identificados dentro de uma rede, é introduzido os vários identificadores como, por exemplo, o DevEUI e AppKey, mas não só, como também, os dois principais métodos de autenticação entre os dispositivos e o servidor. Seguidamente, é apresentado os medidores de energia eletromecânicos e os medidores eletrónicos e a sua classificação ao nível da instalação como invasivos e não invasivos. Por fim, é apresentado o conceito de base de dados relacionais e não-relacionais e o conceito de *API*, terminando o capítulo com uma revisão das tecnologias utilizadas para o desenvolvimento do *backend* e do *frontend* e de ferramentas para a construção de gráficos e interfaces de utilizador, uma vez que, o foco deste trabalho de investigação assenta no desenvolvimento duma arquitetura com uma aplicação *web* para a visualização dos dados.

2.1. Introdução ao IoT

Neste ponto é apresentado a *IoT*, os seus impactos na sociedade, e a sua evolução ao longo dos últimos anos. De seguida é realizado uma breve análise das arquiteturas de referência *IoT*, com a caracterização de cada camada dos diferentes modelos. É feito a classificação de um sensor e a noção da sua importância numa rede e de como permite obter medidas do mundo real. Também é apresentado as topologias das redes de sensores, fazendo uma análise comparativa entre estas. É referido os diferentes protocolos de comunicação sem fios emergentes, relevantes para este projeto, e sua classificação quanto ao alcance e taxa de transmissão de dados relativos. Por fim, é apresentado o conceito de *Cloud Computing* e *Edge Computing*.

2.1.1. Conceito de IoT

A *IoT* é uma estrutura dinâmica que é escalável e automatizada, permitindo que vários equipamentos possam ser ligados para coletar dados e "alimentar" serviços [6]. Isto traz vários benefícios para a qualidade de vida humana, uma vez que o *IoT* melhora significativamente e permite otimizar os processos de negócio e de produção, podendo ser acessível através dos *smartphones*, computadores e outros dispositivos terminais. Um sistema *IoT* deve ser capaz de processar grandes fluxos de dados, provenientes dos sensores, portanto várias características precisam de ser garantidas como a segurança e a privacidade, a viabilidade de acesso ao sistema, e um consumo energético eficiente.

A segurança de um sistema *IoT* deve ser garantida em todos os equipamentos. Por um lado, devem ser usadas autenticações fortes para que apenas utilizadores autorizados possam chegar a um nível de acesso mais administrativo. Assim devem ser usadas criptografias e protocolos de comunicação seguros para que os dados não sejam expostos a terceiros [7]. A localização física dos equipamentos também deve ser de acesso restrito e garantir os meios de segurança de forma a prevenir invasões.

Para além da segurança, um sistema *IoT* deverá estar sempre acessível, ou por outras palavras, deve estar protegido a quedas de energia, por exemplo. Com isto pretende-se que a arquitetura se mantenha disponível e utilizável mesmo que uma das componentes esteja com a desempenho reduzido ou significativamente afetada.

A sustentabilidade de um sistema *IoT* é, também, um tópico que vem a merecer a atenção junto das causas ambientais. A *IoT* veio substituir os métodos tradicionais de sistemas de aquisição de dados, de forma a inovar as indústrias assegurando as novas normas e práticas estabelecidas para a sustentabilidade e preservação do ambiente. O *World Economic Forum* refere que cerca de 84% dos projetos de sistemas inteligentes já estão de acordo com os objetivos globais da sustentabilidade [8]. Isto implica que, para além de ajudar a otimizar os processos de negócio, a instalação e manutenção do sistema não deve impactar o ambiente de forma negativa significativa, antes pelo contrário. Para o efeito, os dispositivos mais recentes estão desenhados para trabalhar menores consumos energéticos e a sua produção recorre a materiais reciclados, o que reduz a pequena pegada ecológica [9].

Em última análise, estes sistemas podem contribuir para o bem-estar das pessoas, para a qualidade de vida ou *quality of life* (*QoL*), que segundo a organização mundial da saúde [10], se define como o gozo da vida de um indivíduo no que respeita aos seus valores, objetivos, expectativas, padrões e preocupações. O grupo da Qualidade Vida da Organização Mundial da Saúde propôs um conceito de *QoL* baseado em quatro domínios principais, a saber, saúde física, psicológica, relações sociais e meio ambiente, que se complementam com outros domínios mais subjetivos ou objetivos como o apoio social, transporte e mobilidade. Um modelo mais completo, proposto pela Eurostat, já define nove dimensões de *QoL*.

O modelo da Eurostat [11] define que a *QoL* consiste nas condições materiais de vida, saúde, educação, atividades produtivas e de valor acrescentado, governação e direitos básicos, lazer e interações sociais, meio ambiente, segurança económica e física e experiência geral de vida. Estas métricas são hoje garantidas através do uso da tecnologia no quotidiano e a relação entre a *QoL* e a tecnologia vem a ser mais reforçada com o passar dos anos e do desenvolvimento tecnológico.

O ponto deste modelo é que a *IoT* não é a base tecnológica da qualidade de vida, mas sim um pilar que, em conjunto com a *World Wide Web* (*WWW*), compõe a *Web of Things* (*WoT*). A *WoT* resolve o problema da fragmentação da *IoT* [12], ou seja, dificuldades com a gestão de equipamentos, integração de aplicativos e conectividade utilizando padrões da web, nomeadamente um conjunto de boas práticas de construção de plataformas e de acessibilidade, assim como de segurança definido pela *World Wide Web Consortium* (*W3C*) [13]. De forma recíproca, a *IoT* permite que a web tenha a perceção e o acesso aos dados do meio ambiente. Outro conceito apresentado neste modelo é o *Quality of Experience* (*QoE*), que é definido como o grau de satisfação de uma pessoa em relação a um serviço ou aplicativo. Por outras palavras, a *QoE* reflete as características percetíveis, reconhecíveis e nomeáveis da experiência do indivíduo perante um serviço. A perceção da segurança é um fator importante para o *QoE* e consequentemente para a *QoL*, pois o utilizador de um serviço precisa de sentir-se seguro face a ameaças e vulnerabilidades do sistema. Ao usar um serviço, estando satisfeito com a sua qualidade, o individuo ganha confiança e utiliza com mais frequência e em maior quantidade, melhorando a sua qualidade de vida. De forma oposta, se o utilizador estiver insatisfeito com o serviço a sua qualidade de vida vai diminuir a longo prazo como consequência do *stresse* e do impacto negativo das suas tarefas [14]. Na Figura 2 encontra-se o modelo definido pela Eurostat.

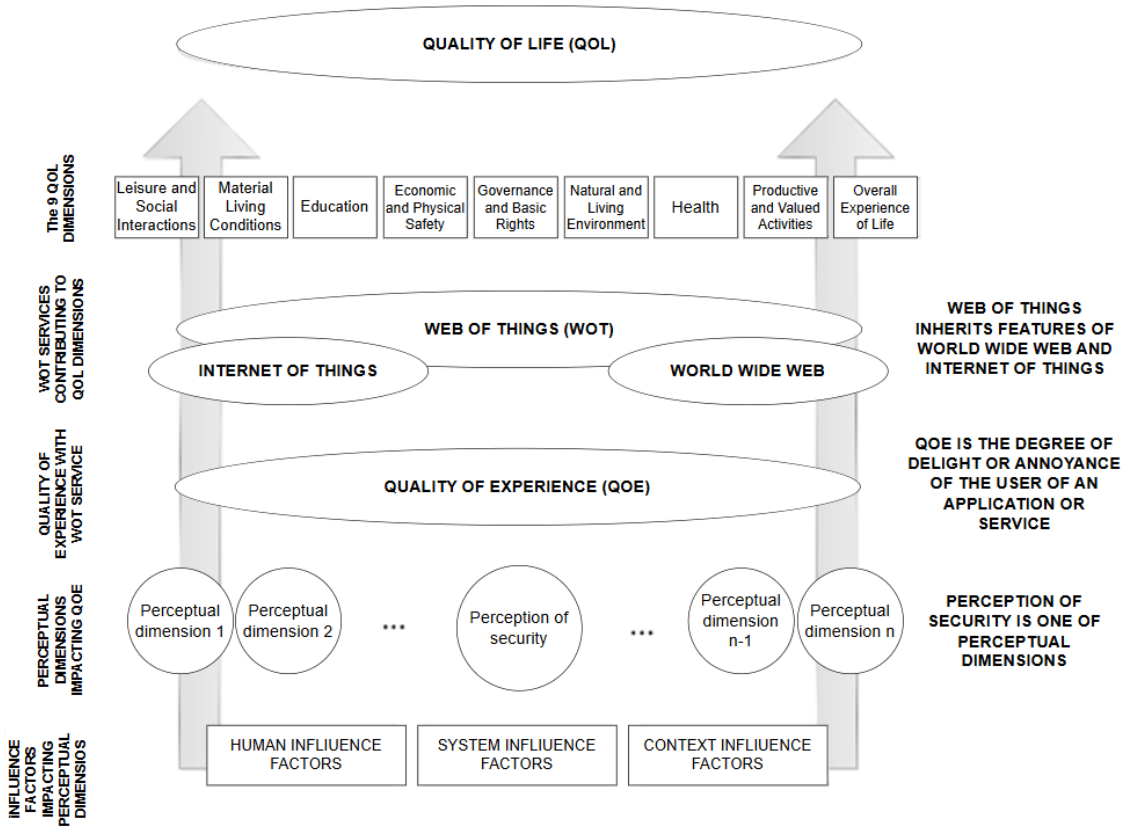


Figura 2 – Modelo de *QoL* definida pela *Eurostat* (Modificado a partir de [11]).

Fruto dos impactos positivos na qualidade de vida e na prestação de serviço, as empresas passaram a aumentar o investimento em tecnologias *IoT* de forma a melhorar a classificação dos seus produtos e aumentar a adesão dos serviços por parte das pessoas. Uma previsão realizada em 2024 [15], sobre a evolução da quota de mercado *IoT* desde 2022 até 2030, confirma esta intenção das empresas de investir mais em tecnologias desta natureza.

No ano de 2022, a quota de mercado rondava os 235×10^9 dólar com a maior parte concentrada em *hardware*, seguindo-se os equipamentos e *software*, e apenas 2% em segurança. No ano seguinte ocorreu um crescimento de 15% chegando à quota de 269×10^9 dólares, com o *hardware* a liderar no que toca ao investimento face aos outros setores do mercado. Ainda segundo [15], uma das principais razões apontadas para o abrandamento na tendência de crescimento diminuição do investimento, ainda que pequena, são as preocupações económicas que têm afetado o mercado tecnológico incluindo o mercado *IoT*.

O estudo presente na Figura 3 estima que haja um crescimento de 12% de 2023 até 2024 e um crescimento médio de 15% de 2024 a 2030, com a China e os Estados Unidos da América na frente no que toca à expansão regional do *IoT*. A quota do mercado irá chegar até aos 690×10^9 dólares em 2030, o que representa um aumento de 389×10^9 dólares face ao ano de 2024, com a especulação de que o setor *software* vai superar o valor de investimento em relação ao investimento *IoT* no geral.

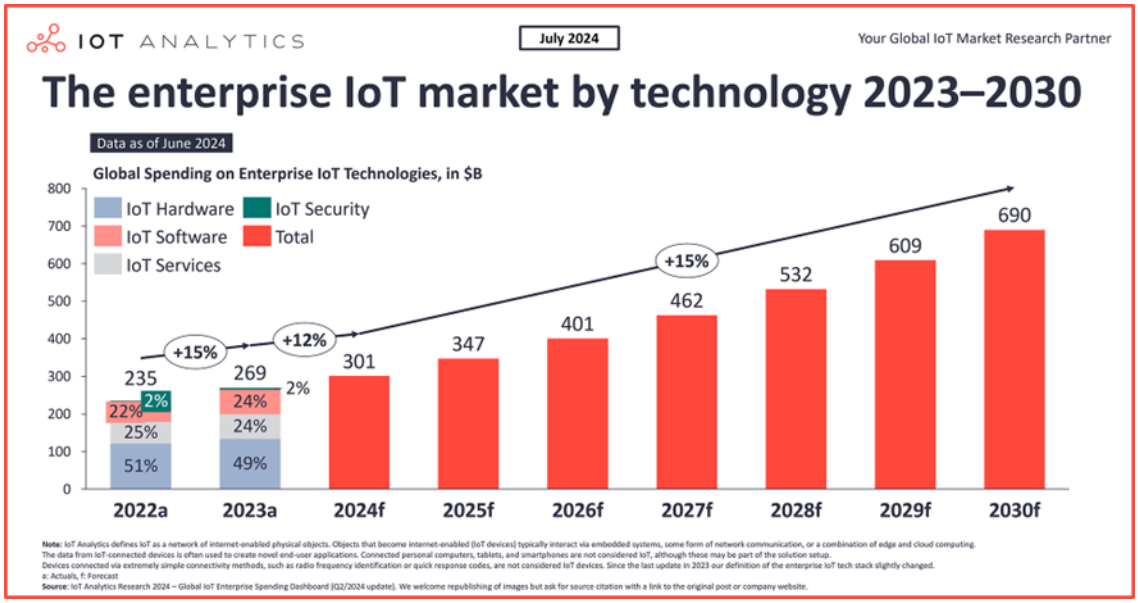


Figura 3 – Evolução da quota de mercado *IoT* de 2022 a 2030 (Extraído de [15]).

Uma outra projeção realizada em 2020 refere que o número de equipamentos *IoT* vai aumentar exponencialmente de 2020 até 2030, começando com cerca de 20×10^9 equipamentos em 2020, até 40×10^9 em 2024 chegando aos 70×10^9 em 2027, e perto dos 120×10^9 equipamentos em 2030.

Um dos principais fatores apresentados nesse estudo para o crescimento são os custos e preços dos mesmos equipamentos, que com o passar dos anos deverão diminuir, e existe um interesse geral em substituir sistemas tradicionais, onde o controlo é manual, por estes que são automatizados e menos dependentes da intervenção humana [16]. Para além disso, outro impulsionador apontado é o melhoramento de processos de negócio e de produtividade, e também a concorrência existente entre diversas empresas que querem oferecer a melhor experiência aos clientes. Em relação à detenção de equipamentos *IoT* per capita, de um ponto de vista regional, o estudo apresenta a China na linha da frente em conjunto com os Estados Unidos da América e, ainda, a Coreia do Sul.

Ao nível de setores da indústria, o estudo refere que a distribuição de equipamentos se concentra na área da manufatura, com 30% dos equipamentos totais. A indústria de produção utiliza essencialmente sensores para fins de análise da integridade em tempo real, de máquinas robóticas e correias transportadoras. O segundo maior setor é o do consumo, onde se encontram os rastreadores de *fitness*, equipamentos com controlo com voz e casas, com o aparecimento de assistentes inteligentes como a Alexa [17] e Assistente Google [18]. Um outro setor emergente e que vem a ganhar muita procura de implementações *IoT* é a saúde, com a introdução de sensores ingeríveis que podem identificar problemas críticos de órgãos internos e ajudar a prevenir doenças.

Os últimos três setores são os transportes com 12%, serviços públicos com 9% e agricultura com 7%. O setor de transportes aposta maioritariamente em veículos autónomos, como é o caso dos novos carros inteligentes da Tesla que oferecem vários níveis de condução autónoma [19]. No setor de serviços públicos, a utilização do *IoT* tem contribuído para a deteção de falhas na pressão de canos e na análise da qualidade da água canalizada. Por último, tem-se o setor da agricultura onde se começa a encontrar novos casos de uso como a medição dos níveis de humidade e azoto no solo, de forma a garantir o rendimento máximo da produção. Na Figura 4 encontra-se a projeção para o mercado de equipamentos *IoT*.

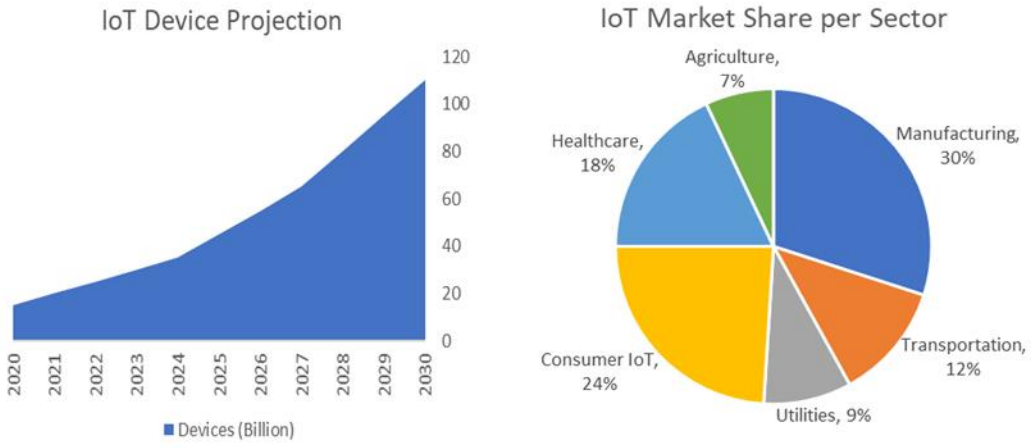


Figura 4 – Projeção para número de equipamentos inteligentes (Extraído de [16]).

2.1.2. Arquiteturas de referência IoT

Um sistema *IoT* não possui atualmente uma arquitetura unanimemente aceite, pelo que os vários modelos existentes são propostos por investigadores [20]. Alguns investigadores defendem um modelo de três camadas, enquanto outros proclamam por um modelo composto de quatro camadas afirmando que a constante evolução do *IoT* não pode ser suportada pelo primeiro. Enquanto isso, considera-se também uma arquitetura de cinco camadas de modo a atender aos requisitos de segurança e privacidade.

O primeiro modelo apresentado nos primórdios do *IoT* é o modelo de três camadas composto pela perceção, rede e camada de aplicação. A camada de perceção é onde se encontra os equipamentos que vão coletar os dados necessários, os sensores são dispositivos sensíveis ao ambiente, projetados para detetar e medir as suas características físicas ou químicas, convertendo-as em sinais utilizáveis para análise e monitorização. A camada de rede, por sua vez, é a camada onde ocorre a transmissão de dados entre a camada de perceção e a camada de aplicação, podendo a transmissão ser feita sobre ou sem fios. É umas das camadas mais importantes de um sistema *IoT*, pois é necessário garantir a confidencialidade e a segurança dos dados que estão a ser transmitidos assim como a estabilidade e a taxa desejável da transmissão.

A camada de aplicação é a camada que define os aplicativos que vão interagir com a tecnologia *IoT*. Nesta camada faz-se o processamento final sobre os dados para que sejam apresentados aos utilizadores de forma clara e objetiva, como por exemplo em gráficos e tabelas. São aplicadas regras de negócio e os dados são restruturados de acordo com cada aplicação, de modo que o indivíduo possa ter a melhor experiência do sistema quer se encontre frente a num computador, ou num *smartphone*, ou num *tablet* [21].

Segundo alguns investigadores a arquitetura de três camadas é muito básica e não acompanha a rápida evolução dos sistemas inteligentes, tendo proposto um modelo com quatro camadas. Este modelo inclui as mesmas camadas do modelo de três camadas, com acréscimo da camada de suporte, que introduziu mecanismos de segurança para o envio de dados entre a camada de rede e a camada de aplicação. A motivação para a criação desta camada baseia-se no facto de no modelo de três camadas o envio de dados entre os sensores e a camada de rede ser direto, o que aumenta a probabilidade de ocorrência de ameaças no sistema.

Com o passar do tempo identificaram-se alguns problemas com o modelo de quatro camadas, nomeadamente em relação à segurança e armazenamento dos dados, pelo que

foi proposta uma terceira arquitetura de cinco camadas. Esta arquitetura é composta pelas três camadas presentes nos modelos anteriores com os nomes de camada de perceção, camada de transporte e camada de aplicação, para além das duas novas camadas de processamento e de negócio. A camada de processamento destina-se a adquirir os dados da camada de transporte e a efetuar a seleção de dados relevantes, formatação e transformação, processo conhecido como *Extract-Transform-Load* (*ETL*). Entende-se por limpeza, o passo de remover dados que sejam irrelevantes, por estarem corrompidos ou por qualquer outro motivo que assim justifique a sua exclusão, de modo a garantir que os dados usados no cálculo de métricas tenham rigor. Os passos de formatação e transformação garantem que os dados e as métricas sejam apresentados na sua unidade de medida normalizada e que sejam preparados para serem enviados para as respetivas aplicações, respetivamente. A camada de negócio tem a função de gerir e controlar aplicativos e não só como, também, a privacidade do utilizador. Esta camada é responsável por aplicar regras de negócio e determinar como os dados devem ser criados, armazenados e alterados. Na Figura 5 apresentam-se os três modelos referidos.

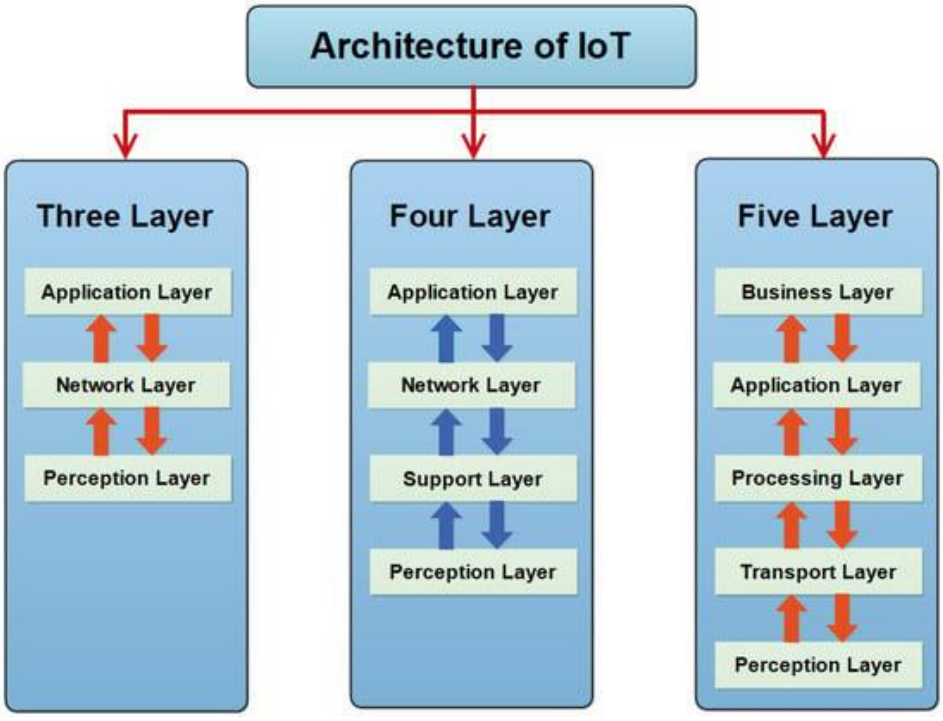


Figura 5 - As três arquiteturas IoT propostas pelos investigadores (Extraído de [20]).

A empresa WSO2 propôs um modelo de referência *IoT* [22]. No entanto, este modelo não se foca em detalhar com uma determinada arquitetura cliente-servidor, pois cada fornecedor de serviços tem o seu conjunto específico de tecnologias. Assim, este modelo foi desenhado para cobrir a maioria dos requisitos de sistemas *IoT* e respetivos projetos e ser um ponto de partida para os desenvolvedores. A arquitetura consiste em cinco camadas, muito semelhante com o modelo de cinco camadas apresentado anteriormente. Este é composto pela camada de dispositivo, a camada de comunicações, a camada de agregação/*Bus*, a camada de processamento e análise de eventos, e a camada de comunicações externas.

A camada de dispositivos está ao nível do *hardware*, tipicamente sensores no contexto *IoT*, em que cada um deve ter um identificador único e comunicação com a Internet, seja de forma direta ou indireta. A camada de comunicações é onde se realizam as ligações, sob diferentes protocolos como o *HyperText Transfer Protocol* (*HTTP*) [23] ou *Constrained Application Protocol* (*CoAP*) [24], dos dispositivos ao destino, e a camada de agregação/*Bus* é responsável por agregar e transformar os dados das diferentes comunicações. A camada de processamento e análise de eventos por sua vez processa e reage a eventos que ocorrem na camada de agregação, podendo ser tomada uma ação ou simplesmente armazenar os dados. Por último tem-se a camada de comunicações externas, onde acontece a interação do utilizador final com a informação do sistema. Para além destas camadas, o modelo de WSO2 ainda distingue duas camadas, transversais, nomeadamente a camada de gestão de equipamentos e a camada de gestão de identidade e acesso. A primeira camada recorre a diversos protocolos de comunicação para comunicar com os equipamentos e controlá-los remotamente, a segunda destina-se a gerir os acessos e segurança do sistema. Na Figura 6 encontra-se o modelo de WSO2.

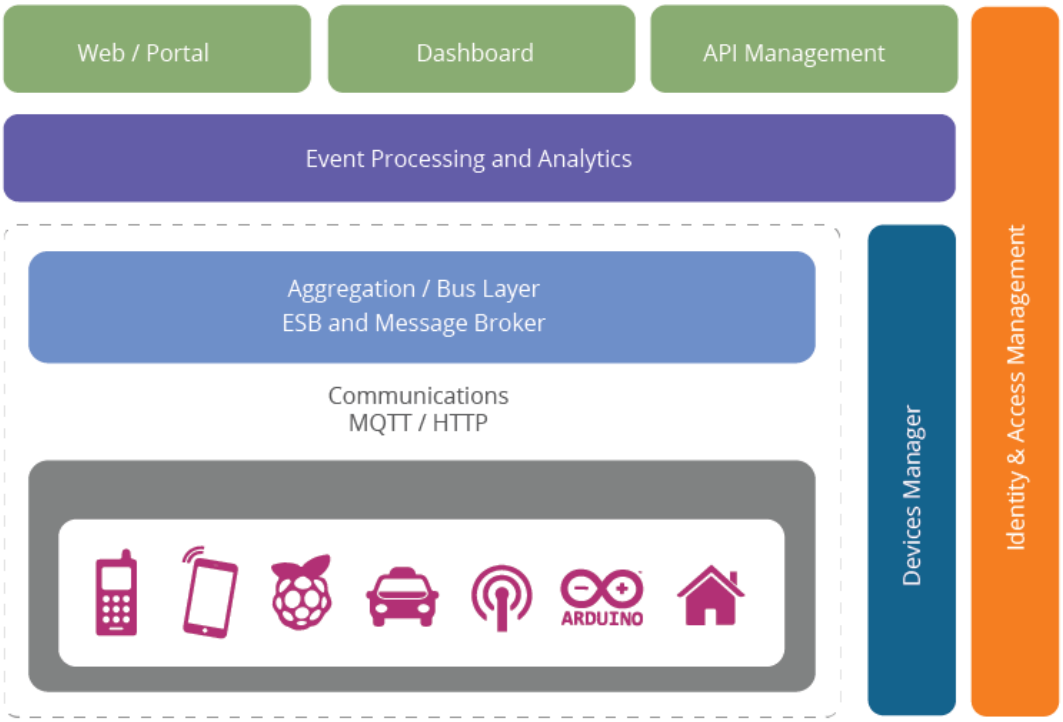


Figura 6 – Arquitetura IoT da WSO2 (Extraído de [22]).

A arquitetura da Figura 7 define um outro modelo de sistema *IoT*, também muito semelhante às arquiteturas anteriores. Esta arquitetura define o sensor como uma componente física que mede parâmetros, ou métricas, do ambiente que o rodeia. Para além deste, o modelo adiciona a camada de atuadores, que é uma componente que permite controlar ou agir sobre o ambiente como ligar e desligar uma lâmpada. A camada de dispositivo representa a componente que integra tanto os sensores como os atuadores, através de *software*. Estes são normalmente ligados a outro sistema, como a camada de *Middleware*. A camada de *Middleware*, por sua vez, é responsável por receber todos os dados dos dispositivos e processá-los.

A ligação entre a camada de dispositivo e a camada de *Middleware* ocorre pela camada intermédia *Gateway*, com o uso de tecnologias de comunicação sem fios, ou diretamente se ambos suportarem o mesmo protocolo de transporte. A camada de aplicação representa o *software* que a camada de *Middleware* usa para obter informações ou para actuar no sistema onde os dispositivos estão ligados [25].

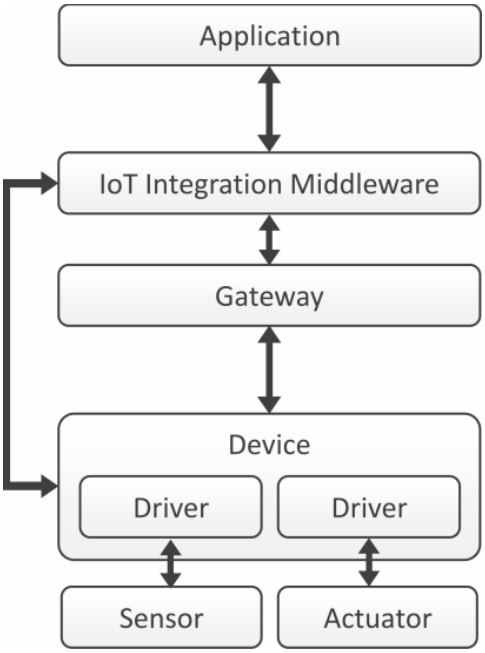


Figura 7 – Arquitetura IoT com atuador (Extraída de [25]).

2.1.3. Sensores e perceção do meio ambiente

O sensor é um dos componentes mais importante de um sistema *IoT*, sendo o responsável por medir e adquirir dados para detetar o estado do meio ambiente. A perceção do meio ambiente é captada através de interfaces eletrónicas que fazem a tradução de um sinal elétrico, que representa o estado do mundo real, para o formato digital [26]. Este também é comumente descrito com ou um transdutor, uma vez que converte uma fonte de energia noutra, ou, neste caso, faz a conversão de um fenómeno físico num sinal elétrico. Na Figura 8 encontra-se o processo de deteção de incêndio por um sensor. Quando ocorre uma mudança do estado de um ambiente, como o início de um incêndio, a temperatura aumenta notavelmente. O sensor monitoriza o ambiente, e pode captar tanto o estado atual da grandeza medida quanto possíveis alterações do meio ambiente que o rodeia, e essa informação é enviada para um computador localizado remotamente.

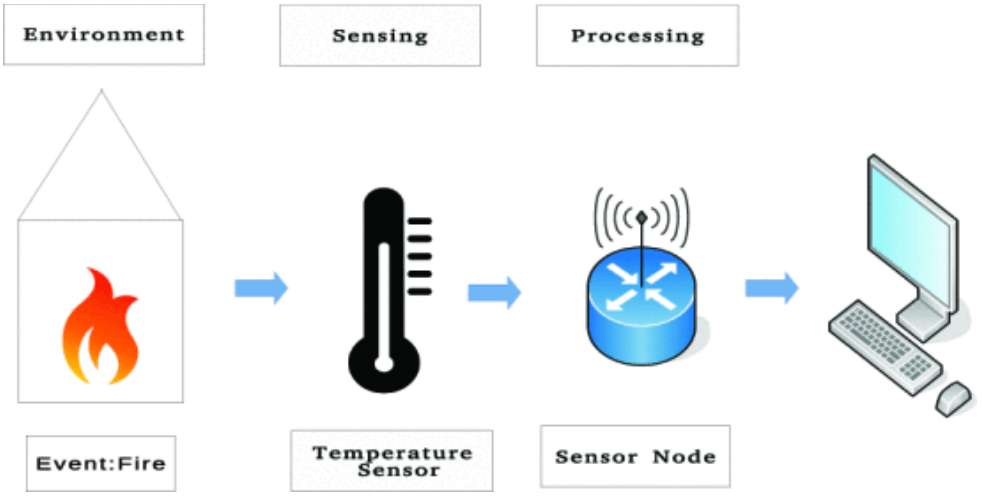


Figura 8 – Sistema básico de perceção de incêndios (Extraído de [27]).

Atualmente existem sensores para extrair diversas características do mundo real, servindo num sistema *IoT* como os sentidos para o ser humano. Ao serem integrados com tecnologias de comunicação e de processamento, rapidamente abrem caminho para soluções automatizadas que podem melhorar a qualidade de vida [27]. Para o efeito, estes devem cumprir com vários desafios nomeadamente terem a precisão adequada e estarem devidamente calibrados para garantir a produção de dados de confiança. Também devem ter uma pegada ecológica reduzida e, portanto, a sua produção, consumo energético e manutenção devem ter impactos reduzidos para o meio ambiente. Por último, mas não menos importante, a segurança dos dados deve ser garantida usando métodos de proteção contra acessos indevidos ao sensor.

2.1.4. Redes de sensores e protocolos de comunicação sem fios

Uma Rede de Sensores Sem Fios (RSSF) é um conjunto de equipamentos chamados de nós sensores, que comunicam entre si ou com nós de gestão da rede e transmitem os dados adquiridos do meio ambiente. A Figura 9 ilustra uma RSSF com base na arquitetura *IoT* de três camadas, nomeadamente a camada de perceção, a camada de rede, e a camada de aplicação. A camada de perceção, onde se encontram os nós sensores, é uma camada de baixa potência enquanto as camadas superiores utilizam uma potência maior para garantir a segurança dos dados [28].

Numa RSSF, os nós realizam a comunicação através de protocolos de transmissão sem fios. Nestas redes, a troca de informação entre dois nós é nomeada de *link*, que é o canal de comunicação estabelecido entre estes para fins como a transmissão de dados, envio de comandos e sincronização da rede.

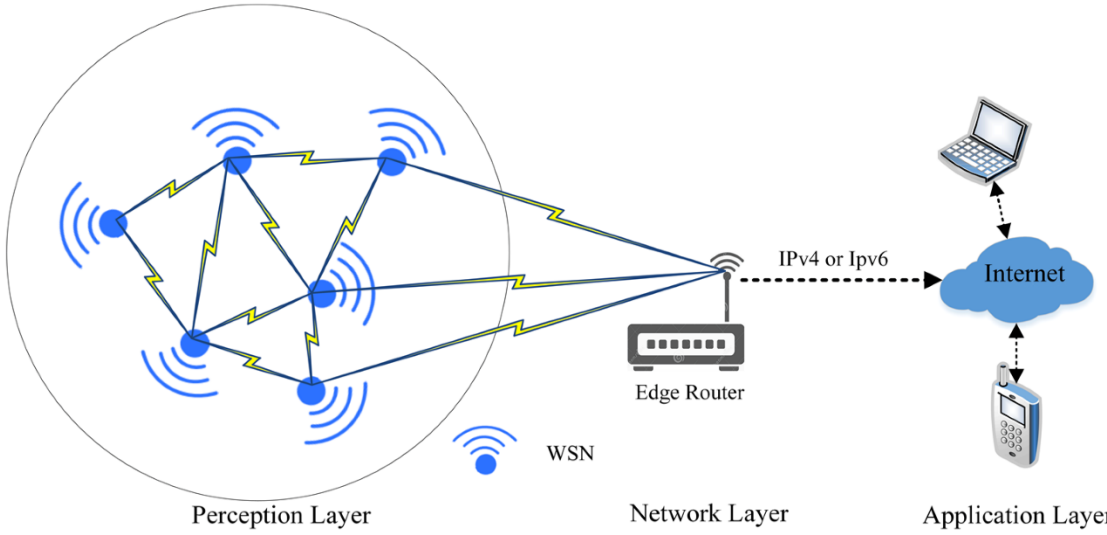


Figura 9 – Arquitetura de três camadas de uma RSSF (Extraído de [28]).

Este é apenas um exemplo simples de uma RSSF, pois uma rede de sensores pode ser construída com diferentes topologias. A IBM define atualmente sete topologias de redes, a saber, a topologia ponto a ponto, a topologia de barramento, a topologia em anel, a topologia em estrela, a topologia em árvore, a topologia de malha, e a topologia híbrida [29].

A topologia ponto a ponto requere apenas que os dois nós sejam ligados por um único *link*, o que apesar de ser fácil de configurar, é inadequado nalguns casos. A topologia de barramento requer que todos os nós sensores sejam ligados ao mesmo *link* central, tornando-se uma topologia económica que permite facilmente a ligação de novos sensores. No entanto, esta estrutura tem como desvantagem a falta de robustez devido à dependência de todos os equipamentos sujeitos ao mesmo *link*. Outra topologia é a topologia em anel que, tal como o nome indica, consiste na ligação circular dos nós. Nesta topologia, os dados circulam apenas numa direção e se a ligação entre dois dos nós se deteriorar toda a rede irá ser impactada. A topologia em estrela, por sua vez, tem todos os nós ligados a um nó central, mas se o nó central estiver inativo toda a rede perde a comunicação. Tal como a rede em anel, esta topologia apresenta uma robustez frágil por depender de um único nó para estabelecer toda a ligação.

A topologia de árvore, por sua vez, resulta da combinação de uma rede de barramento e uma rede estrela. Com isso tem-se as desvantagens das redes de topologia com barramento e estrela, em que a falha de um nó central compromete toda a ligação, mas também tem a vantagem da rede estrela, que é a fácil identificação de problemas com um nó específico.

Também se pode construir a rede ligando todos os nós sensores entre si como na topologia de malha. Esta topologia resolve uns dos principais problemas das redes árvore, barramento e estrela que é a baixa robustez quando o nó central está comprometido, uma vez que ao ligarem-se todos os nós entre si o número de rotas disponíveis é maior. No entanto, este tipo de rede é mais dispendioso para implementar e manter para além do facto de ser mais difícil de configurar devido ao número elevado de ligações. De forma a ter-se uma estrutura que tenha o melhor de todas as arquiteturas, podem combinar-se todas as topologias apresentadas, formando aquilo a que se chama de topologia híbrida. Este tipo de topologia oferece flexibilidade dado que permite adaptar o projeto duma rede às necessidades e especificidades do sistema em causa. Na Figura 10 encontra-se as topologias de redes.

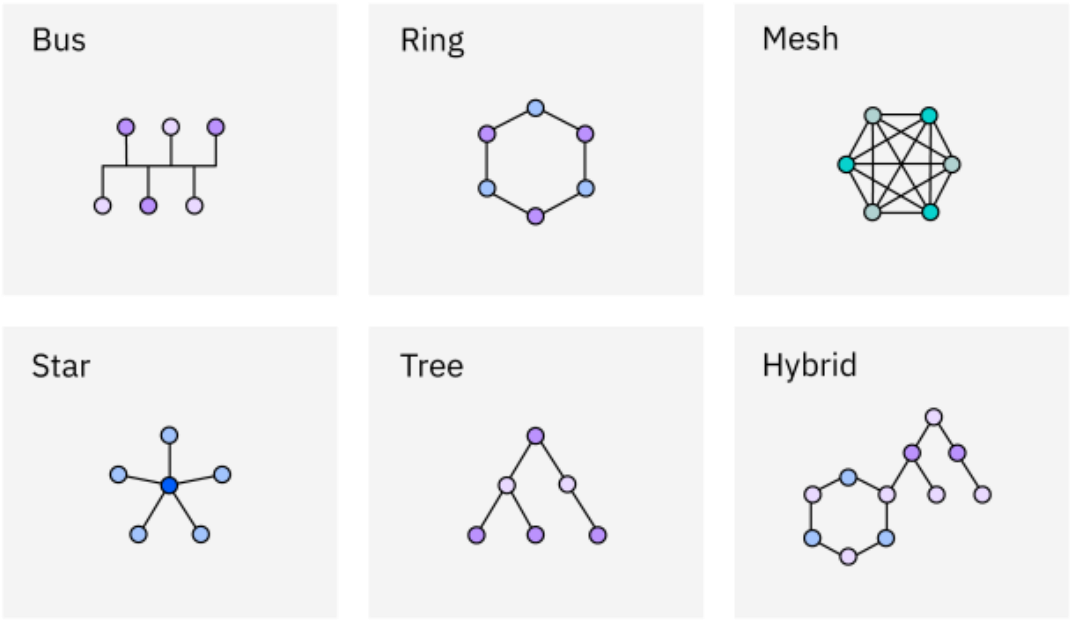


Figura 10 – Topologias de redes (Extraído de [29]).

Para além das topologias, importa também, referir que a forma como se processa a comunicação nos *links* implementados. Existem várias tecnologias para a transmissão de dados entre os nós, que estão classificadas quanto à relação entre o débito binário e a distância de transmissão típicos.

Para comunicações de curto alcance e baixo débito binário, encontram-se as tecnologias de baixo consumo energético Bluetooth e Bluetooth Low Energy (*BLE*) [29], comumente usadas em redes em malha ou redes de estrela. Para transmissões de médio alcance existem tecnologias como o ZigBee [31] e o Wi-Fi [32]. O ZigBee requer baixa potência para transmissão, o que é uma vantagem face ao Bluetooth pois, para sistemas energeticamente autónomos, uma bateria pode alimentar os transmissores por mais tempo a troco da redução do débito binário. De forma oposta, o Wi-Fi permite velocidades de transmissão maiores e com um alcance maior que o ZigBee para além da baixa latência, mas para chegar a este efeito é preciso um maior dispêndio de energia.

Em sistemas que requerem transmissão de dados de longo alcance existem outras opções como o *LoRaWAN* [33], *NarrowBand-Internet of Things* (*NB-IoT*) [34], *Long Term Evolution* (*LTE-M*) [35], *Wi-Fi HaLow* (*Wi-Fi ah*) [36], e o 5G [37]. O *LoRaWAN* é uma tecnologia para comunicação de longo alcance e de baixa potência baseada na modulação *LoRa*, definido em três classes diferentes. A classe 'A' deve ser implementada em todos os equipamentos *LoRaWAN*, enquanto as classes 'B' e 'C' são extensões desta. O conceito e significado destas classes será detalhado no subcapítulo seguinte. A *NB-IoT* é um protocolo *Low-Power Wide-Area Network* (*LPWAN*), também de baixo consumo energético, que aproveita as redes LTE de várias operadoras de telecomunicações para criar uma alta densidade de ligações e cobertura. No entanto, a *NB-IoT* pode ter um maior custo por nó do que nas restantes opções, pois é necessário um contrato com um prestador de serviços de telecomunicações. O *LTE-M* é uma outra vertente do protocolo *LPWAN*, que se torna uma opção relativamente ao *LoRaWAN* e *NB-IoT* pelo facto de permitir um maior débito binário, mas com isso acaba por ter um custo energético superior [38].

A ramificação do Wi-Fi, *HaLow*, trabalha numa frequência de 900MHz ao contrário das outras versões que trabalham em 2.4GHz e 5GHz (802.11n, 802.11ac, 802.11ax). Esta redução da frequência resolve um dos grandes problemas do Wi-Fi, que é justamente o alcance do sinal ser condicionado por objetos e paredes. O *HaLow* é uma opção que faz a mediação entre o alcance e o custo energético, tendo como resultado uma taxa de transmissão menor [39].

A comunicação por satélite é uma tecnologia que permite a transmissão de dados, voz e vídeo a longas distâncias, utilizando satélites em órbita da Terra para receção, amplificação e retransmissão de sinais. Este método de comunicação permite ligar locais remotos, onde as redes terrestres são inviáveis ou economicamente desvantajosas, à *internet*. Um exemplo de algumas soluções com base em satélite e que apareceu recentemente é a Starlink [40].

Por último existe o padrão 5G, que tem permite maior cobertura e baixa latência face à maioria das tecnologias de comunicação referidas, sendo capaz de interligar indústrias e cidades e constituindo uma porta de entrada para a implementação do *IoT* em larga escala. Contudo, a tecnologia 5G ainda é uma tecnologia muito cara à data de escrita deste documento. Na Figura 11 encontra-se as tecnologias de comunicação mencionadas.

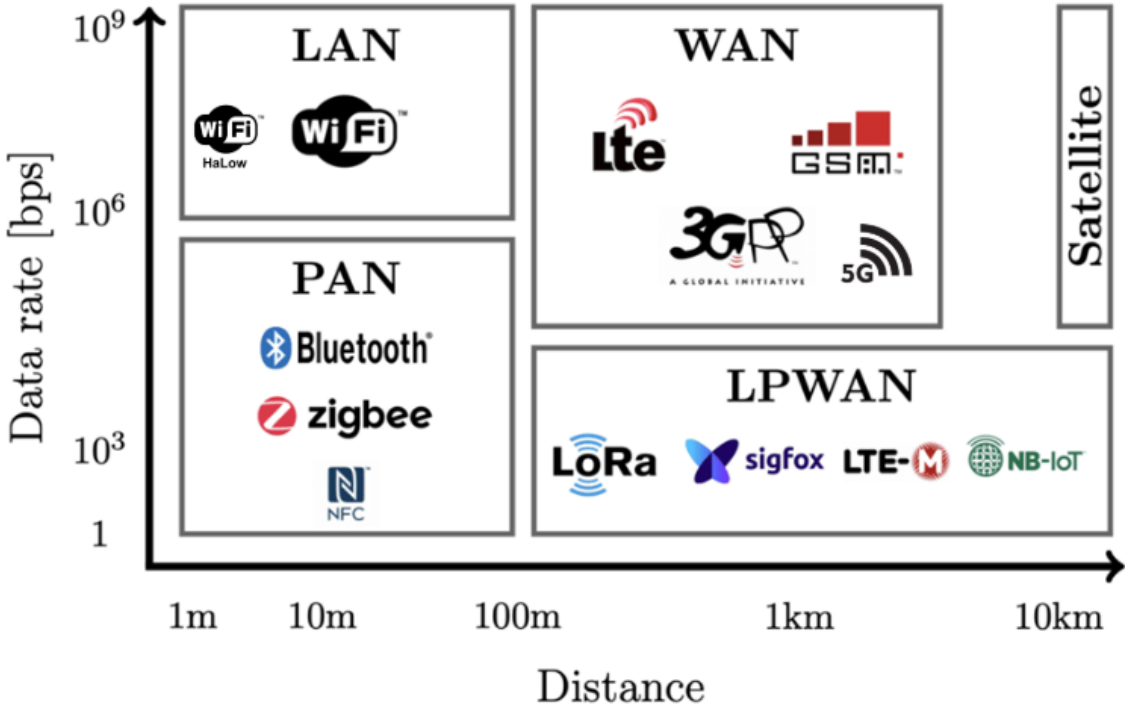


Figura 11 – Tecnologias de comunicação sem fios (Modificado a partir de [41]).

2.1.5. Edge Computing vs Cloud Computing

O *Edge Computing* é um novo paradigma da computação que vem substituir a *Cloud Computing* no que toca à execução de processamento sobre os dados. A *Cloud Computing* é o conceito de adquirir os dados dos sensores e armazená-los num ponto central onde são feitos os cálculos.

A *Cloud Computing* foi introduzida pela Google pela primeira vez em agosto de 2006, usada pelos motores de busca onde começou a mostrar a sua presença. Contundo, com a chegada da *IoT* um grande número de dados tem sido gerado e a largura de banda disponível não tem suportado as necessidades exigidas por estes sistemas [42].

Daí apareceu o conceito de *Edge Computing*, que se trata de focar o processamento dos dados o mais próximo possível ado nível dos sensores, de forma a reduzir a carga aplicada nos servidores e construir-se serviços inteligentes na ”margem” da rede e junto à fonte dos dados. Com isto a baixa latência e a disponibilidade de largura de banda na rede são garantidas devido à descentralização da informação, assim como do seu processamento.

A Tabela 1 apresenta as principais diferenças das duas metodologias. Um dos tópicos apontados é o *stress*a carga submetida sobre a largura de banda disponível ser reduzida no *Edge Computing*, assim como o cálculo e processamento serem efetuados ao nível local, ou por outras palavras, ao nível do nó sensor ou de um servidor periférico na *web*.

Tabela 1 – *Cloud Computing vs Edge Computing* (Extraído de [42]).

	Applicable situation	Network bandwidth pressure	Real-time	Calculation mode
Cloud computing	Global	More	High	Large scale centralized processing
Edge computing	Local	Less	Low	Small scale intelligent analysis

2.2. LoRa

Neste subcapítulo são introduzidos conceitos de *LoRa* e de *LoRaWAN*, uma vez que o ponto central deste trabalho de investigação assenta na utilização destas tecnologias. Inicialmente apresenta-se o conceito de *LoRa* e *LoRaWAN*, explicando a diferença entre as duas nomenclaturas. Segue-se uma introdução teórica sobre a modulação *LoRa*, passando por parâmetros como o *Spreading Factor* e de medidas para a análise da força do sinal.

Introduz-se, ainda, o conceito de classes dos dispositivos e as diferenças entre cada uma delas passando, de seguida, para a noção de servidores de redes *LoRaWAN*. Por fim, definem-se os diferentes identificadores *LoRaWAN* e a sua função dentro da rede, assim como os métodos de autenticação entre os dispositivos e os servidores *LoRaWAN*.

2.2.1. Introdução e conceito de modulação

O *LoRa* é uma tecnologia que permite transmissão da informação a distâncias de até 30km, com o uso da modulação *Chirp* que tem por base o espalhamento do espectro de frequências [43]. Nesta modulação, os dados são transmitidos em *chirps*, onde o *chirp* é um sinal cuja frequência varia linearmente ao longo do tempo entre dois limites que determinam a largura de banda do canal.

A Figura 12 representa o que é um *Chirp LoRa*. Um sinal de frequência que varia linearmente ao longo do tempo entre dois limites definidos composto por símbolos. Os símbolos, que são pequenas fragmentações onde a frequência varia dentro de um *chirp,* estão definidos num intervalo de tempo fixo e representam padrões de bits. Neste caso cada símbolo está definido por dois bits e o sinal *Chirp* representa a informação "001101" ("031" em decimal).

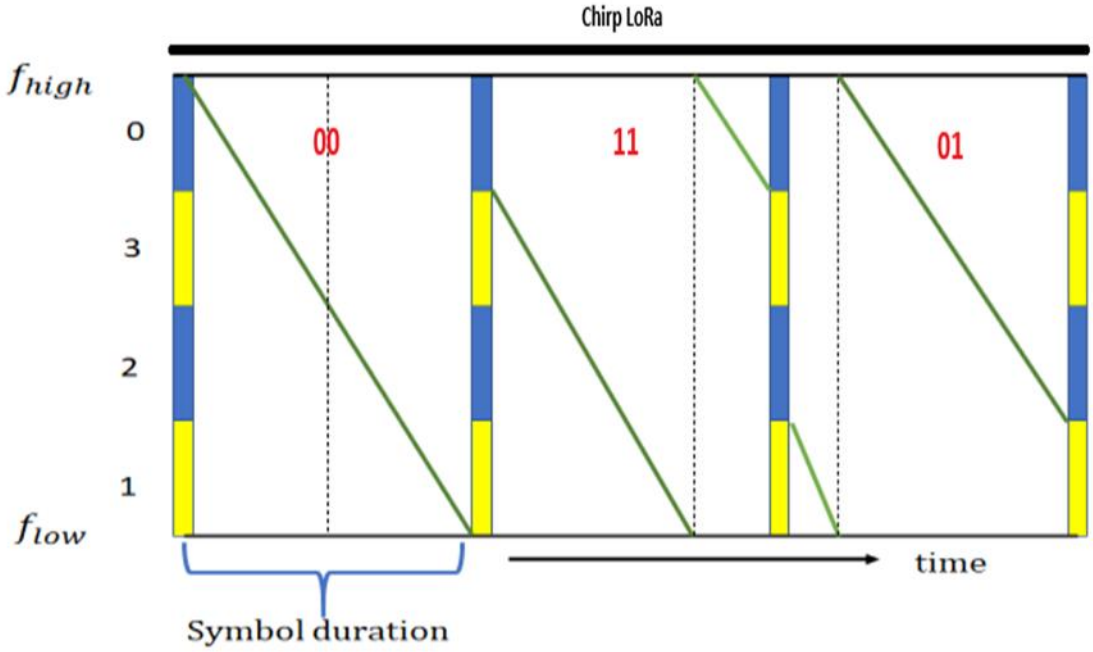


Figura 12 – Chirp LoRa (Modificado a partir de [44]).

A taxa de transmissão da modulação *LoRa* (Rb) [45], está definida pelo produto do *Spreading Factor* (*SF*) com o *Coding Rate* (*CR)* e com a razão entre largura de banda BW (Hz) e a potência de dois elevado a *SF*:

$$Rb = SF \times \frac{BW}{2^{SF}} \times CR \ (bits/segundo) \tag{2.1}$$

onde a largura de banda BW define a janela de frequências onde a transmissão de dados pode ser propagada e o *SF* indica o número de bits que cada símbolo representa num sinal *Chirp*. Por exemplo, para um *SF* de 7 tem-se $2^7 = 128$ bits por símbolo. Isto significa que o alcance e a taxa de transmissão dos dados são condicionados pelo parâmetro *SF* [46]. Por um lado, aumentar o *SF* significa que um símbolo vai representar mais bits e por consequência cada símbolo estará em transmissão por mais tempo, levando à diminuição da taxa de transmissão. De forma oposta, para um *SF* menor um símbolo representa menos bits levando ao aumento da taxa de transmissão. Por outro lado, o aumento do *SF* também implica que a frequência seja espelhada por mais tempo, tornando o símbolo robusto a interferências e ao ruído, enquanto a diminuição do *SF* faz com que o símbolo seja mais compacto no tempo, reduzindo sua robustez a interferências e ruídos.

Para calcular o *SF* é necessário recorrer ao logaritmo de base 2 da razão entre a largura de banda, e a taxa de bits Rs (bits/segundo):

$$SF = log_2 \frac{BW}{Rs} \ (bits/\text{símbolo}) \tag{2.2}$$

O *CR* representa a quantidade de bits para a correção de erros de transmissão, sendo que um *CR* maior aumenta o tempo de transmissão de uma mensagem com a vantagem de tornar a mesma mais robusta. O valor de *CR*, no contexto LoRa, pode ser um do conjunto $\left\{\frac{4}{5}, \frac{4}{6}, \frac{4}{7}, \frac{4}{8}\right\}$, sendo calculado pela razão entre quatro e a soma de quatro e o número de bits adicionais para a recuperação de erros R, com $R \in \{1,2,3,4\}$:

$$CR = \frac{4}{4+R} \ bits \tag{2.3}$$

Substituindo *SF* por 7, BW por 125kHz para um *CR* de $\frac{4}{5}$ em (2.1) tem-se:

$$Rb = SF \times \frac{BW}{2^{SF}} \times CR = 7 \times \frac{125k}{2^7} \times \frac{4}{5} \cong \ 5468 \ bits/\ segundo$$

Por outro lado, substituindo *SF* por 12, BW por 125kHz e um *CR* de $\frac{4}{5}$ em (2.1) obtém-se uma taxa de transmissão menor:

$$Rb = \frac{BW}{2^{SF}} = 12 \times \frac{125k}{2^{12}} \times \frac{4}{5} \cong 292\ bits/segundo$$

Estes parâmetros são configurados ao nível do dispositivo final, onde o *CR* padrão indicado para qualquer versão *LoRaWAN* na região da união europeia 868MHz é de $\frac{4}{5}$ para qualquer *SF* entre 7 e 12.

A potência do sinal recebida no recetor, é defina pelo *Received Signal Strength Indicator* (*RSSI*). Esta medida encontra-se em dBm e o seu valor é negativo, estando tão mais próximo do zero quanto maior for a potência do sinal [47].

O *Signal-to-Noise Ratio* (*SNR*) é a razão entre a potência do sinal recebido e o ruído na escala linear, ou é dada pela diferença das duas grandezas em escala logarítmica:

$$SNR\ (dB) = P_{Sinal_{recebido}}(dBm) - P_{ruído}(dBm) \tag{2.4}$$

Através da Figura 13, é possível visualizar um exemplo que ilustra medidas de potência de um sinal transmitido assim como da potência do ruído. O *RSSI* do sinal transmitido é de -65dBm enquanto a potência do sinal de ruído é de -90dBm.

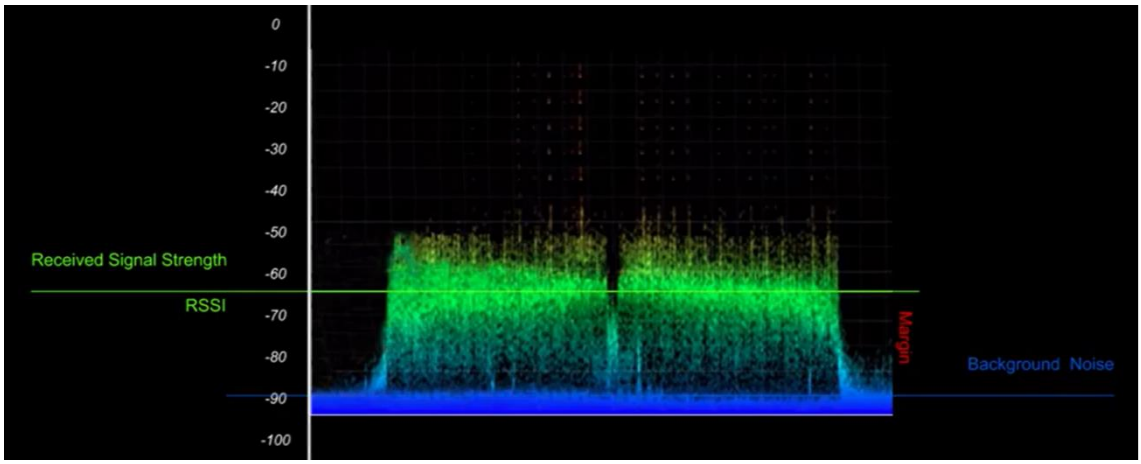


Figura 13 – Sinal obtido num recetor LoRa (Extraído de [47]).

Aplicando estes dados em (2.4) obtém-se um *SNR* positivo o que significa que o recetor seria capaz de demodular este sinal sem qualquer dificuldade:

$$SNR\ (dB) = P_{Sinal\_recebido} - P_{ruído} = (-65) - (-90) = 25dB$$

No entanto, o *LoRa* consegue demodular sinais em que o *SNR* é negativo, desde que este tenha um valor mínimo necessário para a demodulação. O valor mínimo de *SNR* para cada *SF*, de modo que o recetor seja capaz de fazer a demodulação do sinal encontra-se na Tabela 2.

Tabela 2 – Valores mínimos de *SNR* (Extraída de [47]).

SpreadingFactor (RegModulationCfg)	Spreading Factor (Chips / symbol)	LoRa Demodulator SNR
6	64	-5 dB
7	128	-7.5 dB
8	256	-10 dB
9	512	-12.5 dB
10	1024	-15 dB
11	2048	-17.5 dB
12	4096	-20 dB

Quando um sinal é demodulado no recetor, encontra-se uma trama *LoRa* como a da Figura 14. Uma trama *LoRa* é constituída por cinco elementos dos quais o preâmbulo, o *Physical Header* (*PHDR*), *Physical Header Cyclic Redundancy Check* (*PHDR_CRC*), o *Physical Payload* (*PHYPayload*) e o *Cyclic Redundancy Check* (*CRC*) [48].

O preâmbulo é usado para sincronizar o recetor com o transmissor, enquanto o *PHDR* e *PHDR_CRC* são usados para distinguir a carga útil da trama e verificar a integridade do *PHDR*, respetivamente. O *PHYPayload* e o *CRC* são usados para transportar os dados enviados do sensor e para fazer a deteção de erros do *PHYPayload*, respetivamente.

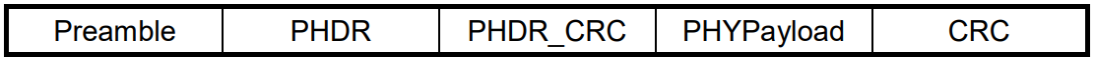


Figura 14 – Trama LoRa (Extraído de [48]).

2.2.2. Servidores de Redes LoRaWAN

A *LoRaWAN* é a normalização que ocorre na adição da camada do modelo OSI *Media Access Control* (*MAC*) *LoRaWAN*. Nesta camada identifica-se fisicamente e exclusivamente um equipamento dentro da rede. As redes *LoRaWAN* baseiam-se na topologia de rede em estrela onde cada nó se liga a um nó central, o que favorece a um consumo energético menor de baterias [49]. A estrutura de uma rede *LoRaWAN* é tipicamente constituída por quatro elementos: os nós, o *gateway*, o servidor da rede e o servidor de aplicações.

Os nós compreendem-se como sensores, atuadores, ou aplicações que fazem o controlo ou a monitorização do ambiente. Dentro da rede, o *gateway* é o responsável por receber informação provenientes dos nós e enviá-los para o servidor da rede, através de protocolos como o Wi-Fi. Por sua vez, o servidor de rede é onde se acumula toda a informação originada nos vários nós da rede e onde a sua integridade é verificada. A partir do servidor de rede podem ser interligadas aplicações finais onde os utilizadores podem ter acesso aos dados ou a métricas deles extraídas. Na Figura 15 encontra-se uma estrutura típica de uma rede *LoRaWAN*, que ilustra a arquitetura descrita acima.

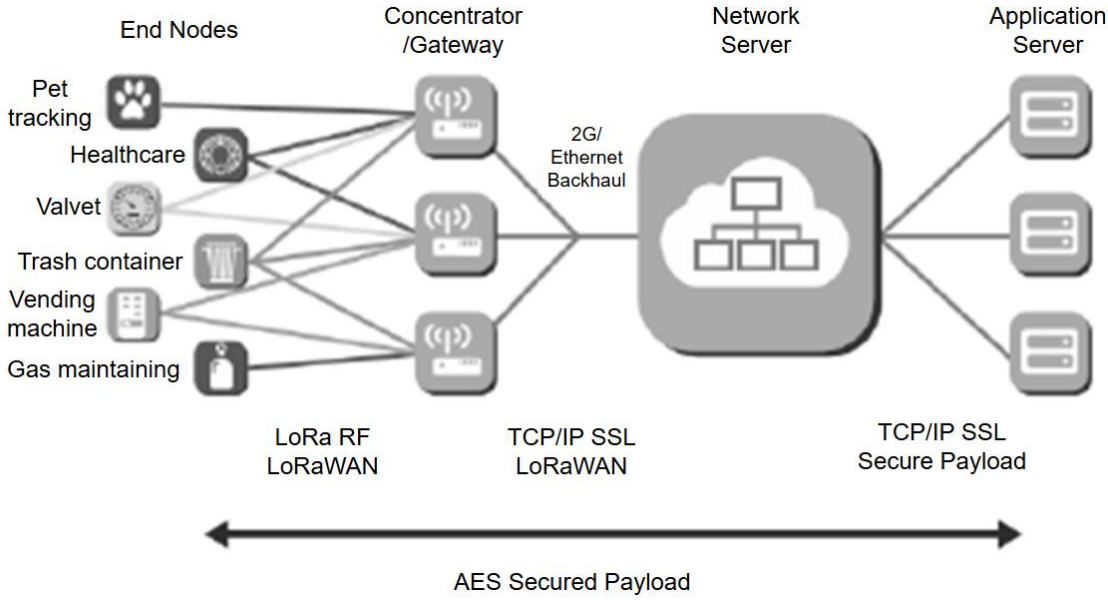


Figura 15 – Estrutura genérica de uma rede *LoRaWAN* (Modificado a partir de [49]).

A transmissão dos dados está apenas ao nível da camada física, representado o meio físico na forma de uma onda eletromagnética onde os dados são transmitidos. A *LoRaWAN* está definida pelas camadas de parâmetros regionais e a camada de *link* [50]. A camada de parâmetros regionais contém informação sobre a frequência da comunicação, assim como de outros fatores técnicos para manter a eficiência da transmissão e o cumprimento das normas locais. Por outro lado, a camada *link* é usada para gerir a comunicação entre equipamentos dentro da rede, como os sensores e o *gateway*. Dentro de um servidor de rede *LoRaWAN*, a camada *LoRa* não existe, assim como a camada *LoRaWAN* para os parâmetros regionais, sendo estas substituídas por camadas de *backend* para definir o *Network server*, o *Join server*, e o *Application server*.

A camada *Network server* é usada para gerir e coordenar a comunicação dos dispositivos dentro da rede, enquanto o *Join server* é responsável pela ativação e autenticação destes. O *Application server* processa os dados e entrega às aplicações finais por meio de integrações. Na Figura 16 ilustra-se as camadas de comunicação de uma rede *LoRaWAN*.

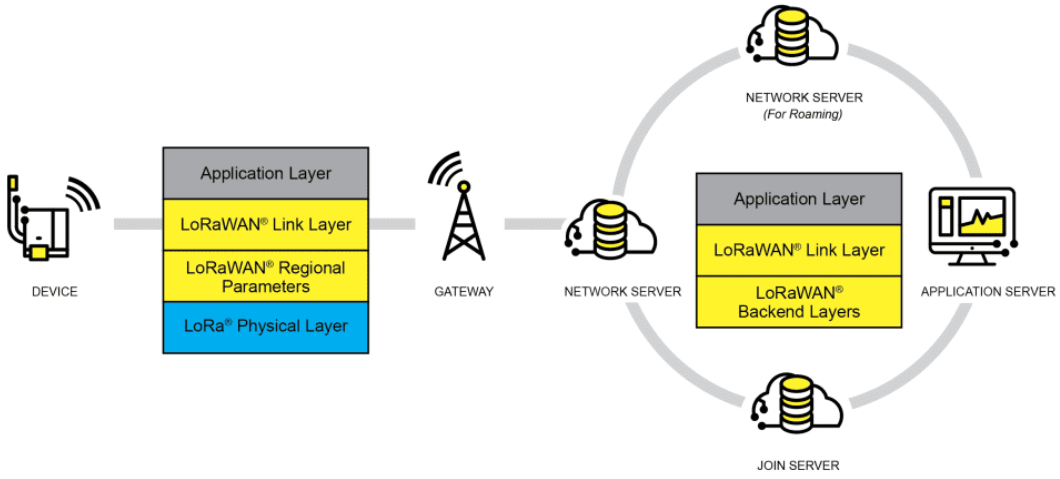


Figura 16 - Camadas do modelo *LoRaWAN* (Extraído de [50]).

Um servidor de rede *LoRaWAN* é o responsável por gerir a comunicação entre os dispositivos finais como sensores e atuadores, e as aplicações finais [51]. Este trata de coordenar os dados dos dispositivos para o servidor e do servidor para o dispositivo, regulando também a rede para otimizar a largura de banda. Um exemplo de plataformas que fornecem um servidor de redes *LoRa* é o The Things Stack da The Things Industries.

O The Things Stack oferece ferramentas e serviços para gerir a conectividade de dispositivos *LoRaWAN*, dando suporte desde pequenas implementações até redes comerciais complexas [52]. A plataforma está disponível em várias edições, incluindo uma versão comunitária gratuita e versões empresariais, que oferecem suporte mais completo, maior escalabilidade, e mais opções de integração. O The Things Stack permite gerir dispositivos fornecendo ferramentas para registar, autenticar e configurar dispositivos de forma simplificada como também tem suporte para conectividade com serviços externos, permitindo enviar dados de dispositivos para aplicações personalizadas através de *webhooks*, *MQ Telemetry Transport* (*MQTT*), ou API REST. Sendo também importante numa rede *IoT*, o The Things Stack suporta redes privadas e públicas com opções de encriptação e autenticação para garantir a segurança dos dados.

A plataforma é atualmente utilizada tanto por desenvolvedores de *IoT* quanto por empresas que precisam de uma solução confiável para implementar redes LoRaWAN personalizadas e seguras. Existem opções de código aberto para implementar um servidor de redes LoRaWAN como o ChirpStack. Este foi desenvolvido com foco na flexibilidade e personalização para atender às necessidades de diferentes casos de uso de *IoT* [53]. O ChirpStack oferece muitas ferramentas de gestão de dispositivos e integração, como o The Things Stack, e é amplamente usado em ambientes onde o controlo total sobre a rede e a privacidade dos dados são prioridades, como em instalações industriais e cidades inteligentes.

2.2.3. Classes de dispositivos

A especificação LoRaWAN define três classes de dispositivos: 'A', 'B' e 'C'. Todos os dispositivos devem implementar a classe A, sendo as classes B e C uma extensão desta. Na Figura 17 encontra-se o modelo de transmissão de um equipamento LoRa de classe A. Este envia uma mensagem do sensor para o servidor de rede (*uplink*) e quando a transmissão é concluída o dispositivo abre uma janela de receção chamada de RX1 para receber informação da rede para o sensor (*downlink*). Se o servidor de rede não responder durante essa janela de tempo, RX1 é fechado e um novo canal RX2 é aberto. O dispositivo espera novamente para receber uma mensagem, mas caso o servidor não responda no fim da janela de tempo, então o próximo *downlink* é agendado imediatamente após a seguinte transmissão de *uplink* [54]. Uma das principais vantagens dos dispositivos de classe A é que estes passam maior parte do tempo suspensos, o que favorece o uso de alimentação por bateria. Contudo, a latência de *downlink* é alta, pois é necessário o envio do *uplink* antes de poderem receber um *downlink*.

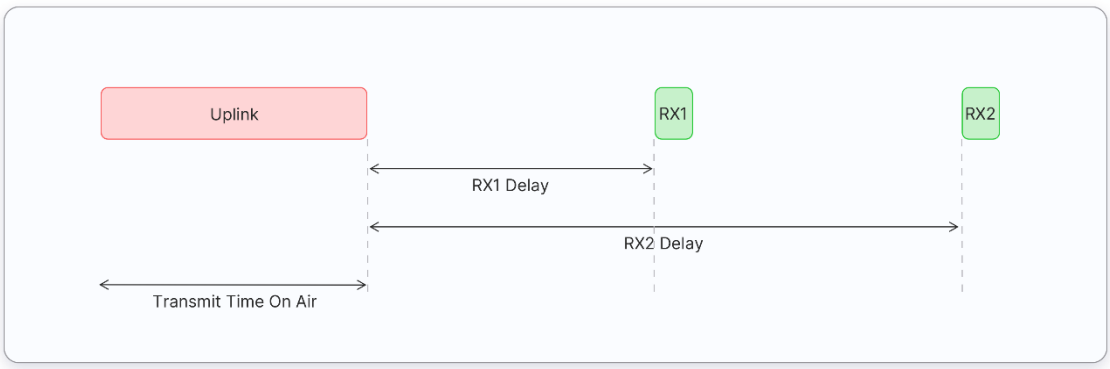


Figura 17 - Funcionamento de um dispositivo LoRa de classe A (Extraído de [54]).

Os dispositivos de classe B, Figura 18, em extensão da classe A, abre uma janela inicialmente para receber um *beacon* enviado pela rede. Este *beacon* permite que o dispositivo final e o servidor estejam sincronizados temporalmente de forma que a rede saiba o momento exato para enviar um *downlink* para um ou mais dispositivos. O tempo entre os dois *beacons* é denominado de período de *beacon*. Após esta sincronização inicial com *beacon*, o dispositivo envia o *uplink* para a rede e repete o processo das janelas temporais da classe A. Em comparação com a classe A, os dispositivos de classe B apresentam consumos energéticos maiores por estarem mais tempo ativos devido à sincronização inicial. No entanto, a latência de *downlink* é menor comparativamente à classe A devido ao sincronismo com o *beacon*.

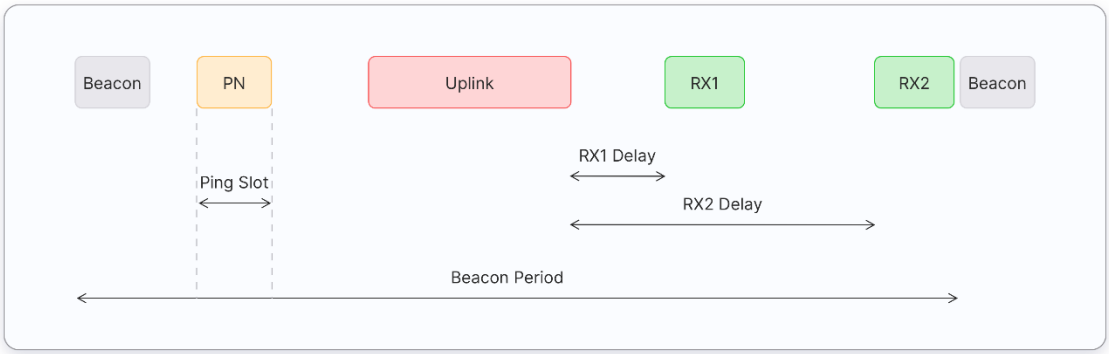


Figura 18 - Funcionamento de um dispositivo LoRa de classe B (Extraído de [54]).

Os equipamentos de classe C, Figura 19, por sua vez, não recorrem a um *beacon* para sincronismo, mas encontram-se continuamente atentos a mensagens de *downlink* do servidor até que seja agendado um *uplink*. Quando estes enviam um *uplink*, é aberta uma janela curta RX2. No fim desta janela, é aberta a janela RX1. Por fim, a janela RX1 é fechada e é aberta novamente a janela RX2 que vai estar continuamente atenta à rede para receber um ou mais *downlinks*. Esta janela é fechada assim que for necessário enviar um *uplink*. A classe C apresenta latência de *downlink* menor quando comparada com as classes A e B, mas a necessidade de estar permanentemente no estado ativo faz com que a alimentação por baterias externas requeira maior capacidade destas para uma mesma autonomia, uma vez que o consumo energético é maior comparado com as classes A e B.

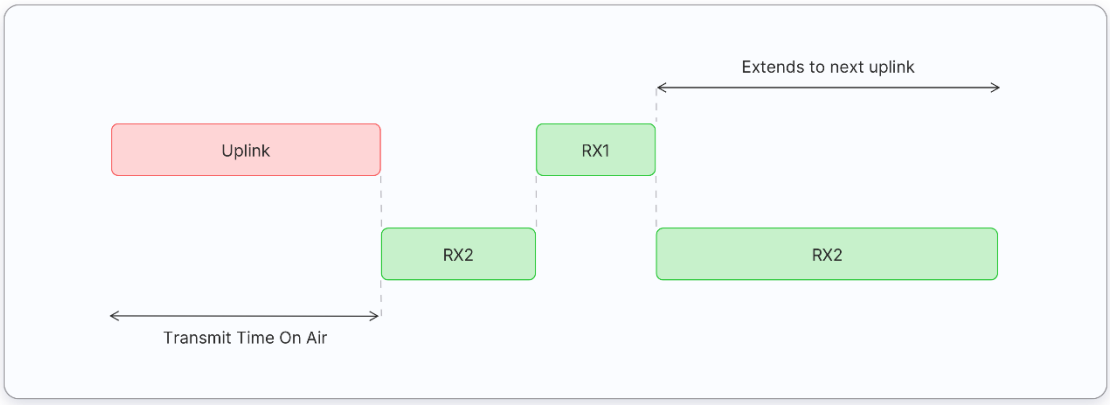


Figura 19 - Funcionamento de um dispositivo LoRa de classe C (Extraído de [54]).

2.2.4. Identificação dos equipamentos e aplicações

Na *LoRaWAN*, à semelhança das outras redes, existem identificadores para os equipamentos ligados na rede [55]. Começando pelo DevEUI que é um identificador único de 64 bits e destina-se a identificador um dispositivo final ligado na rede. Este está associado ao microprocessador *LoRa* pela fabricante. O DevAddr, ou JoinEUI, é um identificador não exclusivo de 32 bits, ou seja, vários equipamentos podem receber o mesmo DevAddr. Este é atribuído aos dispositivos quando se ligam ao *Join server*. A AppEUI, assim como o DevEUI, é um identificador único de 64 bits e serve para identificar exclusivamente uma aplicação criada dentro de um servidor de rede LoRa. Os *gateways* também têm um identificador único de 64 bits GatewayEUI.

2.2.5. Registo de dispositivos

Quando um dispositivo se liga ao *Join server* são trocadas informação para realizar a autenticação. Existem dois métodos de autenticação, nomeadamente o *Over-The-Air-Activation* (*OTAA*) e o *Activation By Personalization* (*ABP*) [56].

O método de autenticação *OTAA* é o método mais seguro e indicado para autenticar dispositivos, uma vez que é feito um pedido de *join-request* para o *Network server* e, a partir deste ponto, as chaves de segurança são negociadas através do *Join server*, retornando um *join-accept* no caso de a autenticação com o servidor ter sido sucedida. A Figura 20 ilustra o processo de autenticação de um equipamento numa rede LoRaWAN.

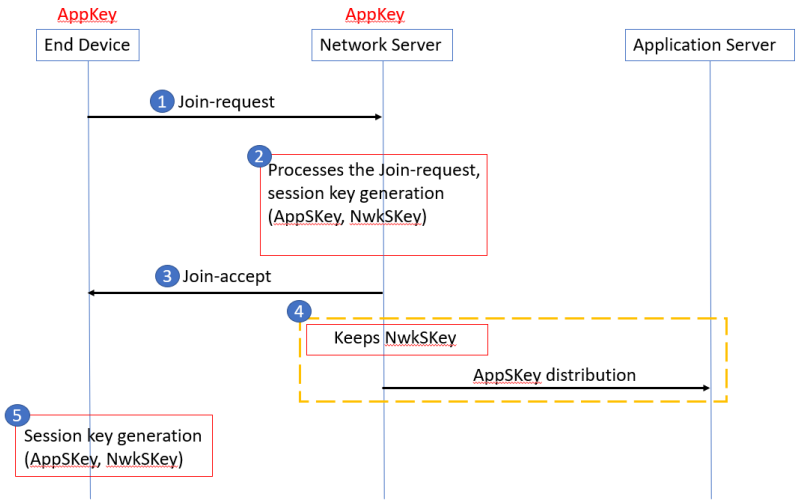


Figura 20 – Processo de autenticação *OTAA* numa rede LoRaWAN (Extraída de [56]).

Por outro lado, a autenticação *ABP* é configurada manualmente e, portanto, é menos segura, pois as chaves são estáticas até ao fim de vida do dispositivo e podem ser descobertas por um atacante. A autenticação *OTAA* com o *Network server* é ignorada e com isso o *Join server* não é invocado neste processo. Na Figura 21 encontra-se o processo de autenticação *ABP* numa rede LoRaWAN e é possível ver que as chaves não foram negociadas, uma vez que já foram configuradas manualmente.

Figura 21 - Processo de autenticação *ABP* numa rede LoRaWAN (Extraída de [56]).

2.3. Medidores de energia

Neste ponto é apresentado uma análise dos tipos de medidores de energia existentes no mercado, explicando o seu método de funcionamento e as suas aplicações típicas. Posteriormente, é realizado a classificação dos medidores quanto ao tipo de instalação na rede elétrica.

2.3.1. Tipos de medidores de energia

Um medidor de energia pode recorrer a diferentes tecnologias para a recolha de sinais que permitem determinar grandezas elétricas relevantes. Até à data de escrita deste documento, distinguem-se dois principais tipos de medidores de energia, nomeadamente os medidores eletromecânicos e os medidores eletrónicos [57]. Os medidores eletromecânicos possuem duas bobinas de indução, que produzem um campo magnético num disco de metal, quando a corrente elétrica passa pelo medidor. A rotação deste disco varia proporcionalmente com a corrente que flui no circuito e, porque a tensão da rede é constante, à potência instantânea, e estas rotações são contadas de forma para obter a energia consumida. Apesar dos medidores eletromecânicos terem demonstrado uma veracidade dos dados suficiente para atender às exigências da sua época, a evolução das necessidades na gestão da rede elétrica e o desenvolvimento de novas metodologias de monitorização levaram à sua gradual substituição ao longo dos anos.

O medidor eletrónico é construído com base em dispositivos eletrónicos para realizar a leitura dos sinais elétricos da instalação e, assim, determinar o consumo energético. Este tipo de medidor abriu portas para novos sistemas de monitorização, uma vez que permitem adicionar novas funcionalidades através da instalação de diversos módulos na mesma instalação. A junção de sistemas de medição de consumos energéticos e de deteção de anomalias baseados em sensores eletrónicos permite uma nova abordagem à gestão de instalações elétricas que não seria possível realizar com os convencionais medidores eletromecânicos.

2.3.2. Medidores invasivos e não-invasivos

Os medidores podem ser classificados quanto ao seu tipo de montagem numa instalação elétrica. Os medidores invasivos são aqueles que requerem a interrupção do circuito elétrico, encontrando-se nesta categoria a maioria dos medidores tradicionais [58] . Tipicamente estes medidores são ligados em série com a carga elétrica para obter a corrente do circuito, o que requer a abertura do mesmo.

Por outro lado, um medidor não-invasivo não requer a interrupção do circuito como é o caso das pinças amperimétricas (*Clamp meters*) [59]. Estes medidores não são ligados em série na carga como os medidores elétricos convencionais, mas sim ligados através de uma pinça com um núcleo de material ferro-magnético. A leitura da corrente elétrica é realizada através de um circuito como o apresentado na Figura 22.

A corrente a ser medida passa pelo enrolamento primário e o campo magnético criado é medido pelo sensor de efeito Hall, cuja saída representa uma tensão proporcional ao campo magnético, que é amplificada e convertida num sinal de corrente que passa pelo enrolamento secundário.

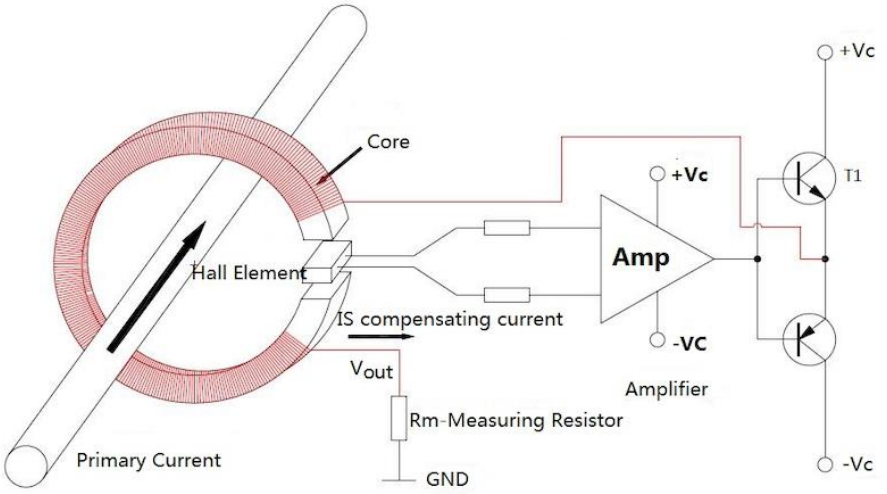


Figura 22 – Circuito de *loop* fechado para a medição da corrente (Extraído de [60]).

A corrente no segundo enrolamento cria um campo magnético inverso ao criado pela corrente no enrolamento primário. Este equilíbrio deve-se à malha de *feedback* composta pelo sensor de efeito de Hall, o amplificador e o restante circuito de transístores. Uma vez que o campo magnético total é zero:

$$N_p E_{up} = N_s E_{us} \tag{2.5}$$

onde N_p é o número de voltas do enrolamento primário, E_{up} a força eletromotriz do enrolamento primário, N_s o número de voltas do enrolamento secundário e E_{us} a força eletromotriz do enrolamento secundário.

Assumindo $N_p = 1$ e sabendo que $V_{out} = R_m \times E_{us}$ em (2.5) obtém-se a tensão à saída do medidor:

$$N_p E_{up} = N_s E_{us} \,(=)\, E_{up} = N_s E_{us} \,(=)\, E_{up} = N_s \frac{V_{out}}{R_m} \,(=)\, V_{out} = E_{up} R_m \frac{1}{N_s}$$

2.4. Tecnologias de processamento e interface gráfica

Neste ponto é feito uma revisão do conceito de base de dados relacional e não-relacional, *API* e as tecnologias utilizadas para o desenvolvimento de aplicações *web* com interfaces gráficas. A leitura deste subcapítulo permite conhecer algumas das ferramentas modernas usadas para a construção de sistemas integrados, que ligam os utilizadores aos dados numa forma objetiva e segura.

2.4.1. Base de dados

Uma base de dados é uma coleção de dados, onde os dados são armazenados e acedidos por um computador. A base de dados é gerida por um motor, *DataBase Management System* (*DBMS*), que manuseia os dados e que insere, atualiza, apaga e extrai a informação [61]. Existem várias categorias de bases de dados, que diferem entre si na forma como armazenam a informação, quanto ao formato dos dados e às relações entre este. Destes, destacam-se as bases de dados relacionais e as bases de dados não-relacionais.

Uma base de dados relacional é uma coleção de dados organizados em tabelas, a partir das quais os dados podem ser acedidos ou reagrupados de várias maneiras. As tabelas podem relacionar-se entre si, o que pode ser representado por uma tabela exclusiva para essa relação.

Ao criar-se uma tabela aplica-se um intervalo de valores possíveis juntamente com restrições aos dados [62]. Cada entrada de uma tabela contém uma instância única de dados, sendo diferenciada das demais com um ID exclusivo. No caso da base de dados relacional, o DBMS executa comandos *Structured Query Language* (*SQL*) para aceder e modificar os dados armazenados. Existe uma grande variedade de base de dados relacionais como o MySQL [63], a Oracle [64], PostgreSQL [65], SQLite [66] e MariaDB [67].

Uma base de dados não-relacional é um conjunto de sistemas que fazem a gestão dos dados e que difere dos sistemas relacionais em muitos aspetos. O principal aspeto que diferencia as bases de dados relacionais das bases de dados não-relacionais é a inexistência de relações entre tabelas e, portanto, a junção dos dados não é possível. Outros fatores que a diferenciam são que não utiliza SQL como sua linguagem de DBMS.

No contexto de NoSQL, os dados são armazenados em formatos estruturados, como documentos, *JavaScript Object Notation* (*JSON*) e *Extensible Markup Language* (*XML*) por exemplo, ao invés de tabelas. Dentro das várias opções para base de dados não-relacionais tem-se o MongoDB [68], Cassandra [69] e Firebase [70].

As bases de dados relacionais seguem as regras Atomicidade, Consistência, Isolamento, Durabilidade (ACID) e possuem um esquema de entidades predefinido, geralmente complexo de aplicar alterações quando assim for necessário. Estas tornam-se uma opção para projetos em que se sabe a dimensão das entidades e que não visem grandes alterações na estrutura dos dados como em modelos de negócio, que requerem registos de transações confiáveis. Por outro lado, as bases de dados não-relacionais têm uma maior flexibilidade no esquema de entidades, uma vez que o conceito de relação não existe. Isto permite que possam ser usadas em projetos onde a dimensão e a estrutura dos dados são alteradas frequentemente, como em sistemas *IoT* e aplicações *web* onde as métricas e os dados são volumosos. Estas também são a base para modelos de *machine learning* que precisam de uma adaptação dinâmica na estrutura dos dados usados para treino [71]. No entanto, pode não ser ideal para sistemas onde as consultas sobre os dados sejam complexas.

2.4.2. Interface de programação de aplicação

As *Application Programming Interface* (*API*) são interfaces de *software* que interligam diferentes sistemas e normalizam a comunicação entre as componentes duma arquitetura. Estas especificam como as solicitações devem ser usadas, assim como as informações que podem ser acedidas e os dados que podem ser enviados como resposta da chamada. Uma chamada para uma *API* é o processo pelo qual um programa ou um cliente envia uma solicitação para aceder determinados dados ou serviços disponibilizados por esta. A chamada segue um protocolo pré-definido, como o *HTTP*, e normalmente inclui as componentes *Uniform Resource Locator* (*URL*), o método, o cabeçalho, o parâmetro e o corpo. O primeiro é o endereço onde a *API* está disponível. O segundo, define a ação a ser realizada dentro de uma lista pré-definida: “GET”, “POST”, “PUT”, “DELETE”, onde o GET é usado para obter dados, enquanto o POST é usado para enviar dados. O PUT e o DELETE são utilizados para atualizar e apagar os dados, respetivamente.

Os cabeçalhos, por sua vez, contêm informação adicionais como a autenticação de um utilizador. Os parâmetros e o corpo são meramente usados para fornecer detalhes específicos dos dados, sendo que o corpo é comumente usado apenas para os métodos POST e o PUT.

As *API* têm um papel importante de não só integrar sistemas como, também, sustentar sistemas de autenticação de utilizadores. Na Figura 23 encontra-se um comparativo entre as três *API* mais usadas, nomeadamente o SOAP [72], REST [73], e GraphQL [74]. As *API* SOAP trabalham com dados no formato de *XML*, enquanto as *API* REST suportam dados no formato *JSON*, *XML*, *Hypertext Markup Language* (*HTML*) e texto. Neste aspeto, o GraphQL não suporta uma estrutura de dados específica, sendo mais flexível face à SOAP e REST. As *API* SOAP são mais seguras daí serem mais procuradas por empresas e em implementações nas linhas de negócio. Já as *API* REST e GraphQL são mais utilizadas para desenvolvimento *web*/móvel e projetos onde a complexidade dos dados é grande, respetivamente.

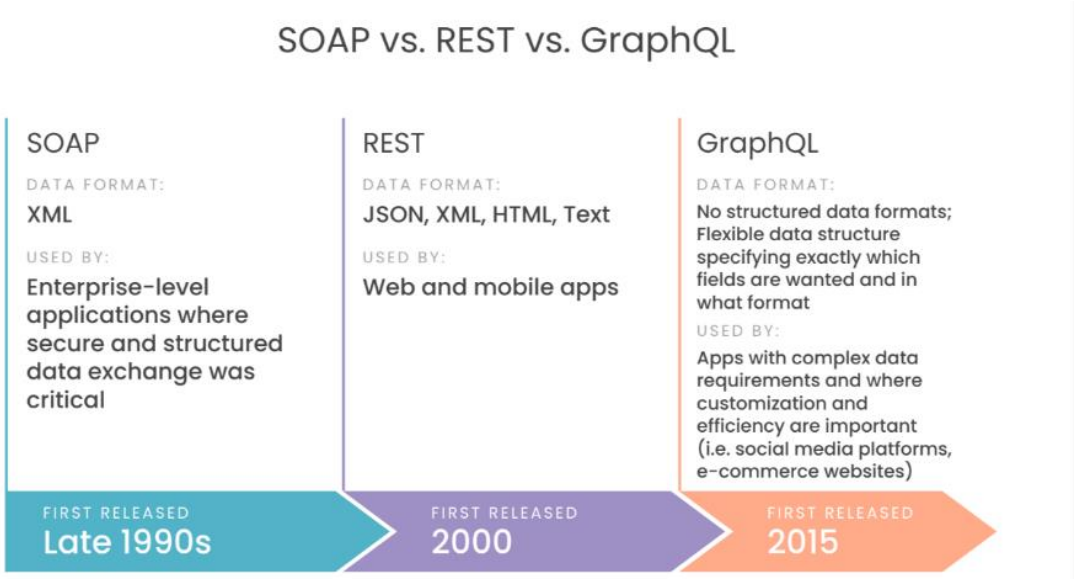


Figura 23 – Comparação entre SOAP, REST e GraphQL (Extraído de [75]).

2.4.3. Tecnologias de backend

O *backend* é a camada de *software* que trabalha no servidor de uma aplicação. Este é responsável por gerir os dados e a comunicação entre a interface do utilizador e a base de dados através de uma *API* [76]. Tipicamente o *backend* é composto por uma *API*, base de dados e servidores e existem várias tecnologias que podem ser usadas na sua construção, das quais se destacam o PHP [77], Node.js [78], Python [79], Java [80] e C# [81] para a implementação do servidor e *API*. A construção de servidores com o Node.js, por exemplo, pode ser realizada com a *framework* Express.js [82]. No caso do Python, podem ser utilizadas *frameworks* como a Django [83] e o Flask [84]. Por sua vez, o Java e o C#, possuem as *frameworks* Spring Boot [85] e o .NET Core [86], respetivamente.

2.4.4. Tecnologias de frontend

O *frontend* é a interface gráfica com o utilizador, sendo interativa e responsável por ligar as pessoas aos serviços da arquitetura. A interface gráfica pode incluir botões, gráficos e menus, que contribuem para uma experiência de utilização intuitiva e agradável [87]. Exemplos de *frameworks* usadas para a construção de *websites* são React [88], Angular [89], Vue.js [90] e Bootstrap [91]. Estas tecnologias permitem a implementação rápida de um *website*, uma vez que aplicam o conceito de componentes reutilizáveis. Entende-se como componentes pequenas partes de uma página da aplicação *web*, como o rodapé ou uma barra lateral. No que toca à construção de gráficos e componentes, existe bibliotecas para este fim como a Nivo [92], Chart.js [93] ou Material-UI [94].

2.5. Resumo

Neste capítulo foi feito a revisão de vários conceitos cruciais para a investigação da solução a desenvolver. Foi introduzido a noção do que é a *IoT* e os impactos para o quotidiano, que sendo positivos acabam por ser um alvo de investimento por parte das empresas e governos mundiais. Foi referido as várias arquiteturas *IoT*, explicando a evolução dos modelos ao longo dos anos e das diferenças entre eles, mas com isto concluiu-se que não existe um modelo de representação de um sistema *IoT* que reuna consenso.

Fez-se uma pequena passagem pelo conceito de sensores e a perceção do meio ambiente, assim como do *Cloud Computing* e do *Edge Computing*, que se descrevem por focar o processamento dos dados ao nível dos servidores ou ao nível dos sensores, respetivamente. O *Edge Computing* realçou-se face ao *Cloud Computing*, uma vez que, resolve o problema do processamento de grandes volumes de dados ao nível de um único servidor central, que ficaria sujeito a uma sobrecarga de processamento e de utilização da largura de banda disponível. Introduziu-se o conceito de *LoRa* e do LoRaWAN, iniciando a revisão com uma análise da trama *LoRa* e prosseguindo para um estudo do funcionamento de uma rede *LoRaWAN*. Com isto, foi possível entender que a rede *LoRaWAN* identifica todos os equipamentos da rede através de um ID, e que faz o processo de autenticação e negociação de chaves através do *Network server* e *Join server* no caso da autenticação *OTAA*, ou apenas através do *Network server* sem negociação no caso do método *ABP*. Fez-se uma descrição dos medidores de consumo de energia elétrica, onde foi possível identificar os dois principais tipos, nomeadamente os eletromecânicos e os eletrónicos.

Concluiu-se que os eletromecânicos estão a ficar obsoletos para alguns casos de utilização uma vez que estes não tão versáteis e não se adaptam às novas exigências do mercado, ao contrário dos medidores eletrónicos. Também se estudou a noção de medidor invasivo e não-invasivo, concluindo que um medidor não-invasivo pode ser ligado à carga sem a necessidade de abrir o circuito, uma vez que o mesmo usa um núcleo magnético e um sensor de efeito de Hall para obter os valores da corrente. Por fim, analisaram-se os tipos de bases de dados e de *API*, finalizando com a identificação das tecnologias e ferramentas para o desenvolvimento *web*. Com isto identificaram-se uma variedade de *frameworks* baseadas em Python, outras em JavaScript e até em C#. Estas *frameworks* podem acelerar o processo de construção de uma aplicação *web*, pois usam o conceito de reutilização de componentes.

3. Casos de estudo

Neste capítulo é apresentado três casos de estudo relacionados com a implementação de arquiteturas *LoRa* para a monitorização do consumo elétrico. Para cada caso é feito um resumo do trabalho desenvolvido, descrevendo a metodologia do projeto e, de seguida, faz-se uma análise critica das vantagens e desvantagens do sistema apresentado.

3.1. Medidor de energia inteligente usando uma rede LoRa em tempo real

Neste ponto é feito uma revisão de um trabalho onde foi desenvolvido um *gateway* e um sensor *LoRa*. O *gateway* desenvolvido permite a ligação de até 255 sensores e reencaminha a informação para o servidor, que usa base de dados e tecnologias de *backend* e *frondend* para apresentar os dados na forma de gráficos numa página *web*.

3.1.1. Solução desenvolvida

O objetivo primordial deste trabalho assenta em projetar uma rede *LoRaWAN* para monitorar os consumos energéticos de um edifício, garantindo segurança operacional de modo que a rede permaneça em funcionamento contínuo. Para além disso, a rede tem um custo de implementação e de manutenção reduzido [96]. Neste projeto, criaram-se dois tipos de componentes chamados de *Gateway for Residential Eletricity Metering Networks using LoRa* (GREMNL) e *Electrical Variable Measuring Devices for Households using LoRa* (EVMDHL), onde o GREMNL foi desenvolvido para atuar como um *gateway* concentrador para até 255 dispositivos EVMDHL.

O GREMNL recebe as medições proveniente dos EVMDHL, através do protocolo *LoRa*, e envia para uma núvem, através do protocolo Wi-Fi, os dados que podem ser armazenados e alimentar aplicações e serviços dos utilizadores, como o *MQTT*, o Node-RED [97], IFTTT [98] e Google Firebase. O GREMNL também pode receber comandos da nuvem, o que permite o controlo remoto da rede. A Figura 24 mostra o esquema da rede com as interações entre os vários componentes do sistema.

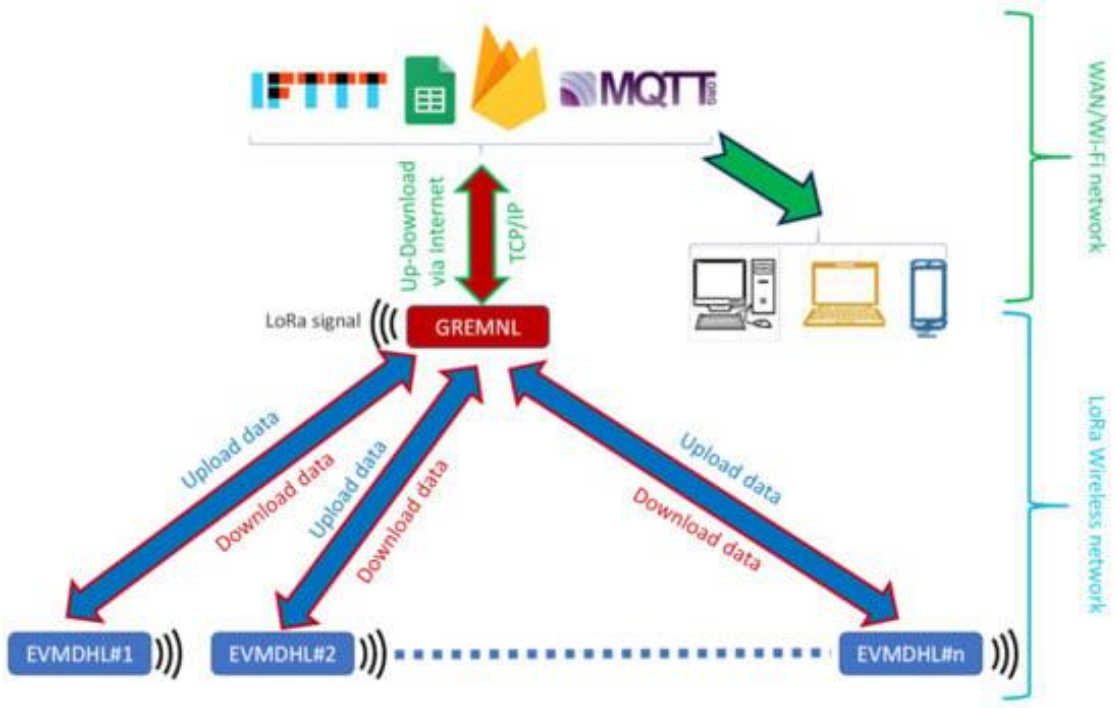


Figura 24 – Esquema da rede para o medidor de energia inteligente LoRa (Extraído de [96]).

O EVMDHL, Figura 25, realiza a transmissão dos dados através do protocolo *LoRa* e para isso foi usado o módulo Dragino LoRa Bee [99]. Este está conectado ao Arduino Nano 1 (AN1), que é responsável pela comunicação *LoRa* e de por controlar o Arduino Nano 2 (AN2) por comunicação série, seguindo a topologia *Master-Slave* [100]. O uso de dois microprocessadores deve-se ao facto da existência do *Data logger* que se liga através da interface ISP e, uma vez que, o AN1 possui apenas uma porta ISP e já se encontra em utilização, o uso de um só Arduino Nano tornou-se inviável. O *Data logger* é usado para de armazenar os dados, uma vez que a comunicação sem fios não é totalmente fiável existindo, apesar de pequenas, algumas perdas de pacotes. As medições são efetuadas ao nível do AN2, onde se encontra o módulo de medição PZEM-004t que está conectado à carga elétrica.

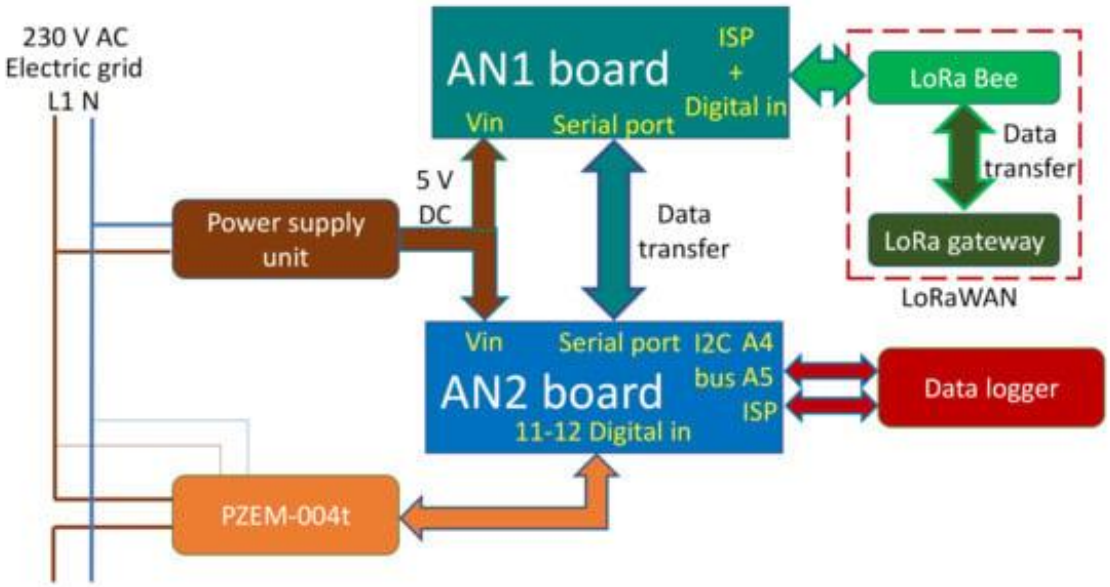


Figura 25 – Diagrama de blocos do dispositivo EVMDHL (Extraído de [96]).

A Figura 26 mostra o fluxograma do modo de funcionamento do EVMDHL. O dispositivo começa por iniciar a rotina dos vários módulos e a mensagem para medição é enviada do AN1 para o AN2, e por fim para o PZEM. Caso o PZEM não responda o processo de envio da mensagem é repetido, mas se o PZEM responder com uma medição, o AN2 vai guardar os dados no *Data logger* e enviar os mesmos para o AN1. Por sua vez, o AN1 ao receber estes dados faz a transmissão e aguarda por uma confirmação do GREMNL. O AN1 vai esperar um tempo determinado para receber a confirmação, caso o tempo de espera seja ultrapassado ele reenvia a mensagem, repetindo este ciclo até obter uma resposta. Se, na janela de tempo de espera, o AN1 receber uma mensagem que não seja a de confirmação este vai reenviar novamente a mensagem. No caso de a confirmação chegar, este fica atento a novas mensagens para alterar os parâmetros da rede ou a dimensão do *payload*, e volta ao ponto de iniciar o pedido de leitura ao módulo PZEM. A leitura dos dados elétricos é feita num intervalo de 0.5s, mas não foi especificado se isto se refere ao intervalo de leitura do PZEM para realizar uma medição, ou se é referente ao intervalo para a medição ser iniciada por ordem do Arduino ao medidor.

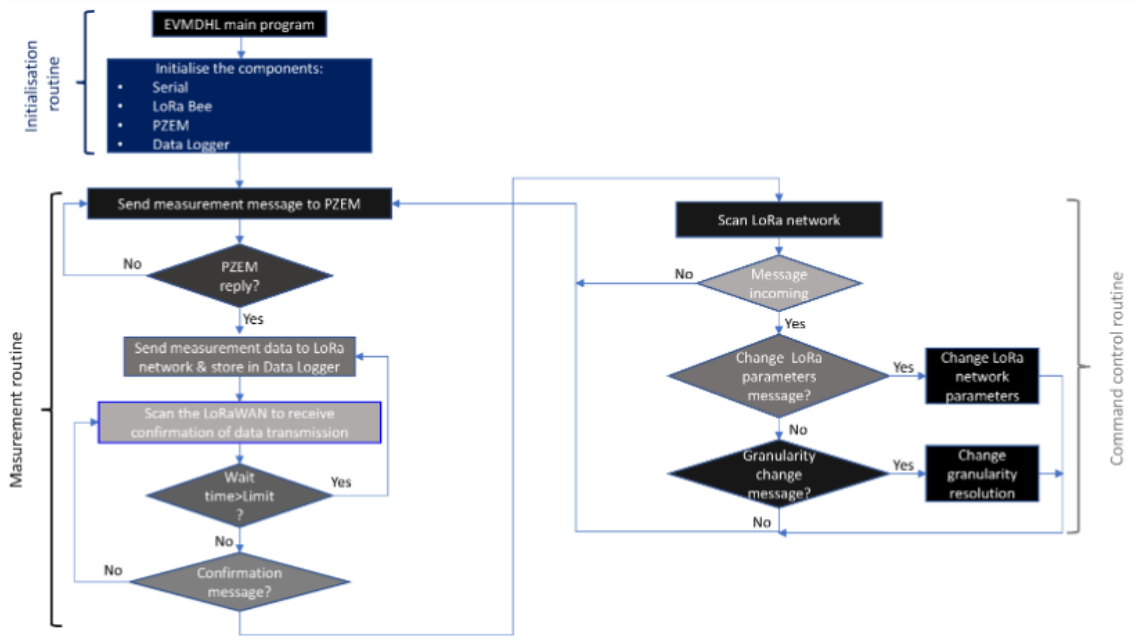


Figura 26 - Fluxograma do EVMDHL (Extraído de [96]).

O GREMNL, Figura 27, assim tal como o EVMDHL, utiliza dois microcontroladores. O Arduino Nano (AN) é usado para gerir a comunicação *LoRa* através de um módulo LoRa Bee e o Wd1M é usado para gerir a comunicação Wi-Fi. Quando os dados são enviados do EVMDHL para o GREMNL, o Arduino Nano recebe os dados e reencaminha-os para o Wd1M através da porta série para serem posteriormente transmitidos por Wi-Fi. Esta comunicação é bidirecional, portanto, comandos também podem ser recebidos pelo Wd1M através da nuvem para serem executados ao nível do AN.

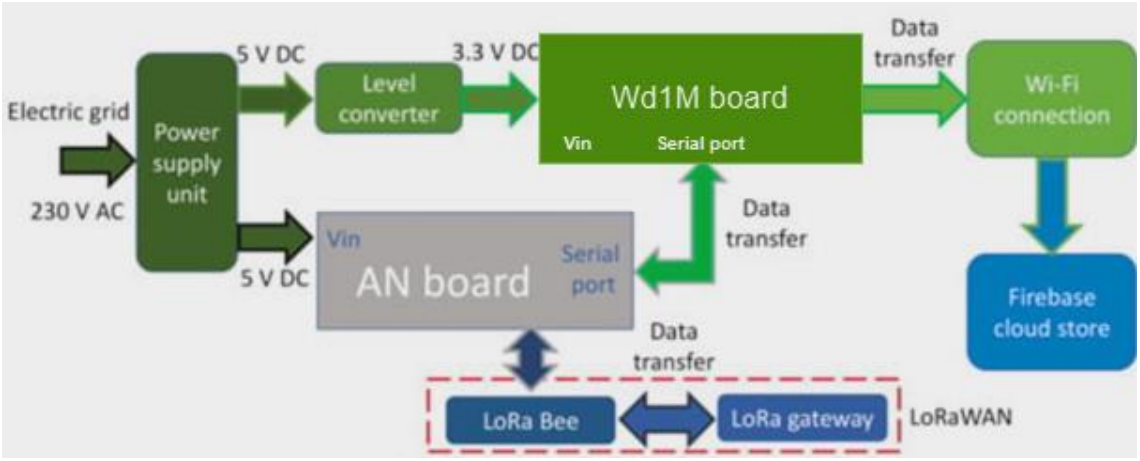


Figura 27 - Diagrama de blocos do GREMNL (Modificado a partir de [96]).

O modo de funcionamento do *gateway* GREMNL fica ilustrado pela Figura 28. A inicialização começa por configurar a porta série e o módulo LoRa Bee, abrindo posteriormente uma janela para mensagens provenientes da nuvem. Se alguma mensagem for recebida, com o propósito de alterar parâmetros de rede ou para alterar a resolução da medição, esta é enviada para os EVMDHL e uma confirmação é enviada para a nuvem. O GREMNL abre uma outra janela para comunicações *LoRa*, onde aguarda para receber dados de medição. Ao receber uma medição, esta é rotulada com a identificação do EVMDHL para ser armazenada na base de dados do serviço da Firebase e é enviada uma confirmação para esse mesmo EVMDHL.

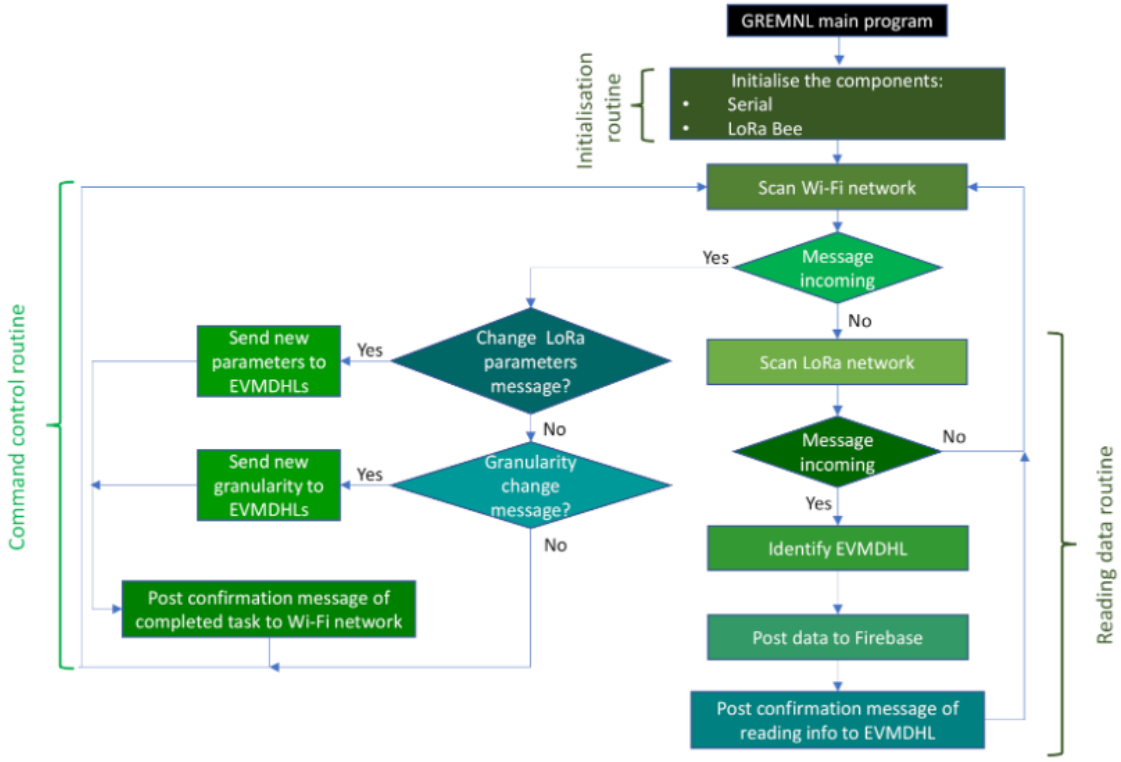


Figura 28 – Fluxograma do GREMNL (Extraído de [96]).

3.1.2. Análise

Relativamente ao trabalho desenvolvido, podem apontar-se algumas considerações sobre a robustez do sistema. Os dispositivos desenhados permitem que os parâmetros da rede sejam alterados remotamente, assim como o tamanho das mensagens. O *gateway* GREMNL permite que até 255 medidores EVMDHL possam ser conectados à rede o que é número interessante para projetos que precisam de vários dispositivos finais ligados na rede.

Um ponto positivo tanto do GREMNL como do EVMDHL é que o envio de dados entre estes e a nuvem é confirmada por mensagens, que na sua ausência após o envio de comandos faz com que os equipamentos saibam que é necessário repetir o envio dos comandos, permitindo contornar problemas como falhas na rede e perdas de pacotes.

A construção de um *gateway LoRa* personalizado também pode ser visto como um ponto interessante neste projeto, uma vez que, é possível converter pacotes *LoRa* para Wi-Fi a um custo mais reduzido do que adquirir uma solução já pronta do mercado. Uma nota que pode ser feita sobre o EVMDHL se o foco do projeto for a redução do custo de implementação reduzido, é a utilização de dois Arduino Nano. No caso do EVMDHL, o uso de dois microprocessadores foi justificado pela necessidade de tanto o módulo Dragino como o *Data logger* precisarem da porta ISP. Contundo, sendo que a perda de pacotes foi classificada como bastante reduzida pode-se descartar a necessidade de adquirir um módulo para armazenar os dados fisicamente e utilizar apenas uma unidade do Arduino, com a vantagem de reduzir o preço de cada medidor.

3.2. Monitorização, em tempo real, de medidores de energia com armazenamento em nuvem

Neste subcapítulo é apresentado um projeto onde foi desenvolvido uma rede *LoRa* com aplicação *web*. O sensor, composto por um Raspberry Pi, está ligado a um medidor de energia através de um optocoplador. O sensor faz a contagem de impulsos elétricos, que se traduzem num nível de energia, permitindo que se faça o cálculo de energia consumida.

3.2.1. Solução desenvolvida

Neste projeto desenvolveu-se um sistema dividido em três componentes principais, a saber, o *hardware*, a rede de comunicação *LoRa* e o *software*. Foi usado um sensor, cujo modelo não é especificado, que faz a leitura da potência em kHz com uma constante de medição de 3200 impulsos por kWh. O optoacoplador PC817 permite traduzir a medição do medidor de energia para impulsos de contagem, onde um impulso equivale a um nível de energia medido e que vai ser enviado para o Raspberry Pi. Por sua vez, o Raspberry Pi é o módulo central do nó sensor que vai enviar os dados para o *gateway LoRa*.

O *gateway LoRa* em questão é o intermediário desta rede e vai enviar os dados dos nós sensores para a nuvem através de comunicação Wi-Fi. O módulo usado é o SX1278 IC, que permite trabalhar em frequências de 433MHz. A nuvem disponibiliza o serviço da Firebase para armazenar os dados, enquanto a *dashboard* utiliza a tecnologia Angular JS e o *backend* o Express.js, que é uma ferramenta do Node.js, para implementar APIs. Na Figura 29 ilustra-se graficamente a rede desenvolvida neste trabalho.

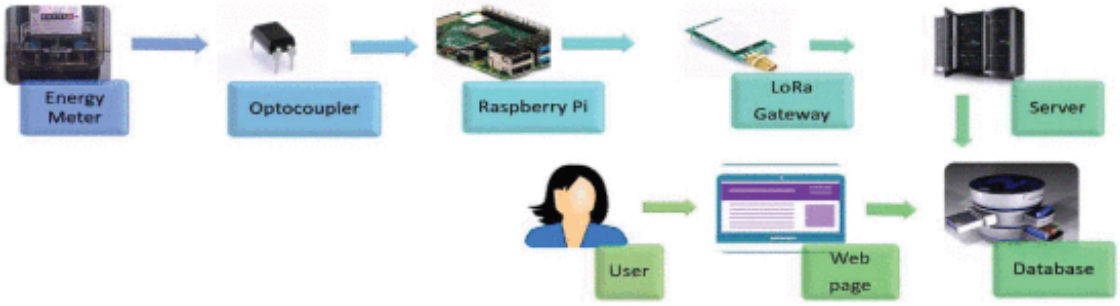


Figura 29 – Esquema da rede para a monitoração em tempo real (Extraída de [101]).

O *website* desenvolvido permite visualizar vários detalhes do consumo de vários utilizadores. Um utilizador só terá acesso aos seus dados, através de um sistema de autenticação que controla os acessos à informação relativamente às faturas de eletricidade e o consumo energético, com recurso a representações gráficas como se pode observar na

Figura 30, que representa a evolução temporal do consumo energético em contagens ao longo dos meses. Também é permitido ao utilizador alterar a escala de tempo e ver os consumos ao nível das semanas e até dias de forma a ter uma visão mais detalhada do uso energia elétrica.

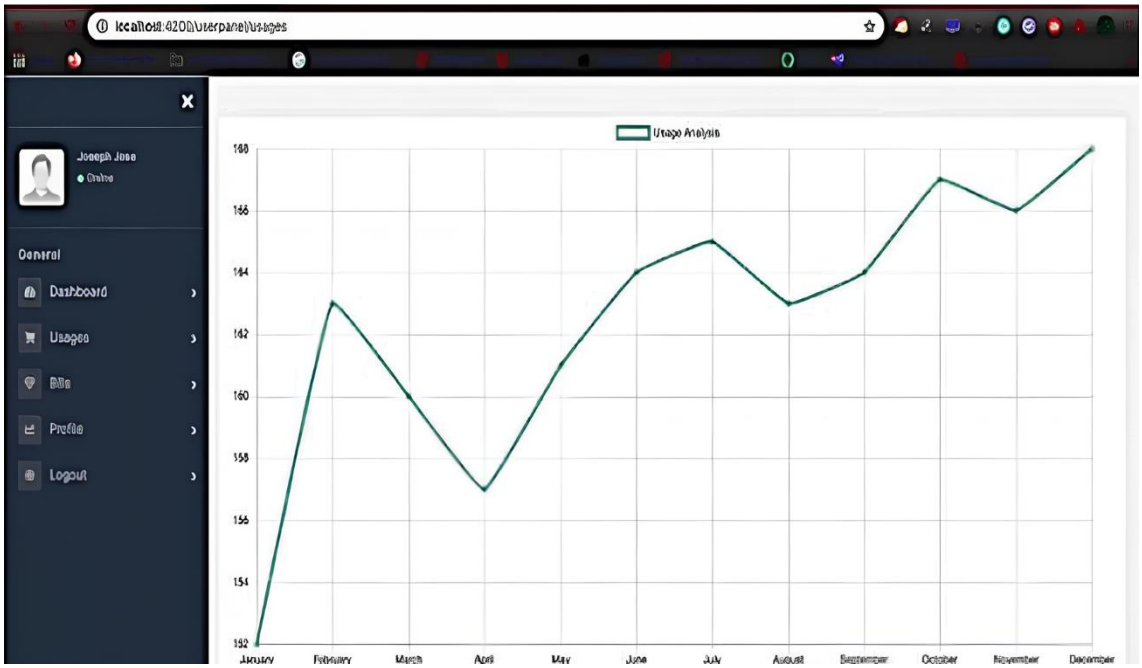


Figura 30 – Aplicação desenvolvida para consumos elétricos (Extraída de [101]).

3.2.2. Análise

A rede desenvolvida apresenta algumas limitações quanto ao *hardware*, nomeadamente a informação que é extraída do medidor de energia, que refere apenas as unidades de kWh consumidas sem revelar mais detalhes como tensão, corrente, distorção harmónica e fator de potência. Outro ponto é o uso do Raspberry Pi, que apesar de não ser especificado o modelo utilizado, este é um computador compacto, mas que encarece o custo do nó sensor. Não sendo apresentados motivos convincentes para a escolha do Raspberry Pi, o uso de um microcontrolador seria mais benéfico para este fim.

A construção de uma *dasboard* personalizada é um ponto positivo, ficando ao critério da empresa ou desenvolvedores decidir como a informação vai ser disponibilizada e quais a funcionalidades do *frontend*, mantendo o nível de privacidade e domínio sobre a plataforma. Seguindo esta linha de pensamento, de construir soluções customizadas, poderia ter-se optado por não usar a tecnologia de terceiros como a Firebase, de forma a tornar este projeto independente de serviços externos e manter o serviço da nuvem ou não nos próximos anos, para além de que o uso é pago após exceder um dado limite de uso.

3.3. Sistema inteligente de monitorização e consumo de energia baseado em IoT

Neste subcapítulo é apresentado um trabalho onde foi construído uma rede com plataforma para cálculo de faturamento da eletricidade. O sensor está ligado a um contador inteligente e recebe os dados através da ligação RS845. Os dados são transmitidos para o servidor através do *gateway LoRa* LG308.

3.3.1. Solução desenvolvida

O objetivo desta solução é criar uma rede com plataforma para cálculo de facturamento de eletricidade, onde as faturas são envias por SMS para os clientes. Na Figura 31 encontra-se a rede *LoRaWAN* implementada. O medidor de energia utilizado foi o Elmeasure LG+3399, que faz a leitura da energia consumida em kWh (1 unidade) e de outros parâmetros da rede como a corrente, a tensão e frequência. Este medidor possui uma porta de comunicação RS845, que envia comandos para um nó sensor *LoRa*, que utiliza um módulo Dragino para estabelecer comunicação com a rede *LoRa*. O *gateway LoRa* utilizado é o Dragino LG308 [102]. O *gateway LoRa* recebe pacotes do nó sensor e envia para o servidor de rede *The Things Stack* da *The Things Network* (*TTN*) através do Wi-Fi, assim como recebe informações do TTS para enviar para o sensor.

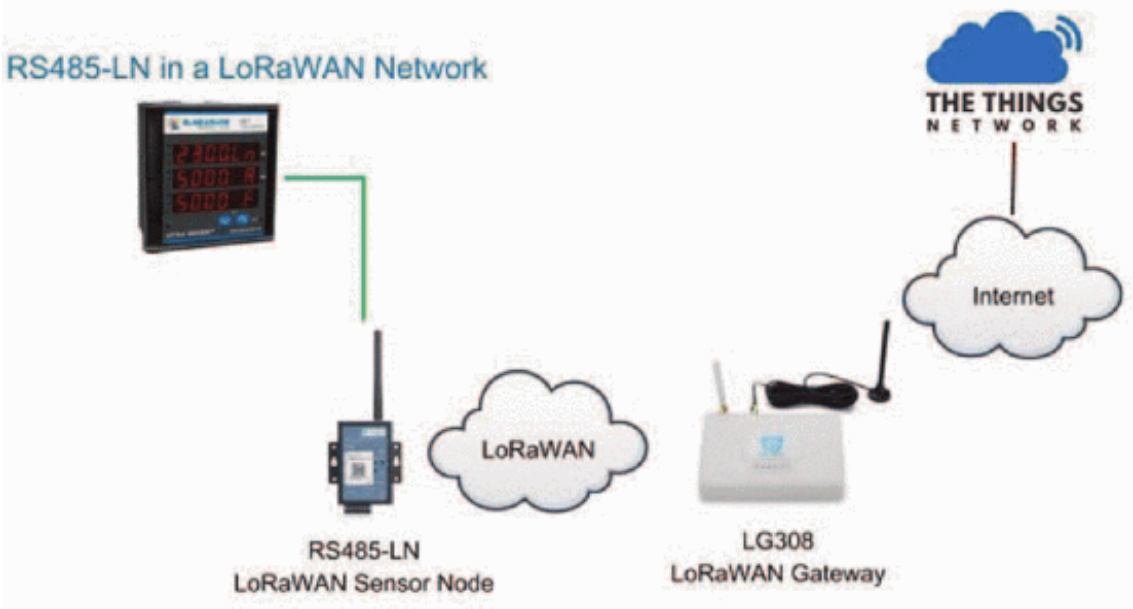
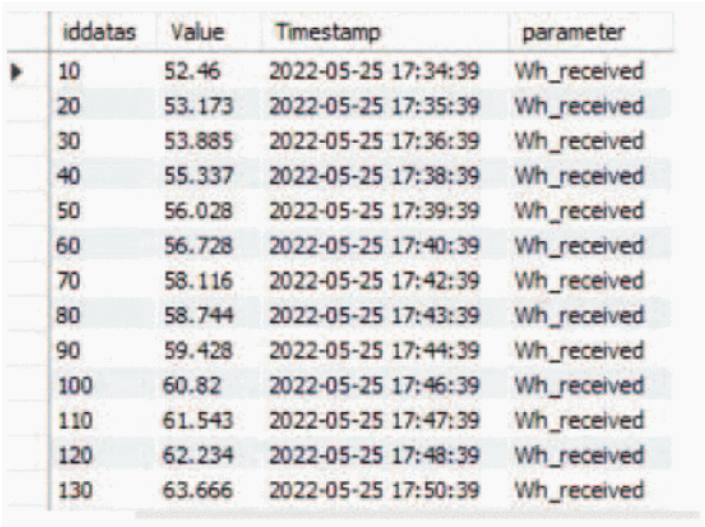


Figura 31 - Esquema do sistema de monitorização elétrica (Extraída de [103]).

Neste projeto configuraram a integração da *TTN* com o *MQTT* de forma a enviar dados para a sua aplicação Node-Red, que vai subscrever os tópicos do *broker*. Como se pode ver na Figura 32, os dados são armazenados numa base de dados MySQL, com indicação do parâmetro que está a ser registado, a data e hora do registo e o valor. Através do campo com a data do registo é possível concluir que o intervalo de amostragem é de um minuto. Para realizar o cálculo da fatura a pagar, os autores optaram por extrair o valor mais alto no último dia do mês e multiplicar pela tarifa aplicada por kWh.



iddatas	Value	Timestamp	parameter
10	52.46	2022-05-25 17:34:39	Wh_received
20	53.173	2022-05-25 17:35:39	Wh_received
30	53.885	2022-05-25 17:36:39	Wh_received
40	55.337	2022-05-25 17:38:39	Wh_received
50	56.028	2022-05-25 17:39:39	Wh_received
60	56.728	2022-05-25 17:40:39	Wh_received
70	58.116	2022-05-25 17:42:39	Wh_received
80	58.744	2022-05-25 17:43:39	Wh_received
90	59.428	2022-05-25 17:44:39	Wh_received
100	60.82	2022-05-25 17:46:39	Wh_received
110	61.543	2022-05-25 17:47:39	Wh_received
120	62.234	2022-05-25 17:48:39	Wh_received
130	63.666	2022-05-25 17:50:39	Wh_received

Figura 32 – Tabela MySQL com os dados das medições (Extraída de [103]).

Para mostrar os dados através de gráficos aos utilizadores, os autores usaram a tecnologia Grafana. A *dashboard* criada permite que se façam análises mais detalhadas aos dados e alterar a representação dos dados para acompanhar as preferências de cada pessoa. Existem vários tipos de representações dos dados no *website* desenvolvido, nomeadamente medidores, gráficos de barras, gráficos circulares, exibição de tabelas e gráficos de tempo. Na Figura 33, é possível ver uma *mock* do ecrã *frontend* construído com o grafana, onde é possível ver um gráfico de evolução temporal de várias métricas da rede elétrica como a fase, a corrente, o consumo energético, e a tensão.

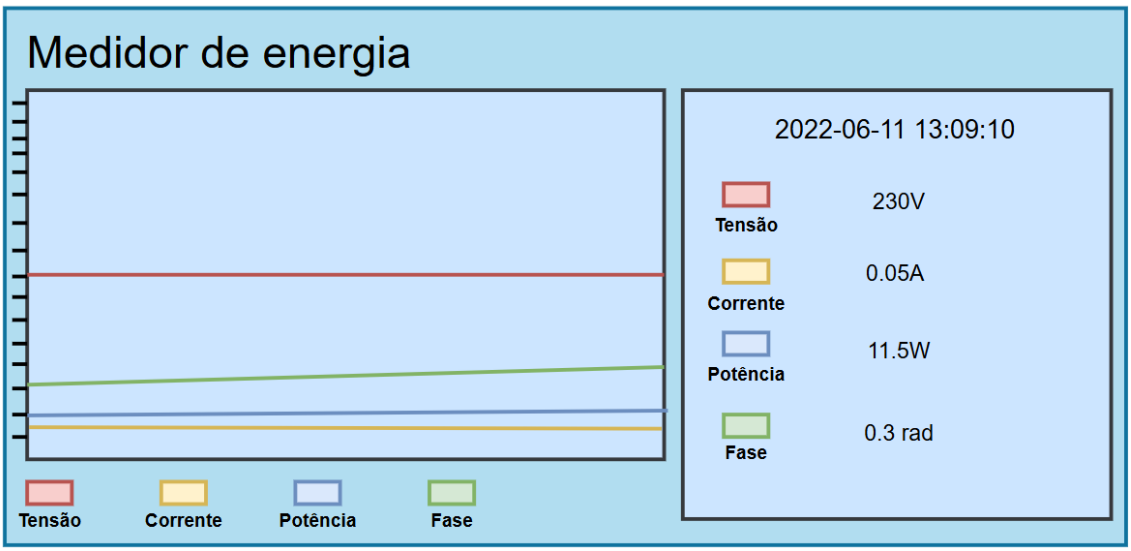


Figura 33 – Mock do gráfico desenvolvido em Grafana (Construída a partir de [103]).

3.3.2. Análise

A solução desenvolvida serve de exemplo de como é fácil integrar a *TTN* e uma aplicação final através do *MQTT*. A *TTN* permite a integração do servidor de rede *LoRaWAN* com qualquer aplicação final, oferecendo várias opções para o efeito. Contundo, o projeto fica dependente da plataforma, ou seja, se a *TTN* decidir tirar a sua solução do mercado a estrutura da rede terá de ser substituída para acomodar um novo servidor de rede *LoRaWAN*. Outro ponto negativo desta rede é o facto da *TTN* ter um uso gratuito limitado a um número de equipamentos conectados. Em relação ao *hardware* utilizado, uma vez que foi adquirido um medidor de energia programável e com porta de comunicação para conectar o nó sensor não é necessário usar um microcontrolador para o circuito do nó sensor.

3.4. Resumo

A análise destes casos de estudo permitiu obter conhecimentos de outros projetos sobre o mesmo tema desde trabalho de investigação, e fazer um levantamento de vantagens e desvantagens das arquiteturas. O primeiro caso de estudo envolveu a construção de um *gateway* e sensor *LoRa* juntamente com uma a implementação de uma *dashboard*. Concluiu-se que um dos dois microprocessadores podia ser descartado, uma vez que a chance de perda de dados é desprezável, o que torna o *Data logger* descartável.

O segundo caso de estudo apresenta uma rede *LoRaWAN* em que o sensor é ligado a um medidor por meio de um optoacoplador para fazer a contagem da energia consumida, juntamente com uma interface gráfica para os utilizadores terem acesso às contas de luz. Com isto percebeu-se que o sistema apenas monitoriza a energia consumida, excluindo outras métricas que também são importantes para o contexto da saúde da rede elétrica. Outro problema apontado foi o uso do Raspberry Pi, o que encarece bastante o nó sensor. Por último, analisou-se um sistema que utiliza um contador inteligente para fazer a medição de algumas métricas como a tensão, corrente e potência. Esta informação é enviada para o nó sensor através da ligação RS485 e posteriormente transmitida para um *gateway LoRa* do mercado. A informação é apresentada em gráficos, através da *framework* Grafana. Com este trabalho concluiu-se que o uso de um microprocessador é descartável, uma vez que, o contador inteligente é programável e poderia ser ligado diretamente ao módulo de transmissão *LoRa*, e que o uso da plataforma *TTN* torna o projeto dependente de tecnologia de terceiros.

4. Implementação do sistema

Neste capítulo é apresentado toda a metodologia de implementação da solução para monitorização dos consumos elétricos. É feito um levantamento de requerimentos do sistema e o respetivo desenho da arquitetura a implementar, assim como a introdução do *gateway* adquirido pelo IPS. É realizado uma análise de mercado sobre os medidores de energia e módulos *LoRa* e é apresentado a escolha do equipamento para desenhar a arquitetura do nó sensor e integrar o respetivo *hardware*. Também é apresentado o desenho da arquitetura da aplicação *web*, assim como a *API* para integrar o *backend* e o *frontend*. É explicado o sistema de autenticação desenvolvido, e a aplicação React em termos de navegação e chamadas efetuadas para a *API*. Por fim, é feito a descrição dos passos necessários para a integração da aplicação *web* e do sensor com o servidor *LoRaWAN* da *TTN*, onde se começa por configurar a aplicação *TTN* e fazer o registo do sensor, partindo daí para a configuração do *broker MQTT* e do desenvolvimento do conector TTN-Servidor.

4.1. Desenho da arquitetura

Neste ponto é mostrado o levantamento dos requisitos do sistema e das funcionalidades necessárias para garantir os objetivos da Instituição. Seguidamente, é apresentado a arquitetura do sistema e o tipo de comunicação que interliga cada componente da rede. Por último, é feito uma referência ao *gateway LoRa* adquirido pelo IPS, aquisição esta que permitiu o desenvolvimento de novos projetos *LoRa* incluindo o apresentado neste documento.

4.1.1. Requerimentos do Sistema

Umas das principais características a ter em conta no desenvolvimento do sistema pretendido é o preço de implementação e o custo de manutenção. O objetivo de construir uma arquitetura *IoT* para medição de consumos elétricos no IPS é, em primeiro lugar, permitir obter conhecimento sobre a utilização da energia elétrica a fim de reduzir os gastos energéticos por forma a tornar o campus mais sustentável.

Contudo, a implementação com vários sensores faz com que o preço total do projeto escale facilmente, daí a preocupação em desenhar uma solução que entregue o necessário para a Instituição alcançar os seus objetivos, mas com um baixo custo de investimento.

Outro ponto pertinente é a segurança e privacidade dos dados, uma vez que referem aos consumos de uma instituição. Garantir que o acesso aos dados seja controlado e gerido de acordo com permissões adequadas e a existência de um sistema de autenticação é crucial para que a informação seja visualizada e usada pelas pessoas indicadas para tal. Numa ótica de pensar em desenvolvimentos para o futuro, a segurança deste sistema pode também justificar a transparência e viabilidade dos dados, para implementar modelos para gerar relatórios e identificar anomalias nos consumos elétricos.

O sensor deve ser capaz de extrair informação suficiente da carga a monitorizar e preferencialmente de forma não invasiva, tornando possível uma análise transparente e detalhada com várias métricas. O número de métricas a extrair deve ser o necessário para que o utilizador consiga fazer a sua análise de forma objetiva, ou seja, também não deve exceder para além do necessário, caso contrário pode dificultar a leitura dos dados para aferir resultados. Para o contexto deste projeto, definiu-se como métricas necessárias a tensão (V), corrente (A), potência aparente (VA), potência ativa (W), potência reativa (VAR), energia consumida (kWh), frequência (Hz) e fator de potência (rad).

Olhando numa perspetiva de mais alto nível, a construção da rede e a sua integração devem ser efetuadas entre o servidor de rede *LoRa* e a aplicação final, que devem ser conectados de modo a preparar os dados para serem apresentados na camada da aplicação.

4.1.2. Arquitetura da solução

A arquitetura da rede a implementar encontra-se na Figura 34.O sensor *LoRa* está ligado a uma carga, para extrair as métricas dos consumos elétricos. Estes dados são transmitidos através do protocolo *LoRa* para um *gateway*, que transforma a trama *LoRa* para uma trama TCP/IP. O *gateway* também pode receber dados do servidor de rede através do protocolo TCP/IP, convertendo a trama para *LoRa* enviando-a para o nó sensor. Em relação ao servidor de rede, escolheu-se o *The Things Network* (*TTN*) por oferecer várias opções para integrar sistemas. Em alternativa, o ChirpStack, na ótica deste projeto, seria a escolha mais apropriada por ser uma tecnologia *Open-Source*, mas esta solução foi identificada numa fase avançada do desenvolvimento e não houve tempo para aplicar as alterações na arquitetura para utilizar o ChirpStack no lugar da *TTN*.

A integração fez-se, então, com o uso do protocolo *MQTT* da *TTN*, onde um tópico é publicado com os dados recorrentemente. O tópico é único e, portanto, todas as métricas são enviadas por aí. O servidor da aplicação final vai ter um conector que subscreve esse tópico e realiza uma extração das métricas e de outras informações como o ID do sensor, faz a formatação dos dados para além dos cálculos indiretos de métricas e trata do seu armazenamento numa base de dados. Este conector é, também, responsável por identificar valores dos dados que passem de um limite estabelecido pela aplicação final, para enviar um email para todos os utilizadores registados na plataforma com alertas sobre a ocorrência. A aplicação final vai recolher os dados conforme a utilização da plataforma pelos utilizadores, através do protocolo Wi-Fi, e disponibilizar os mesmos com recurso a representações gráficas.

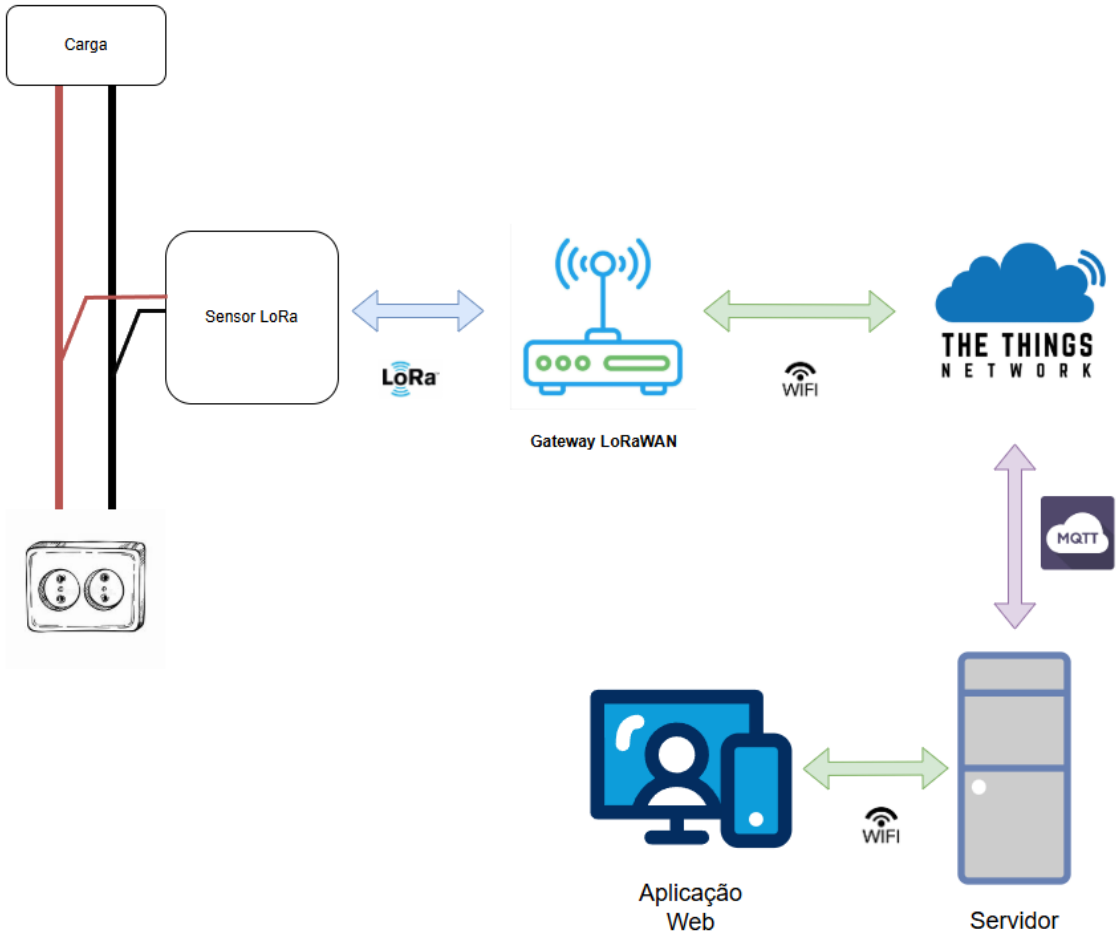


Figura 34 - Arquitetura da rede de monitorização de consumos elétricos LoRa.

4.1.3. Gateway LoRa

Visando a implementação de projetos inovadores *LoRa*, o IPS adquiriu o *gateway* LoRa Kerlink Wirnet iStation [104]. Este *gateway*, Figura 35, possui um total de onze canais RX, onde oito são canais para comunicação de 125kHz, um canal para 250kHz e outro para 500kHz, e ainda um canal para modulação FSK. Este tem um único canal para TX. O Kerlink permite a transformação de tramas *LoRa* para tramas TCP/IP, tendo compatibilidade com a comunicação 3G/4G, Ethernet através de cabo RJ45 e WLAN. O *gateway* tem a tecnologia *Power over Ethernet* (PoE), o que torna a sua alimentação prática ao ponto de poder usar o cabo RJ45 para ser alimentada. A localização do *gateway* encontra-se no Anexo 1.

Figura 35 – *Gateway* LoRa Kerlink Wirnet iStation (Extraído de [104]).

4.2. Desenvolvimento do sensor

Neste ponto é apresentado a análise de mercado relacionada com os medidores de energia, assim como com os módulos de comunicação *LoRa*. É apresentado o diagrama de blocos e o desenho do esquema elétrico do protótipo do sensor desenvolvido e, também, do fluxograma do programa que rege o funcionamento do sensor. Posteriormente, é explicado o funcionamento do medidor PZEM ao nível do *hardware* e da integração deste na arquitetura. Do mesmo modo, também é explicado a metodologia para configurar o módulo LoRa-E5 através dos comandos AT.

4.2.1. Análise de medidores de energia

Com uma grande variedade de medidores de energia elétrica disponíveis no mercado, fez-se uma pequena pesquisa de modo a escolher a solução mais indicada face aos critérios do projeto em questão. Na Figura 36 encontra-se o módulo medidor Seeed 101991032 [105], que é capaz de medir apenas a tensão eficaz (RMS V) de uma carga. O medidor foi construído para medir tensões entre os 120V a 240V AC o que, num ponto de vista de implementação, pode ser interessante para cenários em que apenas se pretende medir instantes onde a tensão se desvia dos 230V nominais da rede elétrica Portuguesa. O módulo tem uma interface IC MCP6002 [106], podendo ser alimentado com uma fonte de 3.3V a 5V. Os sinais dos dados são analógicos, o que permite regular o *ADC* para aumentar ou diminuir a resolução e controlar a frequência de amostragem. Um dos pontos negativos deste módulo é incapacidade de extrair outras grandezas. Estando limitado apenas à tensão eficaz, não serve viabiliza a determinação de outras métricas ainda que calculadas indiretamente.

Figura 36 - Medidor Seeed 101991032 (Extraído de [105]).

Um outro sensor avaliado é o SCT-013-020 [107], que é um medidor de corrente instantânea que opera na gama dos 0A a 20A AC. Trata-se de um sensor não invasivo, Figura 37, na medida em que se liga por uma pinça à carga para permitir determinar a corrente sem a necessidade de interromper o circuito. A saída varia de 0 a 1V de acordo com o valor da corrente à entrada, e no caso deste medidor, a saída é enviada por audio-jack. Um dos pontos negativos deste sensor é novamente a falta de dados de outras grandezas que permitam calcular as métricas desejadas, uma vez que apenas extrai informação da corrente. Mesmo assumindo uma tensão constante de 230V para efetuar o cálculo das restantes métricas, a desvantagem desta abordagem é que a maioria das métricas são obtidas indiretamente, pressupondo um valor fixo para a tensão.

Figura 37 – Medidor de Corrente SCT-013-020 (Extraído de [107]).

O módulo PZEM-004T-100A é um medidor de energia também não invasivo, capaz de extrair um número considerável de métricas nomeadamente a tensão eficaz, corrente eficaz, potência ativa, fator de potência, energia consumida e frequência. Este mede tensões entre os 80V a 260V AC e correntes entre 0A e 100A AC com erros de medição de 0.5% [108]. O PZEM fornece dados sobre a potência ativa com um erro de 0.5%, desde que esteja dentro de um intervalo de 0kW a 23kW. A potência ativa é apresentada com virgula flutuante se a mesma for menor que 1kW, caso contrário será aproximada para um número inteiro. A energia consumida é uma outra métrica interessante, na medida em que pode ajudar a perceber ou obter uma estimativa do consumo total num dado período.

A energia consumida é obtida por este medidor desde que entre 0kWh e 10000kWh, com um erro de leitura de 0.5%. O fator de potência e a frequência, por sua vez, são calculados com um erro de leitura de 1% e 0.5% respetivamente. O medidor em questão tem uma interface TTL para comunicação série com recurso ao protocolo RS-485, o que facilita a integração com microprocessadores como o usado na plataforma Arduino. Na Figura 38 encontra-se o medidor de energia.

Figura 38 – Medidor PZEM-004T-100A (Extraído de [108]).

Fazendo uma comparação entre os medidores analisados, o PZEM parece ser o mais indicado para a implementação deste projeto, uma vez que permite extrair muitas características da rede elétrica com taxas de erro relativamente baixas. Este também é não invasivo, pois utiliza uma pinça para obter a informação da corrente evitando que o sensor tenha de ser ligado em série no circuito elétrico. O SCT-013-020 é um medidor simples e que apenas extrai informação da corrente elétrica enviando o sinal por ligação de áudio-jack, e isso implicaria a necessidade de desenvolver um circuito adicional para a descodificação do sinal. Por outro lado, o Seeed 101991032 tem uma interface de comunicação mais razoável para o envio de dados, mas está restringido à leitura da tensão.

4.2.2. Análise de módulos LoRa

Analisando as opções para módulos *LoRa* disponíveis no mercado, encontraram-se algumas opções acessíveis em termos de preço. Começando pelo LoRa-E5 mini [109], que é suportado pelo microprocessador STM32WLE5JC [110], é compatível com larguras de bandas *LoRa* globais, como 868 MHz na Europa e 915 MHz na América. Este destaca-se pelo baixo consumo de energia, o que o torna ideal para dispositivos alimentados por bateria. Este módulo, ilustrado na Figura 39, oferece interfaces como UART, I2C, SPI, *ADC* e GPIO, permitindo uma integração fácil com sensores. Este é pré-carregado com a *firmware* AT, permitindo que possa ser configurado com comandos AT através das portas série, embora também suporte desenvolvimento através da plataforma de desenvolvimento de software STM32Cube. O módulo é de classe A por padrão, mas pode ser programado para classe B ou C.

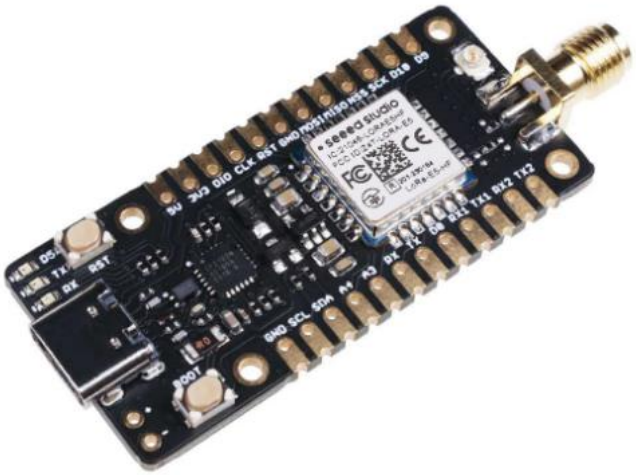


Figura 39 – Módulo LoRa-E5 mini (Extraído de [109]).

À semelhança do LoRa-E5 mini, existe a versão Grove LoRa-E5 [111], Figura 40, que também foi projetado com o microprocessador STM32WLE5JC. Este também é compatível com as larguras de banda 868MHz e 915MHz, possuindo as interfaces UART, I2C, SPI, *ADC* e GPIO. O que o diferencia é o modulo de conectividade Grove, que possibilita a ligação direta com sensores e outros periféricos compatíveis, o preço e a diferença de tamanhos com o LoRa-E5 mini sendo o mais compacto, tal como o nome indica. Este também é de classe A por padrão, mas pode ser programado como classe B ou C.

Figura 40 – Módulo Grove LoRa-E5 (Extraído de [111]).

O RA-02 LoRa SX1278 [112], equipado com o circuito SX1278 da Semtech [113], foi projetado para trabalhar na frequência de 433 MHz. Este módulo, Figura 41, suporta taxas de transmissão até 300 kbps, o que permite equilibrar alcance e velocidade de comunicação conforme as necessidades do projeto. A sua interface SPI facilita a integração com microcontroladores e plataformas populares, como Arduino, Raspberry Pi e ESP32. Além disso, a sua potência de transmissão pode ser configurada até 10 dBm (10 mW). Apesar da frequência 433MHz estar regulada na europa e ser usada em alguns projetos, a frequência 868MHz é mais aceite e também mais utilizada devido a vários fatores, como a menor interferência de ruído e maior potência de transmissão de 14dBm (25mW). O RA-02 LoRa tal como os outros módulos também de classe A e pode ser programado para classe B ou C.

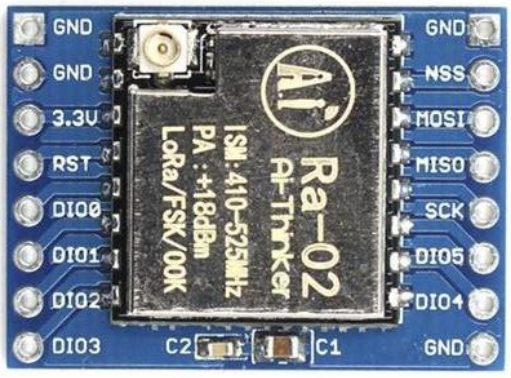


Figura 41 – Módulo RA-02 LoRa SX1278 (Extraído de [112]).

Analisando as características dos módulos *LoRa* acima identificados, pode concluir-se que todos os módulos podem funcionar como classe A, B ou C e que o LoRa-E5, tal como a sua versão mini, são os mais indicados para a implementação do projeto por trabalharem na banda de 868MHz.

Esta banda permite que os dados sejam transmitidos mais rapidamente face à banda 433MHz. Dentro das opções para o LoRa-E5, é uma questão de escolha em relação ao preço e às pequenas diferenças entre os módulos de transmissão disponíveis no mercado. Uma vez que o tamanho do sensor não é um fator fundamental, escolheu-se o Grove LoRa-E5 por ser mais barato face à versão mini.

4.2.3. Arquitetura do sensor

Com base na análise efetuada no último subcapítulo, desenhou-se o protótipo do sensor inteligente cujo diagrama de blocos se encontra na Figura 42. O processamento central ocorre ao nível do Arduino Uno, que vai controlar o módulo de transmissão Grove LoRa-E5 e o medidor PZEM-004t-100A. Usou-se a biblioteca *SoftwareSerial* [114] do Arduino para simular uma segunda porta série, que vai estabelecer comunicação com o módulo *LoRa*. Para comunicar com o medidor PZEM usou-se a biblioteca PZEM004Tv30 [115] para simplificar a integração, e que também recorre do *SoftwareSerial* implicitamente para simular outra porta série. Deste modo a porta série nativa do Arduino Uno fica disponível para outros fins, como ligar a um computador ou outro periférico para ser programado ou para ser analisado no monitor série.

O PZEM também foi ligado a uma carga de 230V AC, de forma a fazer a leitura da informação e enviar para o Arduino, sendo posteriormente reencaminhada para o LoRa-E5 onde é transmitida para o *gateway LoRa*. Para o efeito, o medidor precisou de quatro ligações das quais se destacam as duas que ligam à fase e ao neutro da rede elétrica, que foram ligadas em paralelo à carga de forma a obter-se o valor da tensão. Para obter a corrente, foi necessário ligar a pinça ao circuito da carga (Anexo 2). A alimentação do circuito é feita através de uma pilha de 9V mas pode ser obtida a partir de uma fonte de alimentação ligada à própria rede que se pretende medir.

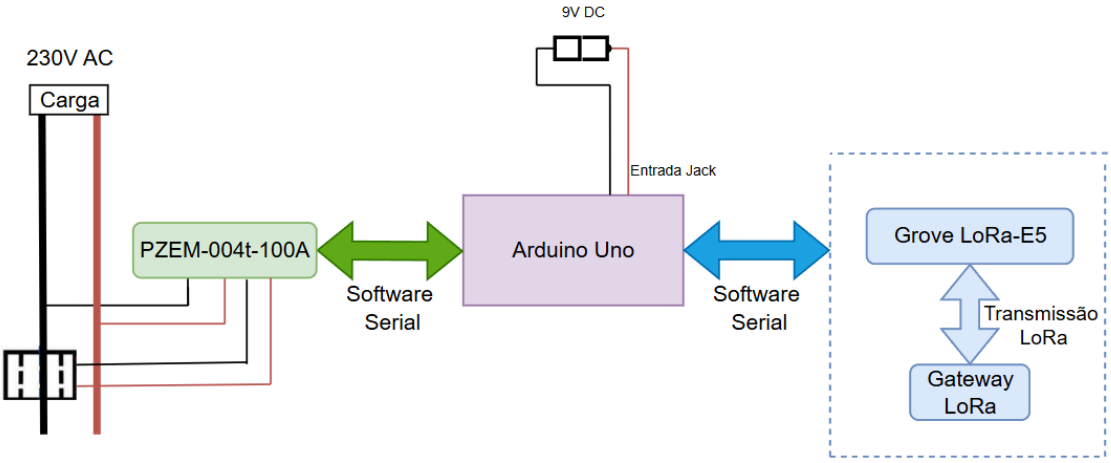


Figura 42 – Diagrama de blocos do protótipo do medidor *LoRa*.

O esquema elétrico do circuito fica ilustrado na Figura 43, com as devidas ligações entre os vários módulos. O módulo LoRa-E5 é alimentado pelo pino de 3.3V do Arduino e a comunicação é efetuada ao nível dos pinos D10 e D11 do Arduino, que foram programados para se comportarem como os sinais RX e TX da porta série, respetivamente. O módulo PZEM, por sua vez, é ligado à alimentação numa fonte de 5V do Arduino e utiliza os pinos D5 e D6 para os sinais RX e TX, respetivamente. A ligação à carga elétrica é feita através do PZEM, onde os pinos P_PZEM e N_PZEM são ligados em paralelo à fase e neutro da rede elétrica. A pinça, ou o núcleo magnético, de acordo com a especificação do fabricante, é presa ao neutro da carga sem que seja necessário qualquer corte no fio ou montagem em série com a carga. Deste modo foi possível desenhar um sensor de baixo custo, mas que, ainda assim, é capaz de fornecer a informação necessária ao nível dos requisitos deste projeto.

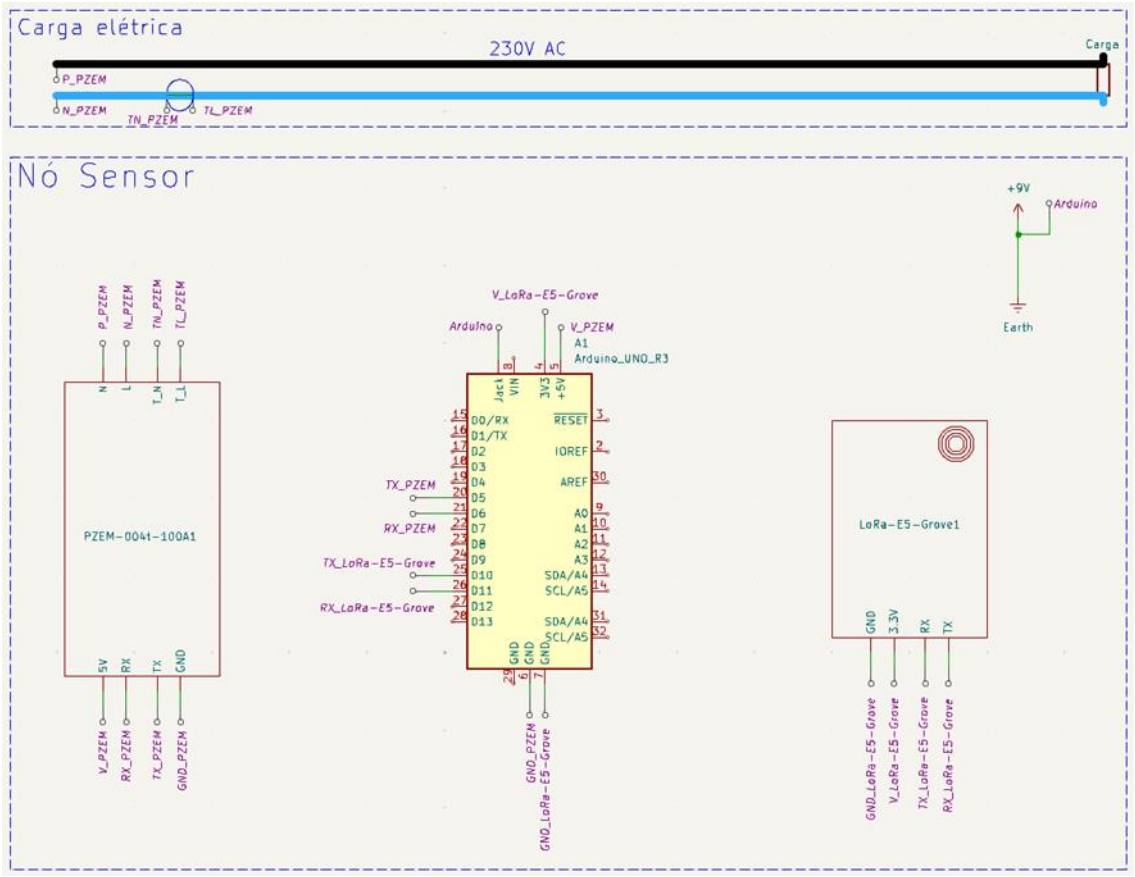


Figura 43 – Esquema elétrico do protótipo do nó sensor.

A rotina começa por iniciar os módulos PZEM e *LoRa*, assim como configurar as portas série do sensor e da interface LoRa. A configuração do módulo LoRa-E5 é feita através de um conjunto de comandos AT definidos no *datasheet* disponibilizado pelo fabricante [116].

O programa prossegue definindo o atributo AppKey que é a chave da aplicação criada no servidor de rede *LoRa TTN* para a qual o sensor deve comunicar. Também se realizaram algumas configurações em relação ao modo de autenticação com o servidor, optando-se pelo método *OTAA* na medida em que a autenticação é feita de forma automática e isso simplifica o processo de adicionar um dispositivo, e a taxa de transmissão, optando-se por um *SF9* com largura de banda de 125kHz, respetivamente.

Uma vez terminada a configuração do LoRa-E5, o Arduino faz um pedido de JOIN para a *TTN*, ficando a aguardar trinta segundos para receber uma resposta deste. Se durante essa janela de tempo a resposta de confirmação não chegar, o Arduino vai repetir o processo de configuração do módulo *LoRa* e forçar novamente o pedido de JOIN.

Se dentro da janela de espera a mensagem de confirmação chegar o Arduino inicia um ciclo de rotina para realizar a leitura da carga elétrica. Durante o processo de medição, várias métricas são obtidas nomeadamente a tensão eficaz, a corrente eficaz, a potência ativa, a energia consumida, a frequência e o fator de potência. A energia consumida é calculada pelo contador do PZEM, que faz a contagem da energia de forma continua desde que é alimentado e não depende do período no qual o Arduino faz a leitura dos dados.

Os dados são posteriormente transmitidos pelo módulo *LoRa*, chegando, assim, ao servidor *TTN*. Para reduzir o consumo energético, o nó sensor foi configurado de modo que, no fim de uma transmissão de dados, o mesmo entre em modo de baixo consumo, desligando o *Analog-to-Digital-Converter* (*ADC*) e o *Brown-Out Detection* (*BOD*) por quatro segundos, retomando depois a atividade e realizando novo processo de medição e transmissão. O fluxograma da lógica programada no microprocessador Arduino encontra-se na Figura 44. O código do programa encontra-se no Anexo 3.

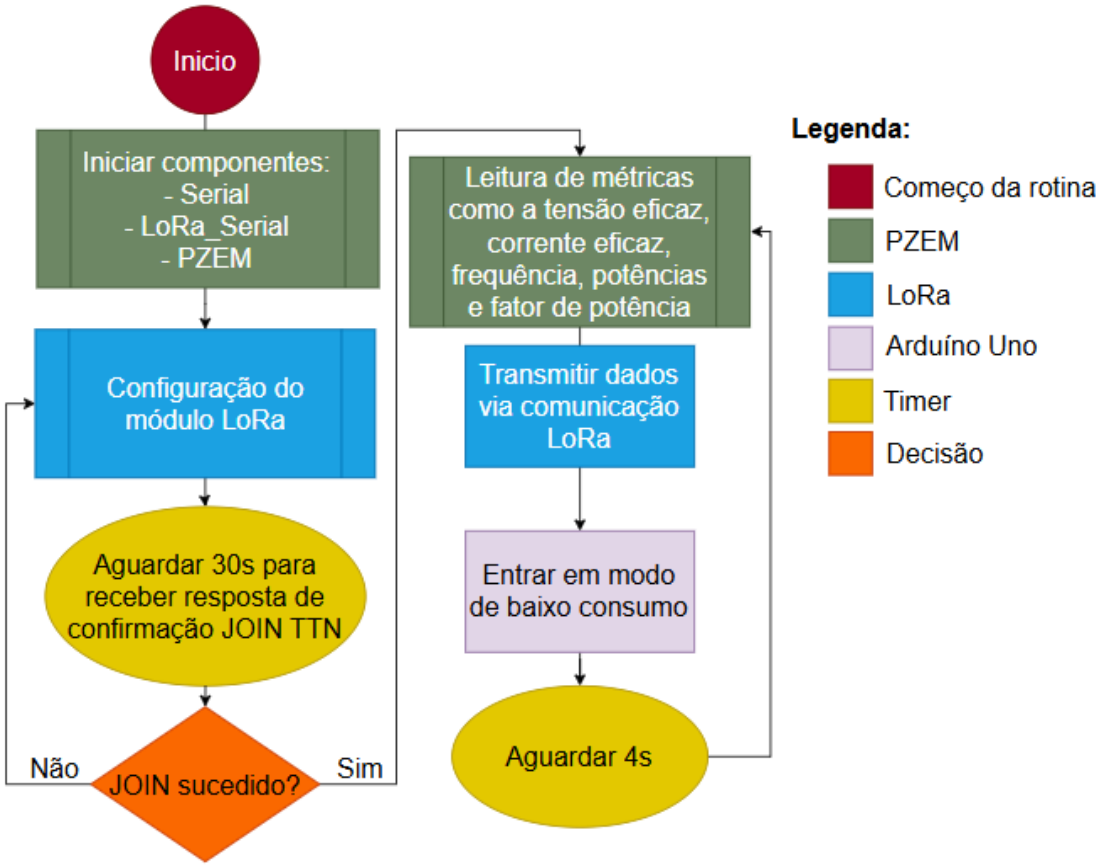


Figura 44 – Fluxograma da rotina programa no protótipo do nó sensor.

4.2.4. Integração do módulo PZEM-004t-100A

Para integrar o PZEM com o Arduino utilizou-se a biblioteca PZEM004Tv30, que simplifica bastante o processo de comunicação entre os dois equipamentos. Contundo, é importante entender o funcionamento desta lógica e o que faz para controlar o medidor. Na Figura 45 encontra-se o diagrama de blocos do PZEM, onde é possível identificar cinco blocos distintos, nomeadamente a fonte de alimentação AC, o bloco de sinal, o sistema de medição, o bloco de isolamento galvânico com recurso a um optoacoplador, e a interface TTL. As entradas N, L e CT (transformador de corrente com o uso do núcleo magnético) alimentam todo este circuito a partir do bloco de alimentação AC. O bloco de sinal é o responsável por acondicionar os sinais de entrada provenientes da rede elétrica, assim como da tensão do sinal CT, que é uma representação do valor da corrente na rede. No bloco Measurement System realiza-se o processamento dos sinais para determinar várias métricas dentro do sistema de medição como a tensão eficaz, a corrente eficaz, a potência ativa, a potência reativa, a energia ativa, o fator de potência e a frequência. O optoacoplador oferece um isolamento galvânico entre a interface TTL e o sistema de medição.

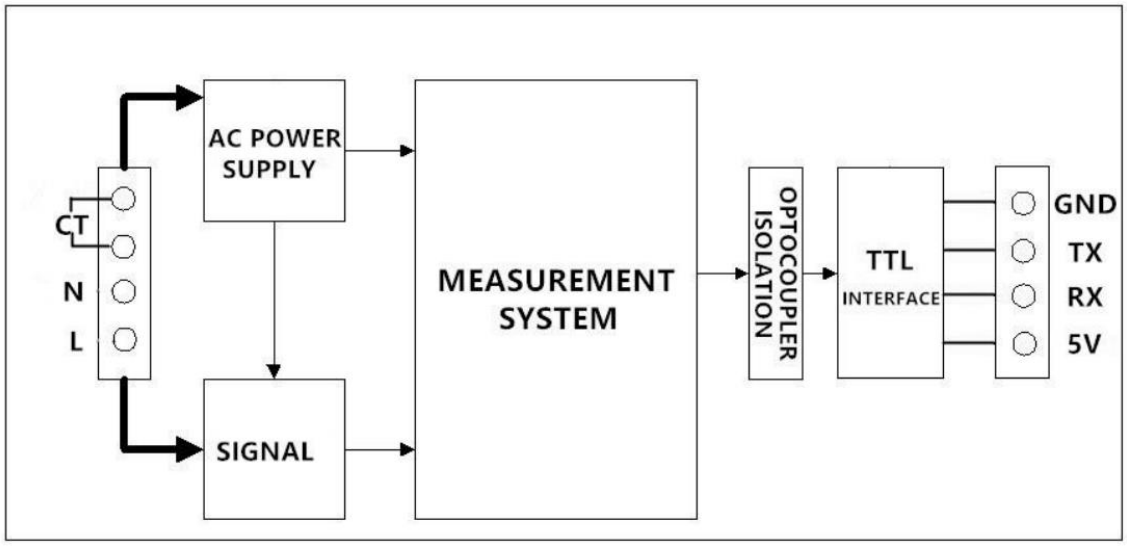


Figura 45 – Diagrama de blocos do medidor PZEM-004t-100A (Extraída de [117]).

As métricas da corrente elétrica são, na sua maioria, calculadas com base em amostras discretas, que são recolhidas pelo PZEM num dado intervalo de tempo. Tanto os números de amostras com o intervalo de tempo de leitura não estão especificados pela fabricante, e também não são programáveis.

A tensão eficaz (V RMS) é calculada pelo sistema de medição do PZEM através da raiz quadrada do produto entre o inverso do número de amostras da tensão (N) e o somatório discreto de cada amostra da tensão v (V) ao quadrado:

$$V_{RMS} = \sqrt{\frac{1}{N}\sum_{j=1}^{N} v^2} \tag{4.1}$$

Através da mesma equação o PZEM obtém a corrente eficaz (I RMS) através da raiz quadrada do produto entre o inverso do número de amostras da corrente (N) e o somatório discreto de cada amostra da corrente i (A) ao quadrado:

$$I_{RMS} = \sqrt{\frac{1}{N}\sum_{j=1}^{N} i^2} \tag{4.2}$$

Para calcular a potência ativa, o medidor recorre do produto da tensão eficaz V_{RMS} (V) da corrente eficaz I_{RMS} (A) e do cosseno do fator de potência cos ϕ (rad):

$$P_{ativa} = V_{RMS} \times I_{RMS} \times \cos\phi \tag{4.3}$$

A diferença de fase (ϕ), por sua vez, é calculado pela diferença de fase entre a forma de onda da tensão e a forma de onda da corrente, que corresponde ao rácio do atraso t_{atraso} (segundos) entre os cruzamentos em zero da tensão e da corrente pelo período t (segundos) das formas de onda multiplicado por 2π:

$$\phi = 2\pi\frac{t_{atraso}}{t}\ rad \tag{4.4}$$

O fator de potência (PF) é obtido através do cosseno de ϕ, pois as formas de ondas da tensão e corrente são próximas de uma onda sinusoidal perfeita e, porque, a impedância do circuito é constante:

$$PF = \cos\phi \tag{4.5}$$

A frequência f do sinal elétrico é calculada pelo método *zero-crossing*, onde o PZEM faz a contagem do tempo até que ocorra uma chegada ao zero por parte da forma da onda elétrica, $t_{zero\ crossing}$ (segundos), onde também ocorre a inversão da polaridade do sinal AC. A partir desta contagem de tempo, que equivale a meio período da onda, o medidor calcula a frequência utilizando a expressão:

$$f = \frac{1}{2t_{zero\ crossing}} \tag{4.6}$$

As restantes métricas, como a potência reativa e potência aparente, não são calculadas pelo PZEM, mas a informação já obtidas permite calculá-las indiretamente. De forma a manter o *payload* da comunicação compacto e evitar sobrecarregar o Arduino com processamento extra, o cálculo destas métricas é realizado ao nível da aplicação final. A potência aparente S (VA) é calculada através da multiplicação da tensão eficaz V_{RMS} (V) com a corrente eficaz I_{RMS} (A):

$$S = V_{RMS} \times I_{RMS} \tag{4.7}$$

A partir do (4.6) é possível calcular a potência reativa Q (VAR), multiplicando a potência aparente pelo seno do desfasamento ϕ (rad), obtido em (4.4):

$$Q = V_{RMS} \times I_{RMS} \times \sin\phi \tag{4.8}$$

A informação de cada métrica é armazenada num endereço específico da memória do medidor PZEM, ao qual o Arduino vai aceder sempre para extrair os dados. A mostra como o endereçamento de cada métrica está organizado dentro da memória, assim como a quantidade de bits para a sua representação. Uma palavra é representada por 16 bits [117].

O valor RMS da tensão encontra-se no endereço de memória 0x0000 e o bit menos significativo (LSB) corresponde a 0.1V, ou seja, a tensão é apresentada por um valor inteiro que representa décimas de Volt. A corrente, tem dois endereços alocados para a sua representação. O endereço 0x0001 é usado para representar os 16 bits menos significativos, enquanto o endereço 0x0002 é usado para os 16 bits mais significativos totalizando 32 bits para a representação da corrente. Esses 32 bits representam a corrente num valor inteiro em mili Ampére. Do mesmo modo, a potência ativa é representada por 16 bits LSB, endereço 0x0003, e por 16 bits MSB também totalizando 32 bits.

Neste caso a precisão é menor, pois o bit menos significativo representa 0.1W, ou seja, o valor inteiro representa décimas de Watt. Seguindo o mesmo exemplo, a energia é armazenada nos endereços 0x0004 e 0x0005, 16 bits LSB e 16 bits MSB respetivamente, com precisão de 1Wh. A frequência e o fator de potência, com 16 bits cada, nos endereços 0x0007 e 0x0008 respetivamente, são representados com uma precisão de 0.1Hz e 0.01 radianos.

Por último, existe um endereço 0x0009 para o estado do alarme que vem implementado no sensor, mas que neste projeto foi descartado porque o mesmo só permite ser ativado através de um limite configurável para a potência ativa. Para além disso, o alarme é apenas lógico sendo necessário adicionar uma forma de sinalizar o evento com, por exemplo, um buzzer ou um LED, e alterar a programação do Arduino para verificar o valor armazenado no endereço 0x0009. A aplicação desenvolvida já contempla um sistema de alarme através do envio de emails, que é mais prático e vai ao encontro da especificação deste projeto.

Tabela 3 – Tabela de endereços de memória do PZEM-004t-100A (Extraído de [117]).

Register address	Description	Resolution
0x0000	Voltage value	1LSB correspond to 0.1V
0x0001	Current value low 16 bits	1LSB correspond to 0.001A
0x0002	Current value high 16 bits	
0x0003	Power value low 16 bits	1LSB correspond to 0.1W
0x0004	Power value high 16 bits	
0x0005	Energy value low 16 bits	1LSB correspond to 1Wh
0x0006	Energy value high 16 bits	
0x0007	Frequency value	1LSB correspond to 0.1Hz
0x0008	Power factor value	1LSB correspond to 0.01
0x0009	Alarm status	0xFFFF is alarm, 0x0000is not alarm

Ao realizar-se a leitura de cada métrica, o Arduino vai criar uma *string* de dados. Esta estrutura permite que os dados sejam apresentados de uma forma legível e que, de um ponto de vista da programação, é fácil de trabalhar. A Figura 46 mostra a *string* construída para o *payload* de dados e enviar.

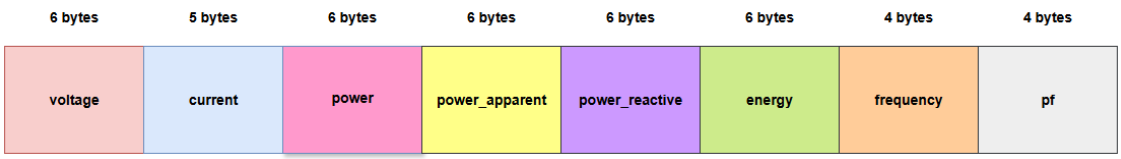


Figura 46 - Estrutura do *payload* enviado pelo sensor *LoRa*.

4.2.5. Integração do módulo LoRa-E5 Grove

Como se referiu anteriormente, todo o controlo do módulo LoRa-E5 se faz com base em comandos AT. Para configurar o módulo em questão e posteriormente enviar dados, é necessário realizar os passos ilustrados na Figura 47. A ordem dos comandos até ao passo 4, inclusive, é irrelevante e estes podem ser executados numa ordem arbitrária, com exceção do comando do passo 1 que serve para formatar as configurações já presentes, que tem de ser o primeiro. Sempre que um comando é enviado para o módulo *LoRa*, este responde com um "OK" caso o comando tenha sido executado com sucesso. Na situação oposta este responde com uma mensagem de erro.

Seguindo os passos pela ordem indicada, começa-se por fazer um *reset* das configurações executando, de seguida, o comando para definir o modo de autenticação para o método *OTAA*. Realizando algumas configurações extra, como a definição da chave da aplicação TTN e da taxa de dados, o nó sensor fica pronto para juntar-se ao servidor de rede *LoRa* através do comando "AT+JOIN" do passo 5. O comando usado para enviar dados pode ser visto no passo 6, através do comando "AT+MSG=payload", que transmite os dados e não aguarda por uma mensagem de confirmação por parte do servidor *TTN*.

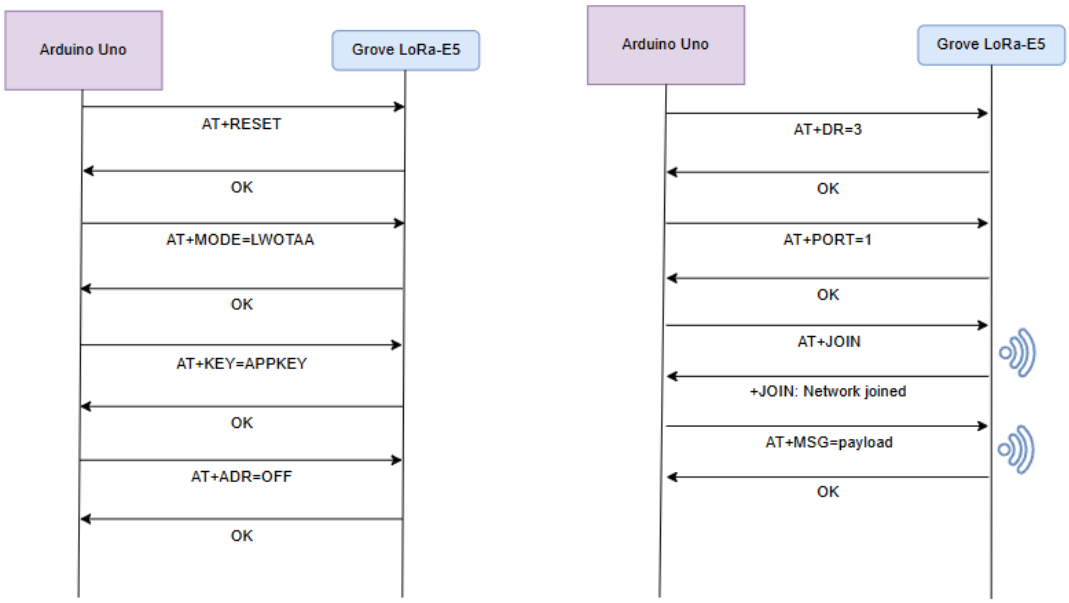


Figura 47 – Passos usados para configurar o módulo *LoRa* e enviar dados para a rede.

4.3. Desenvolvimento da aplicação final

Neste subcapítulo é apresentado o desenho da aplicação final React. É feito uma abordagem à *API* desenhada para integrar a aplicação React com o servidor *backend*, assim como do sistema de autenticação JWT que garante o controlo de acesso aos dados de natureza sensível, através de credenciais. Por fim, é detalhado a forma como a aplicação React comunica com a *API* e esta, por sua vez, com a base de dados, durante a navegação do utilizador assim como na sequência das ações executadas por este.

4.3.1. Desenho da aplicação final

Por forma a criar uma estrutura que suporte uma aplicação robusta e capaz de cumprir as exigências deste projeto, realizou-se o desenho da arquitetura *backend* e *frontend*, Figura 48. O *backend* funciona com base num servidor local Ubuntu, que vai executar várias tarefas e hospedar tecnologias como o NodeJS, onde vai ser implementada uma *API* com o uso de Javascript, e construir uma aplicação *dashboard* em React. A *API* é o ponto de conexão entre a *dashboard*, que está do lado do utilizador, e o servidor local Ubuntu. A *API* deve recolher várias informações dos utilizadores assim como das medições efetuadas, que são armazenadas numa base de dados não-relacional MongoDB.

A integração deste servidor local com a *TTN* realiza-se através de um conector Python, que vai subscrever o tópico da aplicação criada dentro do servidor de rede *LoRaWAN* e armazenar os dados localmente após a devida verificação e processamento da informação. Este conector também verifica a ocorrência de anomalias durante o armazenamento dos dados, enviando um email de alerta com os detalhes para cada um dos utilizadores registados na aplicação.

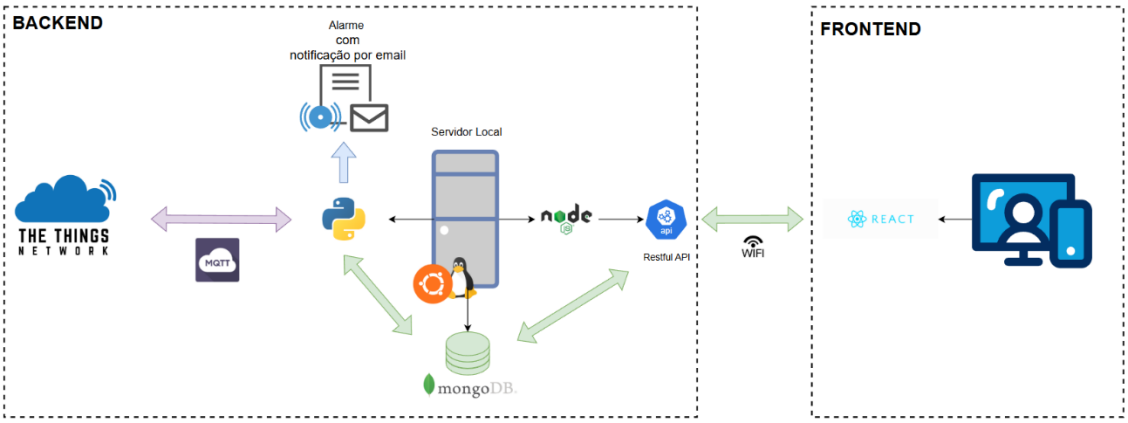


Figura 48 – Arquitetura da aplicação final desenvolvida.

A aplicação *web* é construída em React, que é uma *framework* bastante conceituada do Javascript. O objetivo da aplicação é dar uma nova perspetiva dos dados ao utilizador, de forma que os vários detalhes de uma ou mais cargas elétricas possam ser visualizadas através de gráficos intuitivos e que reduzem o tempo e complexidade da análise da informação produzida pelos sensores.

É necessário tecer algumas considerações em relação à escolha do MongoDB como motor da base de dados. O MongoDB é uma base de dados não-relacional, que armazena os dados em formato de documento *JSON*. Esta estrutura permite que os dados possam ser armazenados sem a necessidade de um cuidado em manter as relações entre entidades, assim como a lógica de programação para manusear as *queries* é menos complexa pela ausência de relações entre entidades. Outro ponto é o facto dos *payloads* da *TTN* já se encontrarem neste formato, o que não só facilita a integração como também evita processamento extra dos dados para a sua restruturação.

A escolha de uma base de dados não-relacional neste contexto também se deve ao facto de não existir uma atividade operacional de negócio para justificar relações entre entidades. Num ponto de vista técnico, uma base de dados não-relacional é mais adequada para este trabalho de investigação, uma vez que o ponto central do projeto é construir uma estrutura LoRa para um grande fluxo de dados elétricos e apresentá-los de forma simples e objetiva. Também se estima a possibilidade destes dados virem a ser usados para explorar soluções que envolvam inteligência artificial. Neste aspeto, o armazenamento não-relacional permite que grandes quantidades de dados sejam extraídas mais rapidamente face aos modelos relacionais, e já num formato adequado para alimentar modelos inteligentes e para aplicações que têm compatibilidade com o *JSON*, como é o caso de aplicações Javascript e, por sua vez, React.

4.3.2. PowerAPI

De modo a integrar a aplicação React com a restante arquitetura *backend*, como a base de dados, utilizou-se a biblioteca express.js para implementar uma *API Restful*, que foi nomeada de PowerAPI. A PowerAPI permite um conjunto de chamadas que a aplicação *web* usa para obter, atualizar, inserir ou apagar dados na base de dados. A chamada "POST /login" serve essencialmente para criar *tokens* de autenticação, que vão ser usados ao longo da navegação para verificar se o utilizador tem permissão para aceder aos recursos, ou se precisa de fazer *login* novamente. Esta chamada faz-se com o envio do *username* e da *password* do utilizador que está a entrar na aplicação. A chamada "POST /authenticate" é usada para validar, durante a navegação, se os *tokens* do utilizador estão expirados. Esta chamada requer os atributos *accessToken* e o *userID* no corpo.

A chamada “POST /logout” tem o propósito de encerrar a sessão do utilizador e elimina os *tokens* para a navegação e, para isso, requer o atributo userID no corpo. Para obter os dados dos sensores, faz-se a chamada “GET /data” que tem como argumento o atributo device_id, para identificar a qual dos sensores se deve extrair a informação armazenada. Para obter informação sobre os sensores que estão ligados na rede, a aplicação faz um pedido para o “GET /sensors” especificando os sensores requeridos pelo seu estado no argumento. Este também tem a chamada “PUT /sensors” para alterar o estado de um sensor, indicando o estado do sensor e o device_id no corpo. A chamada “/users” tem três métodos, nomeadamente o “GET”, “DELETE” e “POST”, em que o primeiro é usado para obter todos os utilizadores, o segundo é usado para apagar um utilizador indicando o userID no corpo, e o terceiro para adicionar um novo utilizador com as informações no corpo do pedido. Por fim, tem-se a chamada “GET /alarms” e “PUT /alarms”. Na primeira, a PowerAPI retorna todos os alarmes configurados e a segunda atualiza o limite de um alarme indicando no corpo a métrica e o valor de *threshold*. A Tabela 4 contém todos as chamadas programadas, assim como o método Restful, o argumento que é passado e o conteúdo do corpo.

Tabela 4 - Tabela de chamadas da PowerAPI.

Chamada	Método	Argumento	Corpo
/login	POST	-	username/ password
/authenticate	POST	-	accessToken/ userID
/logout	POST	-	userID
/data	GET	device_id	-
/sensors	GET	status	
	PUT	-	device_id/ status
/users	GET	-	-
	DELETE	-	userID
	POST	-	userDetails
/alarms	GET	-	-
	PUT	-	metric/ threshold

4.3.3. Sistema de autenticação JWT

A segurança do sistema é um fator muito importante e, de modo a garantir o controlo no acesso aos dados dentro da aplicação, foi desenhado um sistema de autenticação JWT. Este sistema utiliza *tokens* para garantir que o utilizador está autenticado no sistema, e é necessário que o mesmo faça um *login* inicial, inserindo o seu nome de utilizador e uma palavra-passe. Os *tokens*, por sua vez, são criados através da encriptação JWT, onde o RefreshToken e o AccessToken são resultados da encriptação do *userID* com uma chave secreta distinta para cada um. Estas duas chaves são constantes e apenas o sistema *backend* tem acesso a elas.

A Figura 49 ilustra o processo de *login*, onde é feita uma chamada "POST /login" com as credenciais do utilizador, que está a tentar aceder à aplicação. A *API* cria um RefreshToken com um prazo de validade de uma hora e que é armazenado na base de dados, e um AccessToken com um prazo de validade de quinze minutos. O AccessToken é guardado no AuthProvider, que permite transportar a informação entre os ecrãs e que o AccessToken possa ser acedido em qualquer ponto da aplicação React.

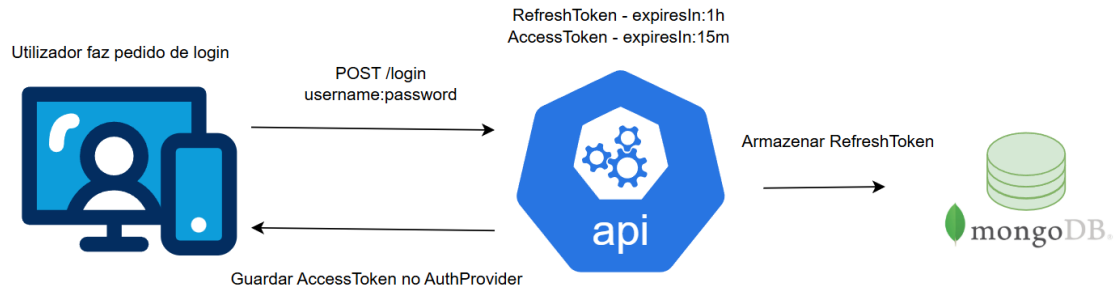


Figura 49 – Processo de *login* desenvolvido.

O processo de *logout*, Figura 50, inicia com o pedido para o "POST /logout", referindo o userID. A *API* envia uma *query* para apagar o RefreshToken na base de dados, e o AccessToken vai ser removido do AuthProvider.

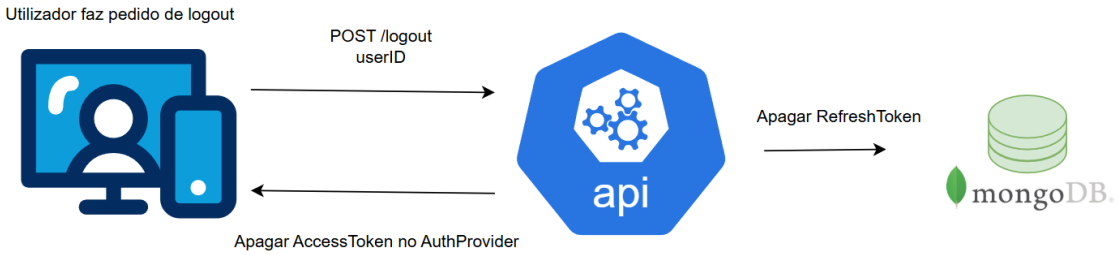


Figura 50 – Processo de *logout*.

A autenticação ocorre sempre que o utilizador faz a transição entre dois ecrãs da aplicação *web*. Isto permite que, antes de aceder ao novo ecrã, o utilizador esteja autenticado com as suas credenciais verificando, também, se o seu tempo de sessão ainda não se encontra expirado. A mostra o processo de autenticação onde o AccessToken está expirado, mas o RefreshToken ainda não. Ao navegar para um novo ecrã, a aplicação faz uma chamada para o "POST /authenticate" indicando o AccessToken e o userID. A *API*, por sua vez, faz uma *query* para obter o RefreshToken pelo userID e valida que este não está fora de prazo. A *API* valida, posteriormente, se o AccessToken está expirado e renová-lo por mais quinze minutos, sendo guardado no AuthProvider.

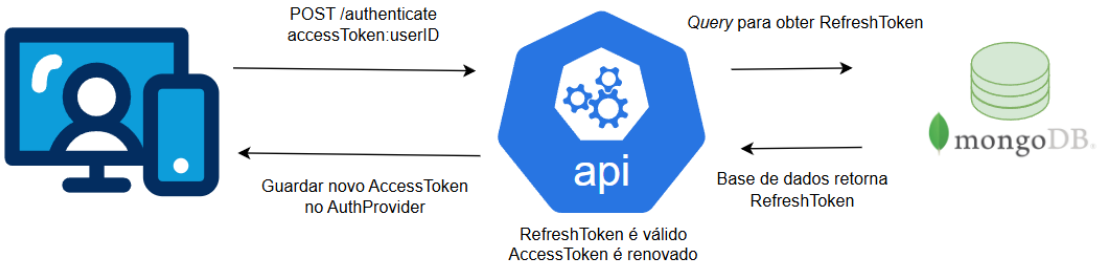


Figura 51 - Processo de autenticação com RefreshToken válido.

A Figura 52 ilustra um cenário onde o utilizador navega para um novo ecrã e tanto o AccessToken como o RefreshToken estão fora de prazo. A *API* ao validar que o RefreshToken está expirado, retorna um erro de autenticação e a aplicação leva o utilizador de volta ao ecrã de *login* para inserir as suas credenciais novamente.

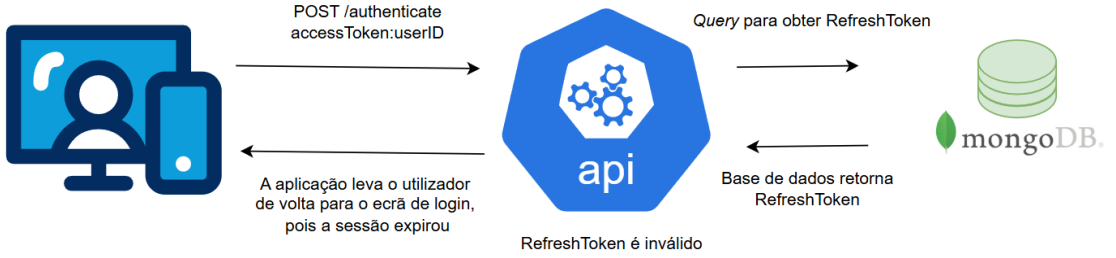


Figura 52 – Processo de autenticação com RefreshToken expirado.

4.3.4. Aplicação React

A aplicação React foi desenhada com a biblioteca Nivo para a construção da *dasboard*. A Nivo possui um vasto conjunto de módulos para implementar gráficos de evolução temporal, gráficos circulares e gráficos de barras, entre outros. A Figura 53 ilustra o processo de navegação da *dashboard* desenvolvida. O utilizador entra na *dashboard*, e o React faz um pedido de obtenção de dados "GET /data" para a API e esta, por sua vez, estabelece uma ligação à base dados e envia uma *query* para ser executada. A *API* aguarda assincronamente pela resposta da base de dados, e enquanto isso a *query* é executada e retorna os dados. A *API* responde com código de sucesso "200", juntamente com os dados requisitados. Estes dados vão ser usados pelas componentes Nivo para a representação gráfica.

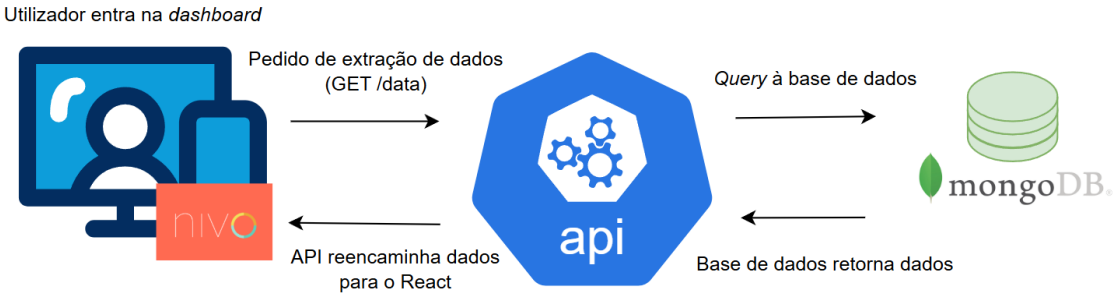


Figura 53 – Processo de obtenção de dados para a *dashboard*.

A *API* também é necessária noutros cenários de utilização, como o de atualizar parâmetros na aplicação, como se pode ver na Figura 54. O utilizador pode alterar o limite máximo e mínimo definido para os alarmes duma métrica da rede elétrica, para que as pessoas registadas na plataforma recebam um email quando existirem ocorrências em que o valor medido pelo sensor não se encontra dentro destes limites. O pedido para atualizar uma métrica é enviado para a *API*, através do "POST /alarm", e esta envia uma *query* para alterar o valor de um campo.

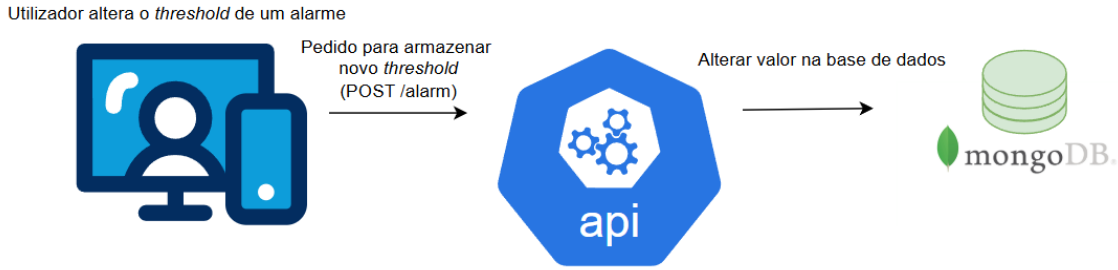


Figura 54 – Processo de atualizar o *threshold* de uma métrica.

Noutras casos de utilização, a *API* faz inserções na base de dados como, por exemplo, durante a criação de um novo utilizador por parte dos administradores. O primeiro administrador (utilizador: su/palavra-chave: ips) é armazenado na base de dados, caso não exista, assim que a aplicação for iniciada. Este processo é visível na Figura 55 onde, após o administrador preencher o formulário com os dados do novo *user* e fazer a submissão para o "PUT /user", se realiza um pedido para a *API* e esta enviar uma *query* para a base de dados armazenar a informação do novo utilizador.

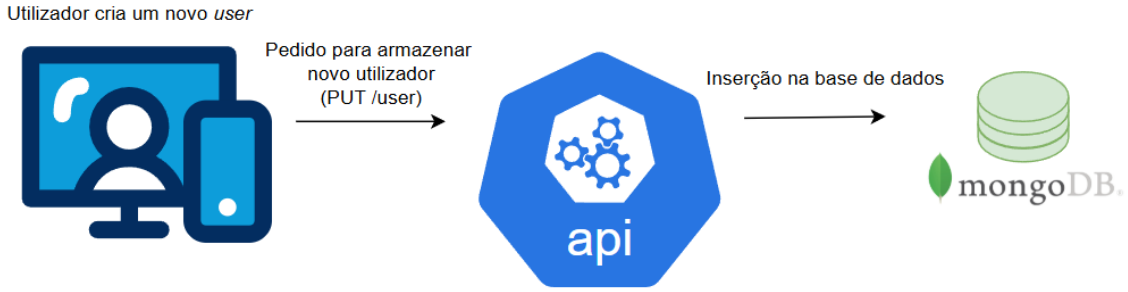


Figura 55 – Processo de inserir um novo utilizador na base de dados.

De forma oposta, também se pode apagar informação. Na Figura 56 encontra-se o processo para apagar um utilizador da base de dados. Se um administrador tomar a ação de apagar uma conta, a aplicação faz um pedido "DELETE /user" para a *API* que envia uma *query* para apagar o utilizador.

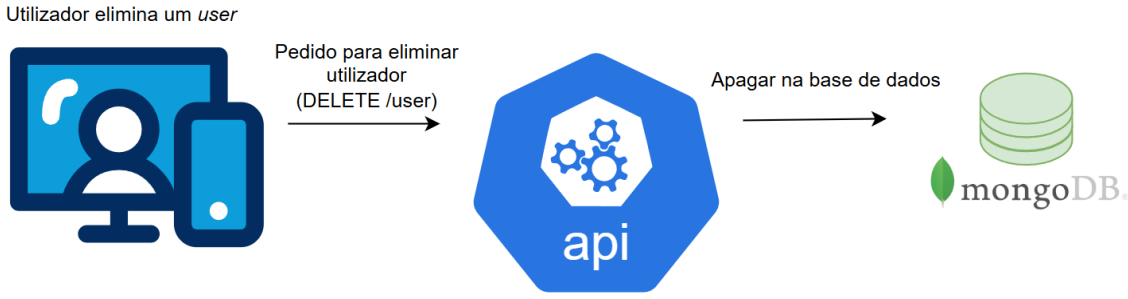


Figura 56 – Processo de eliminar um utilizador na base de dados.

4.4. Integração da rede

Neste subcapítulo é descrito o processo de integração do sensor, da aplicação final, e da TTN começando pela criação da aplicação TTN e do registo do sensor na plataforma. Descreve-se, depois, a metodologia para configurar o *MQTT*, criando credenciais para os clientes *MQTT* subscreverem o tópico da aplicação. Por fim, é mostrado a lógica do conector desenvolvido para ligação ao servidor TTN e ao servidor Ubuntu como também a lógica para enviar emails para envio de alarmes.

4.4.1. Configuração da aplicação e do sensor na plataforma TTN

Num primeiro passo cria-se a aplicação TTN, especificando a *application ID* e a *application name*, assim como a sua descrição como se pode observar na Figura 57.

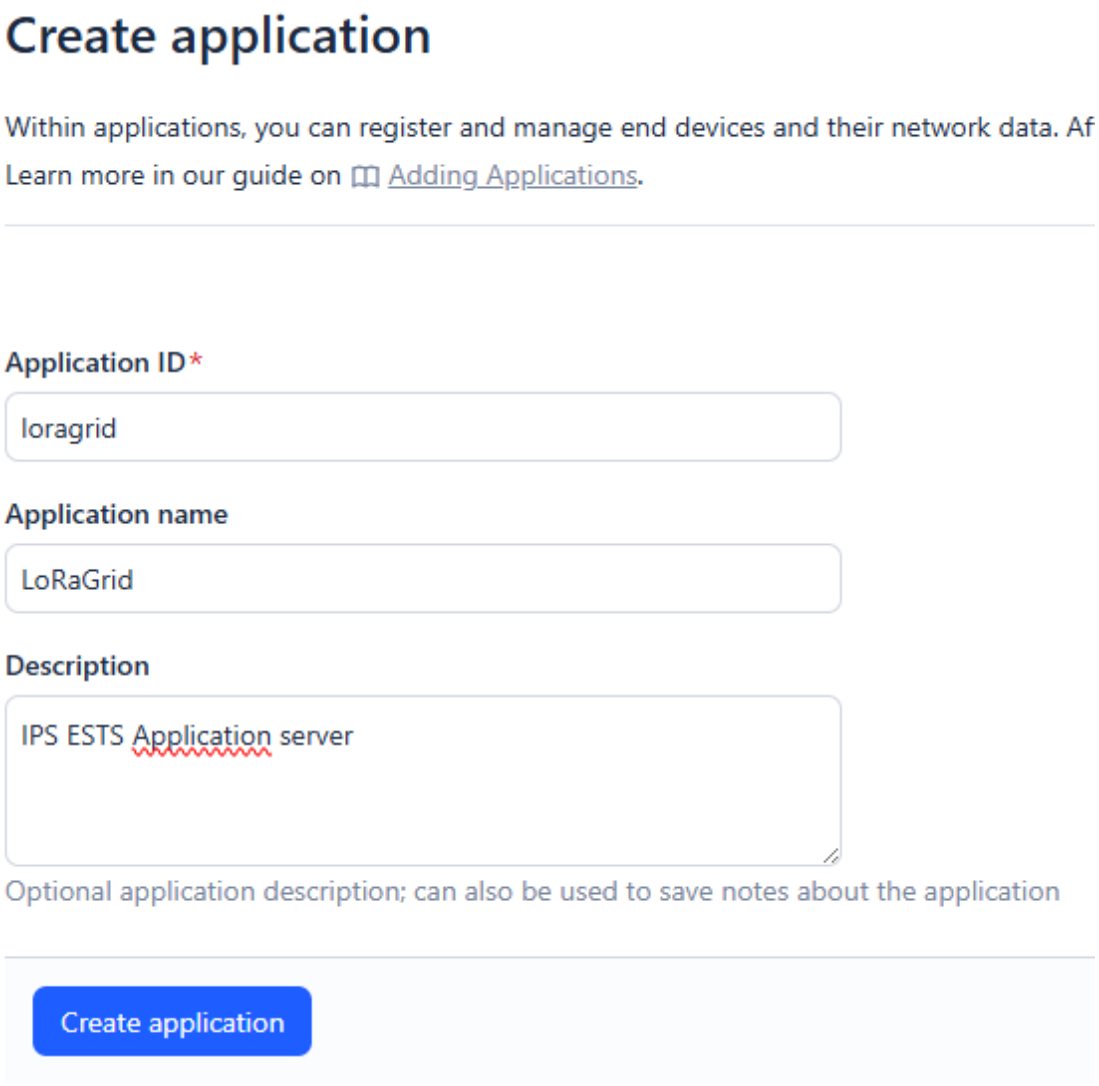


Figura 57 – Criação da aplicação na plataforma TTN.

Uma vez criada a aplicação TTN adiciona-se o dispositivo LoRa como se ilustra na Figura 58. Para tal especifica-se o plano de frequência 863-870MHz SF9 RX2 e a versão LoRaWAN 1.0.3.

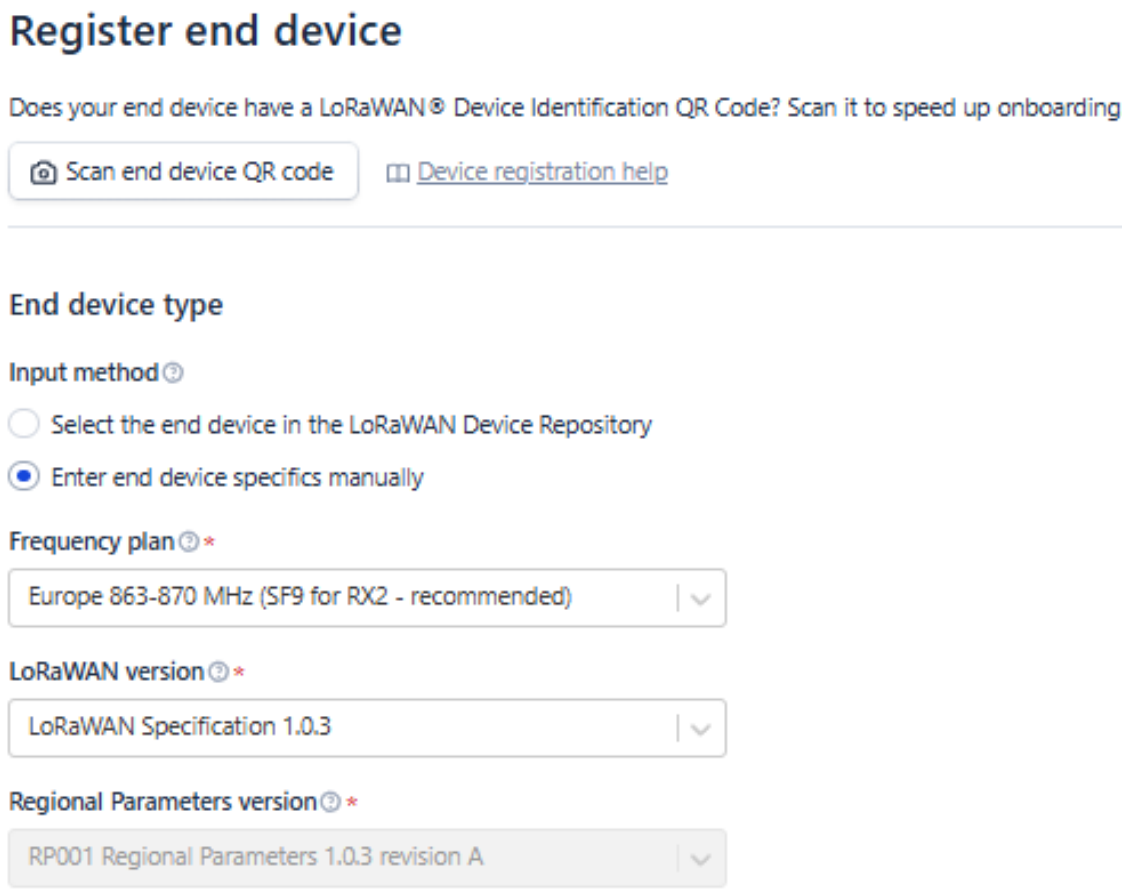


Figura 58 – Configuração do plano de frequências e versão *LoRaWAN*.

Dentro das opções para a autenticação do sensor com o servidor TTN, escolhe-se o método OTAA, uma vez que a sincronização das chaves é feita de forma automática e simplifica o processo de adição de sensores na rede. Para além disso, define-se o dispositivo como classe A, pelo que não se usam preâmbulos nem mantem o dispositivo continuamente ativo para comunicação, como acontece nas classes B e C. Esta configuração encontra-se na Figura 59.

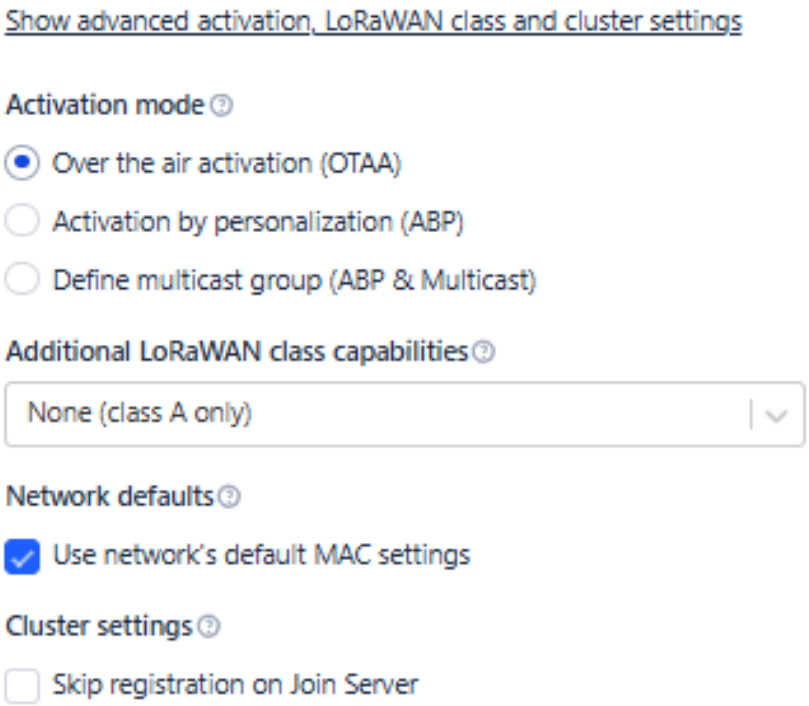


Figura 59 – Configuração do método de autenticação e da classe do sensor.

Os próximos passos encontram-se na Figura 60, em que se faz a especificação das chaves da comunicação. Configura-se o JoinEUI com um valor à escolha, neste caso com o valor de "0000000000000000". O campo DevEUI preenche-se com o ID do módulo LoRa-E5 Grove, "2CF7F1205260AB3A", que pode ser obtido pela leitura do seu código QR ou através do comando "AT+ID". Para o campo AppKey gera-se um valor aleatório, neste caso "C53F64126BC5976155076CEF798775". Este valor será configurado no módulo LoRa com o comando "AT+KEY=AppKey". Para o campo do EndDeviceID, dá-se o nome do sensor como "lora-sensor-ledrgb", uma vez que este sensor está ligado a uma lâmpada Led RGB.

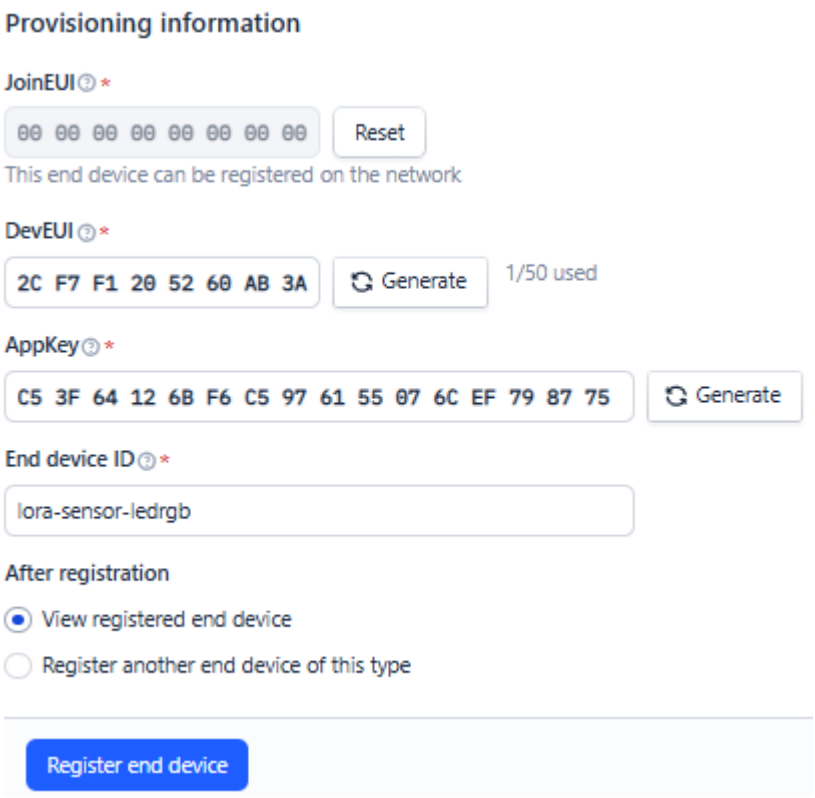


Figura 60 – Configuração das chaves JoinEUI, DevEUI e AppKey.

4.4.2. Configuração do MQTT

Dentro do ecrã de *Integrations* escolhe-se o *MQTT* e inicia-se a sua configuração, Figura 61. O processo de configuração é simples e basta criar uma password aleatória. Estas credenciais serão configuradas no conector TTN-Servidor desenvolvido, para que possa subscrever o tópico e extrair os dados das últimas leituras.

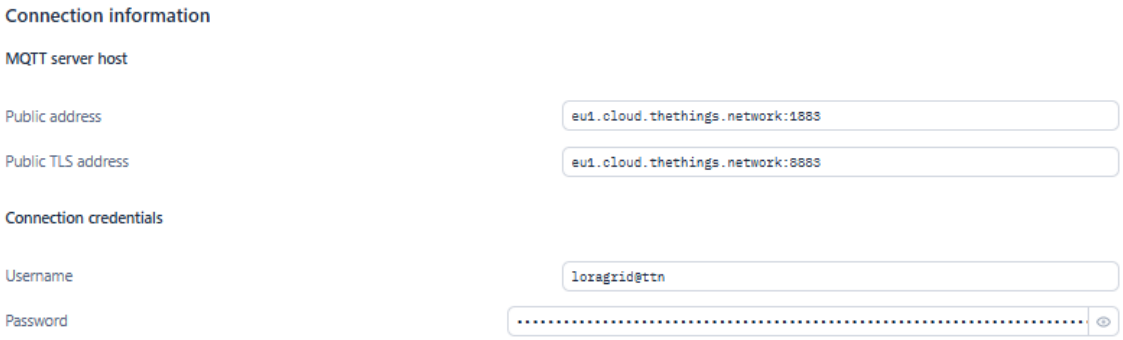


Figura 61 – Configuração do *MQTT*.

4.4.3. Conector TTN-Servidor

O conector que interliga o servidor *TTN* com o servidor local Ubuntu é um *script* python. Na Figura 62 encontra-se o fluxograma deste *script*, que começa por criar uma *thread* para a função que insere os dados sobre um novo sensor na base de dados. Este também faz as configurações inicias do cliente *MQTT*, a configuração da comunicação *MQTT* em modo TLS e a ligação ao *MQTT broker* da *TTN*.

Posteriormente, o conector aguarda para que haja novas mensagens no tópico. Ao receber uma nova mensagem este vai extrair da base de dados o último *payload* armazenado assim como o estado do sensor que enviou a nova mensagem. Algumas formatações são feitas para garantir a normalização dos dados e estruturar os dados para que se encontrem no formato *JSON* desejado. O conector verifica, de seguida, se a nova mensagem contém as mesmas medições que já estavam armazenadas recentemente ou se o sensor que transmitiu os dados está inativo. Em qualquer um dos casos a mensagem deve ser ignorada, uma vez que, ou a mensagem já foi armazenada e esta é um duplicado resultante do reenvio da mensagem por parte da *TTN* ou o sensor foi marcado na aplicação final como inativo e, portanto, deve ser desligado do sistema elétrico. A verificação da mensagem duplicada baseia-se na comparação entre o último *payload* armazenado e a medição mais recente.

Nesta comparação é verificada a *timestamp* assim como o nome do sensor que fez a medição e os respetivos dados da leitura. No cenário positivo em que a mensagem não foi armazenada recentemente nem o sensor se encontra como inativo, o conector verifica se o sensor tem algum registo na base de dados, inserindo os seus detalhes caso seja um novo sensor a transmitir dados, através de uma *thread*. Os dados da medição também são armazenados na base de dados. Antes do conector retornar ao começo da rotina, verifica se cada métrica está dentro dos limites configurados pela interface gráfica da aplicação *web* e, no caso de encontrar uma ou mais valores fora dos limites, envia um email para cada utilizador registado na plataforma. Para tal também é necessário que o sensor não esteja no modo de manutenção. Este modo pode ser ativado na plataforma *web* para situações em que se está a fazer manutenção nos quadros elétricos e, portanto, não devem ser enviados alarmes.

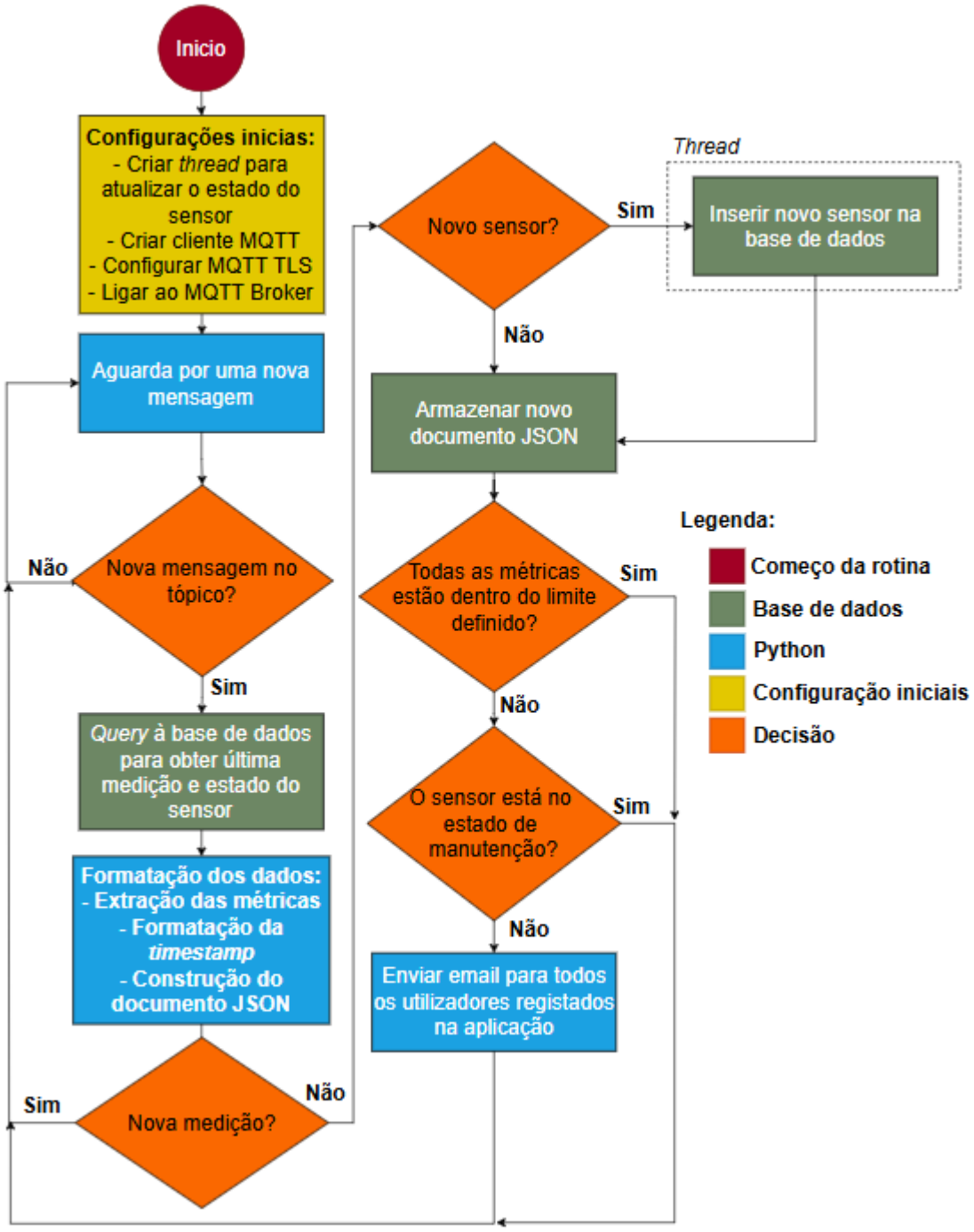


Figura 62 – Fluxograma do conector TTN-Servidor.

4.5. Resumo

Neste capítulo foi apresentado todo o desenho e implementação da arquitetura da solução. Foi feito uma análise dos requerimentos do sistema que se implementou, assim como o desenho do mesmo. Foi descrito o *gateway* adquirido pelo IPS, referindo os canais e as frequências de trabalho. Foi analisado as várias opções tanto de medidores de energia como de módulos LoRa. Tendo em conta as suas características escolheu-se o medidor PZEM-004t-100A por ser um medidor não-invasivo, por conseguir recolher uma quantidade de métricas diversas da rede elétrica e por ter uma interface de comunicação TTL, que facilitou a sua integração com o restante sensor. O LoRa-E5-Grove foi o módulo escolhido por ter uma interface Grove que permitiu a integração mais rápida com a restante arquitetura do sensor e por trabalhar na frequência 868MHz. Foi apresentado a arquitetura do sensor e a respetiva metodologia para a integração de cada módulo que o compõe, nomeadamente o Arduino Uno3, PZEM-004t-100A e o LoRa-E5-Grove. Também foi apresentado o desenho da aplicação final que mostra as interligações entre o *frontend* e o *backend*. Foi discutido a *API* desenvolvida para conectar tanto o *backend* como *frontend* da aplicação, sendo usada maioritariamente para manusear os dados da base de dados MongoDB enquanto o utilizador navega pela aplicação React. As chamadas para a *API* foram explicadas com exemplos de utilização da aplicação, com recurso de uma tabela de todas as chamadas da *API*, o método, o argumento e o corpo. Por fim, foi apresentado, passo-a-passo, o procedimento de configuração da aplicação *TTN* e de registo do sensor terminando com a explicação do conector construído para integrar a aplicação final e o servidor *TTN*.

5. Resultados e discussão

Neste capítulo é apresentado os resultados dos testes realizados para verificar a cobertura da rede desenvolvida em vários pontos do edifício da ESTSetúbal/IPS e para avaliar a funcionalidade de notificação de alarmes aos utilizadores. É explicado os passos de preparação dos testes e apresentam-se os resultados dos mesmos de seguida. Por fim, é feito uma discussão dos resultados obtidos realizando algumas comparações com os trabalhos analisados no capítulo 3 sobre os casos de estudos de aplicações semelhantes à deste projeto.

5.1. Testes de nível de sinal/Spreading Factor

Neste ponto é descrito os testes de nível de sinal e os resultados obtidos. É feito o planeamento e justificação da seleção dos pontos de instalação do sensor. Por fim é apresentado os resultados e é feito uma análise crítica dos mesmos.

5.1.1. Planeamento de testes

De modo a testar o nível de sinal de transmissão do sensor com *gateway* LoRa fez-se o planeamento do teste marcando três pontos considerados relevantes como se pode ver na Figura 63. O ponto um encontra-se no piso -1 do bloco F, estando aproximadamente a 16m de distância do *gateway*. Este ponto foi escolhido por se encontrar numa zona mais remota, uma vez que, entre o sensor e o *gateway* existem várias camadas de paredes de betão que podem potencialmente atenuar de forma significativa a transmissão. O ponto dois encontra-se no fundo do corredor do bloco E, no rés-do-chão em que a distância deste ponto ao *gateway* é de aproximadamente 55m. O ponto três é o cenário mais exigente de todos os pontos, pois encontra-se no piso -1 do bloco D, a cerca de 106m do *gateway*. Este ponto permite testar o nível de sinal num ponto onde existem muitos obstáculos que podem atenuar o sinal e a uma distância razoável do *gateway*.

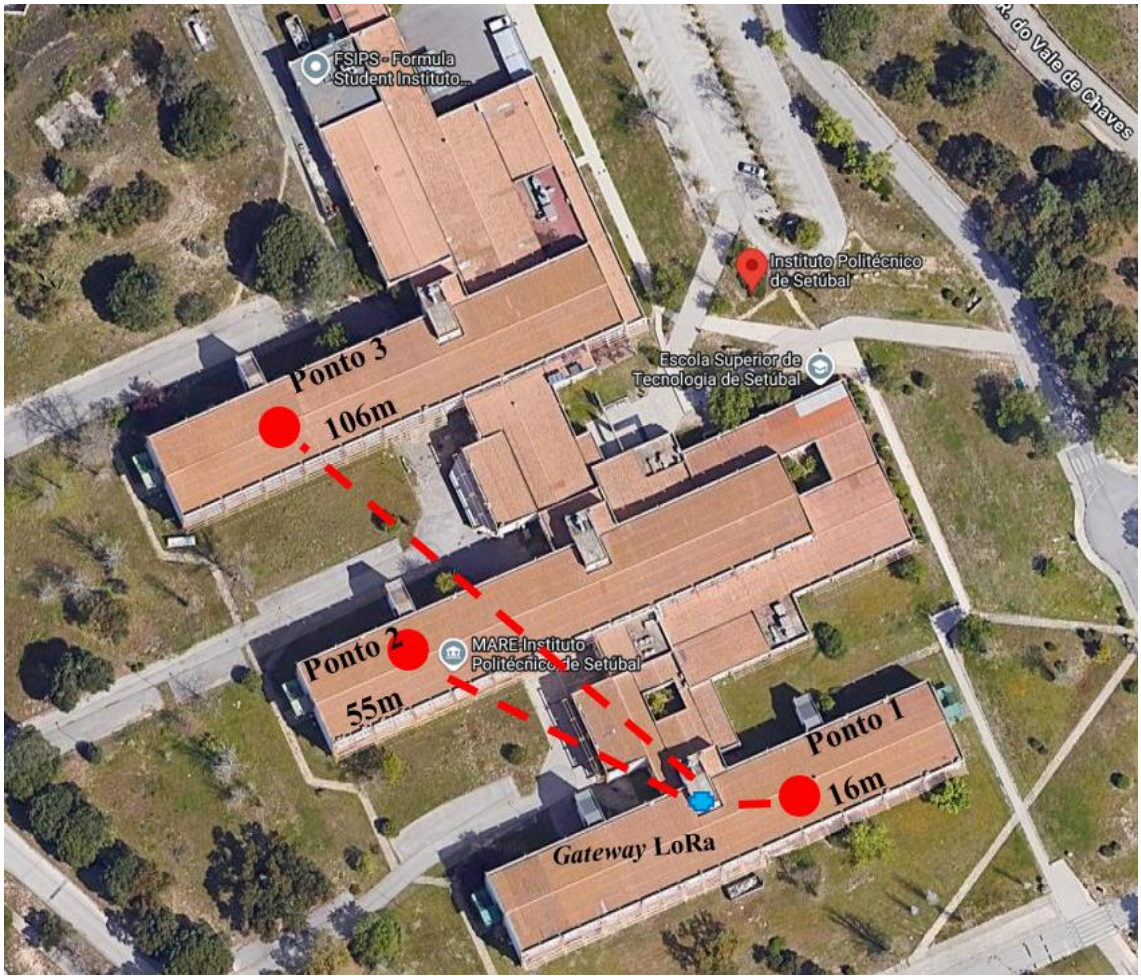


Figura 63 – Pontos de teste (Imagem retirada do Google Maps).

Em cada ponto realizam-se três configurações do *Spreading Factor*, nomeadamente *SF* = 7, *SF* = 9 e *SF* = 12 para uma BW de 125kHz, de modo a entender como o alcance e a taxa de transmissão são afetados pelas estruturas do edifício da ESTSetúbal/IPS. O sensor LoRa foi ligado a uma lâmpada LED de forma a monitorizar os consumos elétricos desta. De forma a garantir que o resultado obtido para cada configuração não represente um caso raro de condições, que tornaram possível o sensor ligar-se ao servidor ou de não se ligar ao mesmo, repetiu-se o teste três vezes.

5.1.2. Resultados

A avaliação de cada teste baseou-se em dois critérios. O primeiro critério para aprovar um teste assenta no número de vezes que o sensor tentou ligar-se ao servidor *TTN* e não teve sucesso. O dispositivo está configurado para tentar ligar-se indefinidamente à *TTN* até que a ligação seja estabelecida, mas para fins práticos reprova-se o teste caso o sensor, ao fim de dez tentativas, não consiga comunicar com o servidor *LoRaWAN*.

O segundo critério de avaliação é medido pela perda de pacotes, pois o sensor pode estar ligado à *TTN*, mas não conseguir enviar informação. Se o número de pacotes recebidos for bastante reduzido ou nulo, então isso pode significar que a potência do sinal transmitido é muito baixa e o recetor não está a conseguir demodular a informação, o que também vai ser avaliado como um teste reprovado. Com estes dois critérios foi possível identificar cenários onde o alcance de transmissão é viável e estável para a transmissão dos dados.

Os resultados dos testes encontram-se na Tabela 5. Os testes foram realizados no período do almoço, quando havia muitos alunos a frequentar a escola e a utilizar a rede Wi-Fi e outras formas de comunicação de radiofrequência. Esta variável podia condicionar os resultados devido ao espaço de radiofrequências, mas todos os cenários de testes foram bem-sucedidos. Realizou-se o teste para cada cenário uma única vez.

Tabela 5 – Resultados do teste de nível de sinal.

Bloco / SF	SF7			SF9			SF12		
F	SNR: 9.8 RSSI: -83	SNR: 7.2 RSSI: -82	SNR: 7.2 RSSI: -85	SNR: 12.5 RSSI: -82	SNR: 12 RSSI: -86	SNR: 10 RSSI: -90	SNR: 12 RSSI: -88	SNR: 13 RSSI: -85	SNR: 9.8 RSSI: -83
D	SNR: 5.2 RSSI: -110	SNR: -3.8 RSSI: -116	SNR: -1.2 RSSI: -114	SNR: -1.2 RSSI: -114	SNR: -1.2 RSSI: -114	SNR: -5.5 RSSI: -116	SNR: 4 RSSI: -114	SNR: -2.8 RSSI: -118	SNR: -2 RSSI: -116
E	SNR: 9.5 RSSI: -85	SNR: 8.2 RSSI: -88	SNR: 9 RSSI: -85	SNR: 9.8 RSSI: -82	SNR: 10.8 RSSI: -98	SNR: -7.8 RSSI: -103	SNR: 13.2 RSSI: -90	SNR: 10.5 RSSI: -88	SNR: 11 RSSI: -79

Na Figura 64 apresentam-se os registos da consola do *TTN*, onde se encontram os pacotes enviados para cada configuração do *SF* no ponto F. Para o *SF7* o *SNR* é de 9.8dB e o *RSSI* foi de -83dBm. À medida que se aumentou o *SF* o *SNR* e o *RSSI* diminuíram. Para *SF9* o *SNR* foi 12.5dB e o *RSSI* -82dBm, enquanto para *SF12* o *SNR* e *RSSI* foram 12dB e -88dBm, respetivamente.

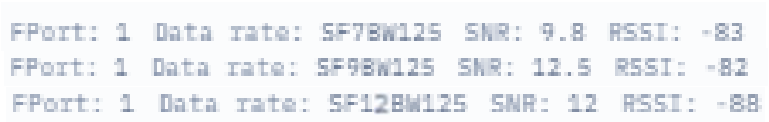


Figura 64 - Registo da chegada de pacotes do primeiro teste.

Na Figura 65 encontram-se os registos efetuados para o segundo teste, onde se notou o mesmo padrão. Para *SF7* o *SNR* foi de 9.5dB e o *RSSI* foi -85dBm. Para *SF9* o *SNR* foi 8.8dB e o *RSSI* -82dBm e no caso do SF12 o *SNR* e *RSSI* foram de 7.8dB e -100dBm, respetivamente.

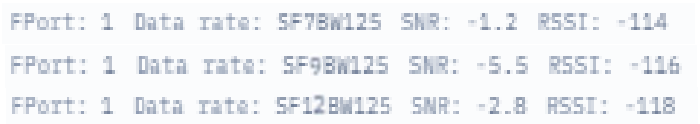


Figura 65 – Registo da chegada de pacotes do segundo teste.

Para o último cenário de teste, no ponto E, o rácio de sinal-ruido ficou abaixo do zero, mas, ainda assim, o recetor foi capaz de demodular o sinal transmitido. Nos testes realizados, Figura 66, com o *SF7* o *SNR* foi 9.5dB e o *RSSI* foi de -85dBm. Para o *SF9* o *SNR* foi de 9.8dB e o -82dBm, enquanto para o *SF12* o *SNR* foi 13.2dB e o *RSSI* foi de -98dBm.

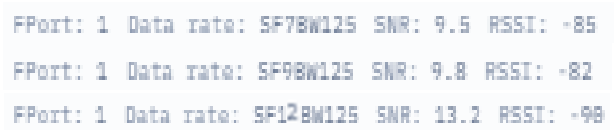


Figura 66 - Registo da chegada de pacotes do terceiro teste.

Na Figura 67 podem visualizar-se os dados que foram medidos durante os testes de alcance tal como apresentados na aplicação desenvolvida. Como se pode observar a interface é intuitiva e fácil de usar e permite ver várias métricas do consumo da lâmpada LED. Aqui são visíveis alguns intervalos correspondentes ao tempo em que não ocorreram medições, e estes representam o momento de reprogramação do *Spreading Factor* do sensor.

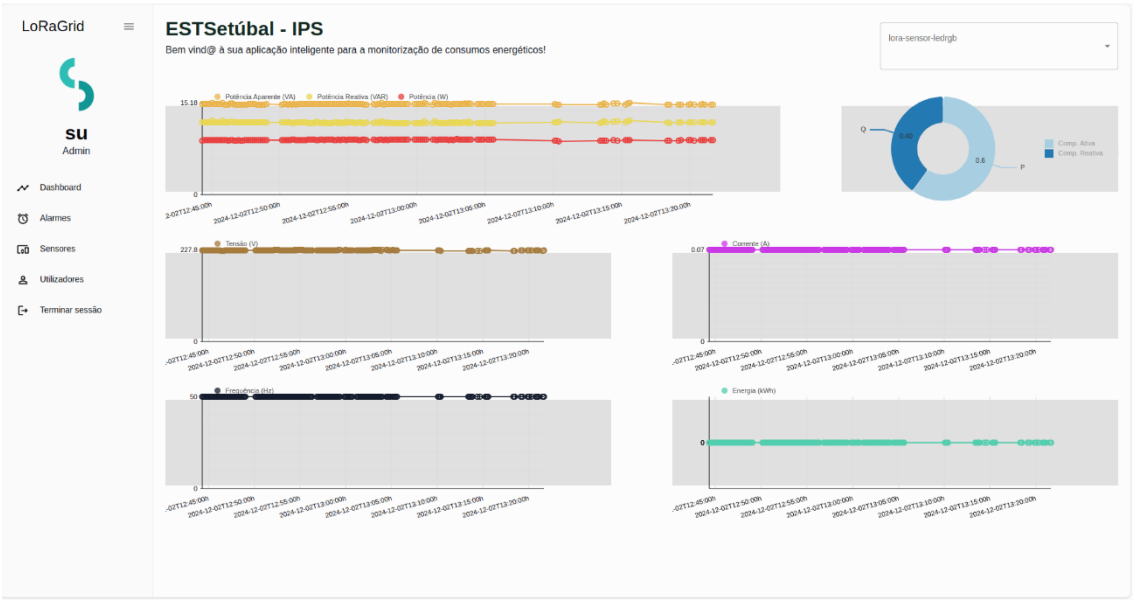


Figura 67 – Visualização dos dados medidos na *dashboard* da aplicação React.

5.2. Teste de alarmes

Neste ponto ilustra-se a configuração dos alarmes para detetar alguns cenários. A configuração fez-se na aplicação *web*, e os resultados mostram a notificação das anomalias através do email.

5.2.1. Configuração dos alarmes

Para preparar o teste de alarmes entra-se no ecrã de alarmes da aplicação *web* e configuram-se os limites de cada métrica. Os campos com o valor de -1 traduzem-se em alarmes desabilitados. Na Figura 68, encontra-se a configuração de um alarme para quando a potência ativa da lâmpada é inferior a 1.5W, o que pode ajudar na deteção de lâmpadas queimadas, ou circuitos abertos. Apresenta-se, também, a configuração para um alarme quando a corrente é maior que 0.05A na ótica de controlar a corrente da lâmpada e detetar situações onde esta está em sobrecarga, ou com a luminosidade no máximo brilho que também causa um acréscimo consumo energético. De modo a reproduzir-se estes cenários bastou desligar a lâmpada reduzindo instantaneamente a potência ativa para 0W e aumentar o brilho da mesma para também aumentar a corrente do circuito.

Figura 68 – Configuração dos alarmes na aplicação *web*.

5.2.2. Resultados

Os resultados resumem-se no email enviado pelo sistema a alertar a ocorrência de uma anomalia detetada na medição do sensor. Na Figura 69 encontra-se a notificação do sistema quando se detetou uma medição com a potência ativa era inferior a 1.5W e na Figura 70 uma notificação relativa ao valor da corrente acima dos 0.05A. Note-se que os alarmes têm a informação do sensor onde a anomalia foi detetada, assim como a data e hora da deteção, para além dos dados da medição.

Figura 69 – Alarme de notificação para potência inferior a 1.5W detetada.

Figura 70 - Alarme de notificação para corrente superior a 0.05A detetada.

5.3. Análise dos resultados

Os testes realizados permitiram avaliar a arquitetura desenvolvida quanto aos alcances de transmissão e à funcionalidade de notificação por email. Os resultados dos testes de alcance foram positivos e mostraram alguns pontos interessantes da transmissão LoRa, nomeadamente, os valores de *SNR* desceram e os de *RSSI* aumentaram com o aumento do SF. Isto deve-se ao facto de que o transmissor levar tanto mais tempo para transmitir um *Chirp* quanto maior for o *SF* e, portanto, a potência do sinal é distribuída por mais tempo refletindo-se numa redução no ganho como se pode confirmar pelos valores do *RSSI*. No entanto, o teste demonstrou que a rede *LoRa* desenvolvida mesmo em locais remotos da escola foi capaz de transmitir dados, o que viabiliza a instalação de mais sensores e a respetiva monitorização de mais pontos da rede elétrica da instituição.

Os testes realizados para os alarmes, por sua vez, permitiram concluir que os utilizadores podem ser notificados de falhas nas cargas elétricas e também de valores anormais. O *website* desenvolvido é intuitivo e a navegação dentro desta assim como a configuração dos alarmes é fácil de realizar.

A solução desenvolvida tem a funcionalidade dos alarmes que as soluções analisadas nos casos de estudos não tinham e que, do ponto de vista da implementação, é importante numa arquitetura para a monitorização de consumos elétricos e deste projeto em particular. No entanto, o trabalho desenvolvido carece de um cálculo de faturação que foi uma característica presente nos trabalhos analisados. O nó sensor da solução desenvolvida tem uma arquitetura simples e que envolve apenas um microprocessador ao contrário dos nós sensores das arquiteturas dos casos de estudo que recorriam a um medidor robusto ou até a dois microprocessadores para a medição das métricas. A arquitetura também não permite o controlo remoto dos sensores através da aplicação *web*, que é algo presente no primeiro caso de estudo. Contudo, é possível de adicionar esta funcionalidade alterando a aplicação e o conector para que se possa enviar mensagens para o sensor, e programar o sensor para executar comandos que venham da aplicação, nomeadamente.

5.4. Resumo

Neste capítulo fez-se o planeamento dos testes e a apresentação dos resultados. Conclui-se que o sistema implementado é capaz de trabalhar em zonas mais remotas e rodeadas de obstáculos, como também traz uma visão clara dos dados em gráficos e notificações aos utilizadores. Comparou-se este trabalho com os casos de estudos analisados previamente no capítulo 3 concluindo-se que o nó sensor desenvolvido neste projeto é mais simples, mas a arquitetura necessita de algumas funcionalidades presentes nos outros projetos, como o sistema de cálculo de faturação e a capacidade de controlar os sensores remotamente através da aplicação *web*.

6. Conclusão

Com esta tese pretendeu-se construir um sistema LoRa para fazer a monitorização de consumos elétricos no IPS que fosse eficiente, tivesse uma arquitetura simples, que contribuísse para sustentabilidade do IPS, fosse seguro, e que fosse independente de tecnologias de terceiros. Com isto não se conseguiu cumprir todos os objetivos, pois a solução utiliza o serviço TTN como servidor de rede LoRaWAN. No entanto, foi possível cumprir com os outros objetivos, nomeadamente:

- Construir um sensor de baixa complexidade, baixo custo e fácil instalação para a medição de consumos elétricos, capaz de obter um conjunto de informações relevantes para a análise objetiva dos consumos de energia elétrica no IPS.
- Desenvolver uma aplicação *web* para a apresentação dos dados e configuração de alarmes para as métricas obtidas a partir dos sensores.
- Implementar uma arquitetura *IoT* assente numa rede de sensores LoRa devidamente integrada desde os sensores à aplicação.
- Apresentar uma solução que contribua para a sustentabilidade do IPS, na medida em que permite a recolha de dados para futura caracterização do consumo energético na Instituição e, assim, permitir melhorar a sua eficiência energética.
- Garantir que o sistema seja seguro e faça o permita o controlo do acesso aos dados.

Conclui-se que esta solução cumpre os requisitos para a monitorização dos consumos elétricos na Instituição, e representa um ponto de partida para a implementação de futuros projetos do IPS como a da instalação de paneis fotovoltaicos. Quanto aos tópicos de trabalho futuro sugere-se um conjunto de melhorias para este protótipo. Sugere-se utilizar soluções *open-source* como o ChirpStack ao invés da TTN, para implementar um servidor de rede LoRa num computador da escola e que não fique dependente de terceiros ficando, assim, a escola com controlo total da arquitetura *IoT* desenvolvida. Outro ponto a ponderar é a remoção do Arduino da arquitetura do medidor, pois o módulo LoRa-E5 Grove tem um microcontrolador STM32WLE5JC, que potencialmente pode substituir o processador do Arduino para integrar e controlar módulos.

Neste caso, o STM32WLE5JC pode ser programado com o IDE STM32CubeMX. Também será relevante o desenho de uma placa de circuito impresso para a construção integrada do nó sensor. Para além disso o sistema desenvolvido apenas permite que os dados fluam do sensor para o servidor *LoRa* e aplicação final, posteriormente. Será interessante para as soluções futuras permitir que a aplicação final possa controlar diretamente os sensores. Nesta solução, por uma questão de simplicidade, optou-se por uma alimentação do sensor através de pilhas mas o nó sensor pode ser alimentado através da instalação elétrica que está a monitorizar reduzindo a necessidade de intervir no sensor quando a bateria acaba.

Bibliografia

[1] K. Foote, "A Brief History of the Internet of Things," 2022 Dataversity, [Acedido: 08/06/2024] Link: https://www.dataversity.net/brief-history-internet-things/

[2] Visiat, "How Internet of Things (IoT) technology helps address climate change and environmental challenges," 2024 Viasat, [Acedido: 08/06/2024] Link: https://news.viasat.com/blog/corporate/how-internet-of-things-iot-technology-helps-address-climate-change-and-environmental-challenges

[3] Computador raspberry PI, Raspberry PI Ltd, [Acedido: 08/06/2024] Link: https://www.raspberrypi.com

[4] A. Mishra, J. Srivastav, M. Srivastav, "Real - Time IoT - Based Energy and Power Monitoring and Management System for Small - Scale Applications Using a Raspberry Pi Web Interface," 2022 Internation Journal of Science and Research (IJSR), [Acedido: 08/06/2024] Link: https://www.ijsr.net/archive/v13i6/SR24603083200.pdf

[5] ETSI, "Smart Grids and Meters," 2023 ETSI IoT Conference (ETSI IoT Week 2023), [Acedido: 08/06/2024] Link: https://www.etsi.org/technologies/smart-grids-and-meters

[6] H. Atlam, R. Walters, G. Wills "Internet of Things: State-of-the-art, Challenges, Applications, and Open Issues," 2018 Internation Journal of Intelligent Computing Research (IJICR), 2018, volume 9, issue 3, [Acedido: 09/06/2024] Link: https://infonomics-society.org/wpcontent/uploads/ijicr/published-papers/volume-9-2018/Internet-of-Things-State-of-the-art-Challenges-Applications-and-Open-Issues.pdf

[7] T. Lynn, P. Endo, A. Ribeiro, G. Barbosa, P. Rosati. (2020). The Internet of Things: Definitions, Key Concepts, and Reference Architectures, [Acedido: 10/06/2024] Link: https://www.researchgate.net/publication/342744519_The_Internet_of_Things_Definitions_Key_Concepts_and_Reference_Architectures

[8] World Economic Forum, "The effect of the Internet of Things on sustainability," 2018 World Economic Forum Annual Meeting, [Acedido: 10/06/2024] Link: https://www.weforum.org/agenda/2018/01/effect-technology-sustainability-sdgs-internet-things-iot/

[9] N. SHARMA, D. PANWAR, "GREEN IOT: ADVANCEMENTS AND SUSTAINABILITY WITH ENVIRONMENT BY 2050," 2020 8TH INTERNATIONAL CONFERENCE ON RELIABILITY, INFOCOM TECHNOLOGIES AND OPTIMIZATION (TRENDS AND FUTURE DIRECTIONS) (ICRITO), NOIDA, INDIA, 2020, PP. 1127-1132, [Acedido: 11/06/2024] LINK: https://ieeexplore.ieee.org/abstract/document/9197796

[10] WORLD HEALTH ORGANIZATION, "THE WORLD HEALTH ORGANIZATION QUALITY OF LIFE (WHOQOL)," 2022 GUIDANCE, [Acedido: 12/06/2024] LINK: https://www.who.int/publications/i/item/WHO-HIS-HSI-Rev.2012.03

[11] S. BARAKOVIĆ, J. BARAKOVIĆ, D. MARAJ, A. MARAJ, O. KREJCAR, P. MARESOVA, F.J. MELERO, "QUALITY OF LIFE, QUALITY OF EXPERIENCE, AND SECURITY PERCEPTION IN WEB OF THINGS: AN OVERVIEW OF RESEARCH OPPORTUNITIES." 2020 ELECTRONICS, [Acedido: 15/06/2024] LINK: https://www.mdpi.com/2079-9292/9/4/700

[12] M. ALY, F. KHOMH, Y. -G. GUÉHÉNEUC, H. WASHIZAKI, S. YACOUT, "IS FRAGMENTATION A THREAT TO THE SUCCESS OF THE INTERNET OF THINGS?" 2019 IEEE INTERNET OF THINGS JOURNAL, VOL. 6, NO. 1, PP. 472-487, [ACEDIDO: 15/06/2024] LINK: https://ieeexplore.ieee.org/document/8424819

[13] W3C, "MAKING THE WEB WORK," 2024, [ACEDIDO: 18/06/2024] LINK: https://www.w3.org

[14] K. WAC, M. FIORDELLI, M. GUSTARINI, H. RIVAS, "QUALITY OF LIFE TECHNOLOGIES: EXPERIENCES FROM THE FIELD AND KEY CHALLENGES," 2015 IEEE INTERNET COMPUTING, VOL. 19, NO. 4, PP. 28-35, [ACEDIDO: 27/06/2024] LINK: https://ieeexplore.ieee.org/abstract/document/7106384?casa_token=htFM23sqiZMAAAAA:uJsFz2a5SnuAMu_B47DhXBCMKuiq2ZM4_zZq8ta640lJt857Hi7heOr0GkjZs9ULFkxODAN-J2c

[15] J. FERNANDEZ, "IOT MARKET UPDATE: ENTERPRISE IOT MARKET SIZE REACHED $269 BILLION IN 2023, WITH GROWTH DECELERATION IN 2024," 2024 IOT ANALYTICS ARTICLE, [ACEDIDO: 27/06/2024] LINK: https://iot-analytics.com/iot-market-size/

[16] H. SINGH, "IOT DECONSTRUCTED," 2020 CAPGEMINI INSIGHT, [ACEDIDO: 28/06/2024] LINK: https://www.capgemini.com/insights/expert-perspectives/iot-deconstructed/

[17] ASSISTENTE INTELIGENTE ALEXA, AMAZON, [ACEDIDO: 28/06/2024] LINK: https://www.alexa.com/

[18] ASSISTENTE INTELIGENTE GOOGLE, GOOGLE, [ACEDIDO: 28/06/2024] LINK: https://assistant.google.com/intl/pt_pt/

[19] D. OMEIZA, H. WEBB, M. JIROTKA, L. KUNZE, "EXPLANATIONS IN AUTONOMOUS DRIVING: A SURVEY," 2022 IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 23, NO. 8, PP. 10142-10162, [ACEDIDO: 30/06/2024] LINK: https://ieeexplore.ieee.org/document/9616449

[20] M. BURHAN, R.A. REHMAN, B. KHAN, B.-S KIM, "IOT ELEMENTS, LAYERED ARCHITECTURES AND SECURITY ISSUES: A COMPREHENSIVE SURVEY", 2018 SENSORS, [ACEDIDO: 30/06/2024] LINK: https://www.mdpi.com/1424-8220/18/9/2796

[21] S. VASHI, J. RAM, J. MODI, S. VERMA, C. PRAKASH, "INTERNET OF THINGS (IOT): A VISION, ARCHITECTURAL ELEMENTS, AND SECURITY ISSUES," 2017 INTERNATIONAL CONFERENCE ON I-SMAC (IOT IN SOCIAL, MOBILE, ANALYTICS AND CLOUD) (I-SMAC), PALLADAM, INDIA, 2017, PP. 492-496, [ACEDIDO: 30/06/2024] LINK: https://ieeexplore.ieee.org/document/8058399

[22] P. FREMANTLE, "A REFERENCE ARCHITECTURE FOR THE INTERNET OF THINGS," WSO2 2015 WHITE PAPER, [ACEDIDO: 30/06/2024] LINK: https://wso2.com/whitepapers/a-reference-architecture-for-the-internet-of-things/

[23] MDN, "UMA VISÃO GERAL DO HTTP," 2024 MDN WEB DOCS ARTICLE, [ACEDIDO: 30/06/2024] LINK: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview

[24] C. BORMANN, A. P. CASTELLANI, Z. SHELBY, "COAP: AN APPLICATION PROTOCOL FOR BILLIONS OF TINY INTERNET NODES," 2012 IEEE INTERNET COMPUTING, VOL. 16, NO. 2, PP. 62-67, [ACEDIDO: 30/06/2024] LINK: https://ieeexplore.ieee.org/document/6159216

[25] J. GUTH, U. BREITENBÜCHER, M. FALKENTHAL, F. LEYMANN, L. REINFURT, "COMPARISON OF IOT PLATFORM ARCHITECTURES: A FIELD STUDY BASED ON A REFERENCE ARCHITECTURE," 2016 CLOUDIFICATION OF THE INTERNET OF THINGS (CIOT), PARIS, FRANCE, 2016, PP. 1-6, [ACEDIDO: 02/07/2024] LINK: https://ieeexplore.ieee.org/abstract/document/7872918

[26] D. SEHRAWAT, N. S. GILL, "SMART SENSORS: ANALYSIS OF DIFFERENT TYPES OF IOT SENSORS," 2019 3RD INTERNATIONAL CONFERENCE ON TRENDS IN ELECTRONICS AND INFORMATICS (ICOEI), TIRUNELVELI, INDIA, 2019, PP. 523-528, [ACEDIDO: 03/07/2024] LINK: https://ieeexplore.ieee.org/document/8862778

[27] K. SAJJA, B. BHOWMIK, "SENSOR CLASSIFICATIONS AND THEIR APPLICATIONS IN IOT SYSTEMS," 2023 IEEE INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING, VLSI, ELECTRICAL CIRCUITS AND ROBOTICS (DISCOVER), MANGALORE, INDIA, 2023, PP. 293-298, [ACEDIDO: 04/07/2024] LINK: https://ieeexplore.ieee.org/document/10316731

[28] R. AHMAD, R. WAZIRALI; T. ABU-AIN, "WIRELESS SENSOR NETWORK," 2022 ENCYCLOPEDIA, [ACEDIDO: 05/07/2024] LINK: https://encyclopedia.pub/entry/25316

[29] IBM, "WHAT IS NETWORK TOPOLOGY?", 2024, [ACEDIDO: 05/07/2024] LINK: https://www.ibm.com/topics/network-topology

[30] BLUETOOTH, "BLUETOOTH® WIRELESS TECHNOLOGY", 2024, [ACEDIDO: 08/07/2024] LINK: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[31] ZIGBEE, "WHAT IS ZIGBEE?", 2024, [ACEDIDO: 12/07/2024] LINK: https://www.ti.com/technologies/wired-wireless-connectivity/zigbee/overview.html

[32] WI-FI ALLIANCE, "DISCOVER WI-FI", 2024, [ACEDIDO: 12/07/2024] LINK: https://www.wi-fi.org/discover-wi-fi

[33] AMAZON AWS, "WHAT IS AWS IOT CORE FOR LORAWAN?", 2024, [ACEDIDO: 12/07/2024] LINK: https://docs.aws.amazon.com/iot-wireless/latest/developerguide/what-is-iot-lorawan.html#what-is-lorawan

[34] GSMA, "NARROWBAND – INTERNET OF THINGS (NB-IOT)", 2024, [ACEDIDO: 12/07/2024] LINK: https://www.gsma.com/solutions-and-impact/technologies/internet-of-things/narrow-band-internet-of-things-nb-iot/

[35] GSMA, "LONG TERM EVOLUTION FOR MACHINES: LTE-M", 2024, [ACEDIDO: 12/07/2024] LINK: https://www.gsma.com/solutions-and-impact/technologies/internet-of-things/long-term-evolution-machine-type-communication-lte-mtc-cat-m1/

[36] WI-FI ALLIANCE, "WI-FI CERTIFIED HALOW", 2024, [ACEDIDO: 13/07/2024] LINK: https://www.wi-fi.org/discover-wi-fi/wi-fi-certified-halow

[37] QUALCOMM, "EVERYTHING YOU NEED TO KNOW ABOUT 5G.", 2024, [ACEDIDO: 13/07/2024] LINK: https://www.qualcomm.com/5g/what-is-5g

[38] U. RAZA, P. KULKARNI, M. SOORIYABANDARA, "LOW POWER WIDE AREA NETWORKS: AN OVERVIEW," 2017 IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 19, NO. 2, PP. 855-873, [ACEDIDO: 13/07/2024] LINK: https://ieeexplore.ieee.org/document/7815384

[39] L. QIAO, Z. ZHENG, W. CUI, L. WANG, "A SURVEY ON WI-FI HALOW TECHNOLOGY FOR INTERNET OF THINGS," *2018 2ND IEEE CONFERENCE ON ENERGY INTERNET AND ENERGY SYSTEM INTEGRATION (EI2)*, BEIJING, CHINA, 2018, PP. 1-5, [ACEDIDO: 14/07/2024] LINK: https://ieeexplore.ieee.org/document/8582141

[40] SOLUÇÃO SATÉLITE STARLINK, [ACEDIDO: 18/07/2024] LINK: https://www.starlink.com

[41] E. PAGLIARI, "INTERNET OF THINGS WIRELESS NETWORKS: WHICH ARE THE MOST COMMON IOT WIRELESS COMMUNICATION PROTOCOLS?", 2024, [ACEDIDO: 26/07/2024] LINK: https://emanuelepagliari.it/2020/10/13/internet-of-things-wireless-communication-protocols

[42] K. CAO, Y. LIU, G. MENG, Q. SUN, "AN OVERVIEW ON EDGE COMPUTING RESEARCH," 2020 IEEE ACCESS, VOL. 8, PP. 85714-85728, [ACEDIDO: 27/07/2024] LINK: https://ieeexplore.ieee.org/abstract/document/9083958

[43] SEMTECH, "WHAT IS LORA?", 2024, [ACEDIDO: 31/07/2024] LINK: https://www.semtech.com/lora/what-is-lora

[44] A. A. KHERANI, K. M. P. MAURYA, "IMPROVED PACKET DETECTION IN LORA-LIKE CHIRP SPREAD SPECTRUM SYSTEMS," 2019 IEEE INTERNATIONAL CONFERENCE ON ADVANCED NETWORKS AND TELECOMMUNICATIONS SYSTEMS (ANTS), GOA, INDIA, 2019, PP. 1-4, [ACEDIDO: 01/08/2024] LINK: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9118076

[45] M. N. OCHOA, A. GUIZAR, M. MAMAN, A. DUDA, "EVALUATING LORA ENERGY EFFICIENCY FOR ADAPTIVE NETWORKS: FROM STAR TO MESH TOPOLOGIES," 2017 IEEE 13TH INTERNATIONAL CONFERENCE ON WIRELESS AND MOBILE COMPUTING, NETWORKING AND COMMUNICATIONS (WIMOB), ROME, ITALY, 2017, PP. 1-8, [ACEDIDO: 01/08/2024] LINK: https://ieeexplore.ieee.org/document/8115793

[46] THE THINGS NETWORK, "SPREADING FACTORS", 2024, [ACEDIDO: 01/08/2024] LINK: https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/

[47] THE THINGS NETWORK, "RSSI AND SNR", 2024, [ACEDIDO: 02/08/2024] LINK: https://www.thethingsnetwork.org/docs/lorawan/rssi-and-snr/

[48] THE THINGS NETWORK, "LORA PHYSICAL LAYER PACKET FORMAT", 2024, [ACEDIDO: 02/08/2024] LINK: https://www.thethingsnetwork.org/docs/lorawan/lora-phy-format/

[49] S. DEVALAL, A. KARTHIKEYAN, "LORA TECHNOLOGY - AN OVERVIEW," 2018 SECOND INTERNATIONAL CONFERENCE ON ELECTRONICS, COMMUNICATION AND AEROSPACE TECHNOLOGY (ICECA), COIMBATORE, INDIA, 2018, PP. 284-290, [ACEDIDO: 05/08/2024] LINK: https://ieeexplore.ieee.org/abstract/document/8474715?casa_token=wXUBkm9WXtUAAAAA:sklG23MiJ6eCqCpWv01IZ_EvFiLC-yxKZHcfX6V8e6uBPJTKMz5svSVC4uXodNbbVP7Cg1IX2cw

[50] LORA ALLIANCE, "WHAT IS LORAWAN?", 2024, [ACEDIDO: 05/08/2024] LINK: https://lora-alliance.org/about-lorawan/

[51] E. PAGLIARI, "WHAT'S THE DIFFERENCE BETWEEN LORA AND LORAWAN?", 2024, [ACEDIDO: 07/08/2024] LINK: https://emanuelepagliari.it/2020/10/13/what-is-lorawan-internet-of-things-protocol/#Whats-the-difference-between-LoRa-and-LoRaWAN

[52] THE THING INDUSTRIES, "THE THINGS STACK", 2024, [ACEDIDO: 30/08/2024] LINK: https://www.thethingsindustries.com/stack/

[53] CHIRPSTACK, "CHIRPSTACK, OPEN-SOURCE LORAWAN® NETWORK SERVER", 2024, [ACEDIDO: 30/08/2024] LINK: https://www.chirpstack.io

[54] THE THINGS NETWORK, "DEVICE CLASSES", 2024, [ACEDIDO: 01/09/2024] LINK: https://www.thethingsnetwork.org/docs/lorawan/classes/

[55] THE THINGS NETWORK, "ADDRESSING & ACTIVATION", 2024, [ACEDIDO: 01/09/2024] LINK: https://www.thethingsnetwork.org/docs/lorawan/addressing/

[56] THE THINGS NETWORK, "END DEVICE ACTIVATION", 2024, [ACEDIDO: 01/09/2024] LINK: https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/

[57] D. AVANCINI, J. RODRIGUES, S. MARTINS, R. RABÊLO, J. AL-MUHTADI, P. SOLIC, "ENERGY METERS EVOLUTION IN SMART GRIDS: A REVIEW", 2019 JOURNAL OF CLEANER PRODUCTION VOLUME 217, 20 APRIL 2019, PAGES 702-715, [ACEDIDO: 05/09/2024] LINK: https://www.sciencedirect.com/science/article/pii/S0959652619302501?via%3Dihub#bib67

[58] M. ZHUANG, M. SHAHIDEHPOUR, Z. LI, "AN OVERVIEW OF NON-INTRUSIVE LOAD MONITORING: APPROACHES, BUSINESS APPLICATIONS, AND CHALLENGES," 2018 INTERNATIONAL CONFERENCE ON POWER SYSTEM TECHNOLOGY (POWERCON), GUANGZHOU, CHINA, 2018, PP. 4291-4299, [ACEDIDO: 14/09/2024] LINK: https://ieeexplore.ieee.org/document/8601534

[59] FARNELL, "FLUKE 325 CLAMP METER", 2024, [ACEDIDO: 16/09/2024] LINK: https://pt.farnell.com/en-PT/chauvin-arnoux/p01120944/clamp-meter-true-rms-1-7kv-1-5ka/dp/4216124

[60] S. ARAR, "HALL EFFECT CURRENT SENSING: OPEN-LOOP AND CLOSED-LOOP CONFIGURATIONS", 2021, [ACEDIDO: 16/09/2024] LINK: https://www.allaboutcircuits.com/technical-articles/hall-effect-current-sensing-open-loop-and-closed-loop-configurations/

[61] S. KRISHNA, "INTRODUCTION TO DATABASE AND KNOWLEDGE-BASE SYSTEMS", WORLD SCIENTIFIC SERIES IN COMPUTER SCIENCE, INDIA, VOL.28, [ACEDIDO: 24/09/2024] LINK: https://books.google.pt/books?hl=ptPT&lr=&id=HPvfYeFaA3oC&oi=fnd&pg=PA1&dq=introduction+to+database&ots=hPzly2lsK4&sig=AhhfVk7RO9LVrS_JWhKqELuXe_w&redir_esc=y#v=onepage&q=introduction%20to%20database&f=false

[62] N. JATANA, S. PURI, M. AHUJA, I. KATHURIA, D. GOSAIN, "A SURVEY AND COMPARISON OF RELATIONAL AND NON-RELATIONAL DATABASE", 2012 INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT), VOL.1, ISSUE 6, [ACEDIDO: 01/10/2024] LINK: https://d1wqtxts1xzle7.cloudfront.net/76957411/a-survey-and-comparison-of-relational-and-non-relational-database-libre.pdf?1640094061=&response-content-disposition=inline%3B+filename%3DA_Survey_and_Comparison_of_Relational_an.pdf&Expires=1733396290&Signature=XEjorXCQ9sgVSNmFshnZzCPPpPt8vVF1wQdvQbAG0oZpiCGSptAS7JSQ9~tqyVK9ALiLlmwImDa0saTJNPLZ0mUVXNzR4cUVF9V8rOLaxp6163j4r6dC0kO1W~YoSW2kPozkCDVT~tcq1nBur2OCFq4kGVC~S5IM2Fe1TDtlkbCRKTJDhofCuU9GB1nhHwXFydzfFYQIkx-rOEj~SNaA4LWv63ZEYNiq3x4JBo50DZiKUzGw8FkZRlrip-clWz-9UnV-x8Gq~FsOzVNweCTs4IrNa~c1tMW8k67uLW9wELTg1uU4b0jsvuAc6dx5Vs2LXxsKho4QiSrf-gGNJwGqgg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

[63] BASE DE DADOS MYSQL, [ACEDIDO: 02/10/2024] LINK: https://www.mysql.com

[64] BASE DE DADOS ORACLE, [ACEDIDO: 02/10/2024] LINK: https://www.oracle.com/database/

[65] BASE DE DADOS POSTGRESSQL, [ACEDIDO: 02/10/2024] LINK: https://www.postgresql.org

[66] BASE DE DADOS SQLITE, [ACEDIDO: 02/10/2024] LINK: https://www.sqlite.org

[67] BASE DE DADOS MARIADB, [ACEDIDO: 02/10/2024] LINK: https://mariadb.org

[68] BASE DE DADOS MONGODB, [ACEDIDO: 02/10/2024] LINK: https://www.mongodb.com/pt-br

[69] BASE DE DADOS CASSANDRA, [ACEDIDO: 02/10/2024] LINK: https://cassandra.apache.org/_/index.html

[70] BASE DE DADOS FIREBASE: https://firebase.google.com/?hl=pt-br

[71] W. KHAN, T. KUMAR, C. ZHANG, K. RAJ; A.M. ROY, B. LUO, "SQL AND NOSQL DATABASE SOFTWARE ARCHITECTURE PERFORMANCE ANALYSIS AND ASSESSMENTS—A SYSTEMATIC LITERATURE REVIEW. BIG DATA COGN. COMPUT." 2023, 7, 97, [ACEDIDO: 02/10/2024] LINK: https://www.mdpi.com/2504-2289/7/2/97

[72] IBM, "WHAT IS SOAP?, 2024, [ACEDIDO: 03/10/2024] LINK: https://www.ibm.com/docs/en/integration-bus/10.0?topic=services-what-is-soap

[73] RED HAT, "WHAT IS A REST API?", 2024, [ACEDIDO: 08/10/2024] LINK: https://www.redhat.com/en/topics/api/what-is-a-rest-api

[74] GRAPHQL, "A QUERY LANGUAGE FOR YOUR API", 2024, [ACEDIDO: 09/07/2024] LINK: https://graphql.org

[75] I. YU, "GUIDE TO APIS: HTTP & 3 API PROTOCOLS YOU NEED TO KNOW", 2023, [ACEDIDO: 10/10/2024] LINK: https://www.skiplevel.co/blog/guide-to-api-part-1

[76] X. HUANG, "RESEARCH AND APPLICATION OF NODE.JS CORE TECHNOLOGY," 2020 INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING AND HUMAN-COMPUTER INTERACTION (ICHCI), SANYA, CHINA, 2020, PP. 1-4, [ACEDIDO: 11/10/2024] LINK: https://ieeexplore.ieee.org/document/9424850

[77] TECNOLOGIA PHP, [ACEDIDO: 12/10/2024] LINK: https://www.php.net

[78] TECNOLOGIA NODEJS, [ACEDIDO: 12/10/2024] LINK: https://nodejs.org/en

[79] TECNOLOGIA PYTHON, [ACEDIDO: 12/10/2024] LINK: https://www.python.org

[80] TECNOLOGIA JAVA, [ACEDIDO: 12/10/2024] LINK: https://www.java.com/en-US/

[81] TECNOLOGIA C#, [ACEDIDO: 12/10/2024] LINK: https://learn.microsoft.com/en-us/dotnet/csharp/

[82] BIBLIOTECA EXPRESSJS, [ACEDIDO: 13/10/2024] LINK: https://expressjs.com

[83] FRAMEWORK DJANGO, [ACEDIDO: 13/10/2024] LINK: https://www.djangoproject.com

[84] FRAMEWORK FLASK, [ACEDIDO: 15/10/2024] LINK: https://flask.palletsprojects.com/en/stable/

[85] FRAMEWORK SPRING-BOOT, [ACEDIDO: 16/10/2024] LINK: https://spring.io/projects/spring-boot

[86] TECNOLOGIA .NET, [ACEDIDO: 19/10/2024] LINK: https://dotnet.microsoft.com/en-us/download

[87] G. KAUR, R. G. TIWARI, "COMPARISON AND ANALYSIS OF POPULAR FRONTEND FRAMEWORKS AND LIBRARIES: AN EVALUATION OF PARAMETERS FOR FRONTEND WEB DEVELOPMENT," 2023 4TH INTERNATIONAL CONFERENCE ON ELECTRONICS AND SUSTAINABLE COMMUNICATION SYSTEMS (ICESC), COIMBATORE, INDIA, 2023, PP. 1067-1073, [ACEDIDO: 22/10/2024] LINK: https://ieeexplore.ieee.org/document/10192987

[88] FRAMEWORK REACT, [ACEDIDO: 25/10/2024] LINK: https://react.dev

[89] FRAMEWORK ANGULAR, [ACEDIDO: 26/10/2024] LINK: https://angular.dev

[90] FRAMEWORK VUEJS, [ACEDIDO: 26/10/2024] LINK: https://vuejs.org

[91] FRAMEWORK BOOTSTRAP, [ACEDIDO: 26/10/2024] LINK: https://getbootstrap.com

[92] BIBLIOTECA NIVO, [ACEDIDO: 26/10/2024] LINK: https://nivo.rocks/

[93] BIBLIOTECA CHARTJS, [ACEDIDO: 26/10/2024] LINK: https://www.chartjs.org

[94] BIBLIOTECA MUI, [ACEDIDO: 26/10/2024] LINK: https://mui.com

[95] G. DALPIAZ, A. LONGO, M. NARDELLO, R. PASSERONE, D. BRUNELLO, "A BATTERY-FREE NON-INTRUSIVE POWER METER FOR LOW-COST ENERGY MONITORING," 2018 IEEE INDUSTRIAL CYBER-PHYSICAL SYSTEMS (ICPS), ST. PETERSBURG, RUSSIA, 2018, PP. 653-658, [ACEDIDO: 26/10/2024] LINK: https://ieeexplore.ieee.org/abstract/document/8390784

[96] F. SÁNCHEZ-SUTIL, A. CANO-ORTEGA, J.C HERNÁNDEZ, "DESIGN AND IMPLEMENTATION OF A SMART ENERGY METER USING A LORA NETWORK IN REAL TIME. ELECTRONICS", 2021, 10, 3152, [ACEDIDO: 28/10/2024] LINK: https://www.mdpi.com/2079-9292/10/24/3152

[97] TECNOLIGA NODERED, [ACEDIDO: 28/10/2024] LINK: https://nodered.org

[98] SERVIÇO DE INTEGRAÇÃO IFTTT, [ACEDIDO: 28/10/2024] LINK: https://ifttt.com

[99] MÓDULO DRAGINO LORA BEE 868MHZ, DRAGINO, [ACEDIDO: 28/10/2024] LINK: https://www.botnroll.com/pt/lora-868/4333-dragino-lora-bee-868mhz.html

[100] IBM, "MASTER/SLAVE MODEL", 2024, [ACEDIDO: 29/10/2024] LINK: https://www.ibm.com/docs/en/aix/7.1?topic=models-masterslave-model

[101] R. SHEEBA, "REAL-TIME MONITORING OF ENERGY METERS USING CLOUD STORAGE," 2021 IEEE INTERNATIONAL POWER AND RENEWABLE ENERGY CONFERENCE (IPRECON), KOLLAM, INDIA, 2021, PP. 1-5, [ACEDIDO: 01/11/2024] LINK: https://ieeexplore.ieee.org/document/9640636

[102] GATEWAY LORAWAN LG308, DRAGINO, [ACEDIDO: 01/11/2024] LINK: https://www.dragino.com/products/lora-lorawan-gateway/item/140-lg308.html

[103] A. JALEEL, S. AZIZ, H. P. R, S. MARY, K. VENUSAMY, "IOT BASED SMART ENERGY CONSUMPTION AND MONITORING SYSTEM," 2023 3RD INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND SOCIAL NETWORKING (ICPCSN), SALEM, INDIA, 2023, PP. 956-961, [ACEDIDO: 05/11/2024] LINK: https://ieeexplore.ieee.org/abstract/document/10266183?casa_token=s19bOK_ZGxgAAAAA:zA_MilT2ITJVdXfxldJXxj0xHVZeb0VnCOfBQJ0UqR6YARFJvznXD4zXWdDgmMXcLxYkAml3TL8

[104] GATEWAY WIRNET STATION, KERLINK, [ACEDIDO: 06/11/2024] LINK: https://www.kerlink.com/product/wirnet-istation/

[105] MÓDULO GROVE AC VOLTAGE SENSOR, SEEEDSTUDIO, [ACEDIDO: 06/11/2024] LINK: https://www.seeedstudio.com/Grove-AC-Voltage-sensor-p-5540.html

[106] MICROPROCESSADOR MCP6002, MICROCHIP, [ACEDIDO: 06/11/2024] LINK: https://www.microchip.com/en-us/product/MCP6002

[107] SENSOR DE CORRENTE NÃO-INVASIVO SENSOR-SCT-013-20A-MAX, YHDC, [ACEDIDO: 06/11/2024] LINK: https://www.botnroll.com/en/current/3986-non-invasive-ac-current-sensor-sct-013-20a-max.html

[108] MEDIDOR PZEM-004T-100A, PEACEFAIR, [ACEDIDO: 06/11/2024] LINK: https://mauser.pt/catalog/product_info.php?products_id=096-8731

[109] MÓDULO LORA-E5 MINI, SEMTECH, [ACEDIDO: 07/11/2024] LINK: https://mauser.pt/catalog/product_info.php?products_id=096-9675

[110] MICROPROCESSADOR STM32WLE5JC, [ACEDIDO: 07/11/2024] LINK: https://www.st.com/en/microcontrollers-microprocessors/stm32wle5jc.html

[111] MÓDULO GROVE LORA-E5, SEMTECH, [ACEDIDO: 07/11/2024] LINK: https://mauser.pt/catalog/product_info.php?products_id=096-9677

[112] MÓDULO LORA-SX1278, SEMTECH, [ACEDIDO: 07/11/2024] LINK: https://solectroshop.com/pt/modulos-wi-fi/5456-development-board-ra-02-lora-sx1278-ra-02-module-433mhz-long-range-5905323235656.html

[113] MÓDULO LORA SX1278, SEMTECH, [ACEDIDO: 09/11/2024] LINK: https://www.semtech.com/products/wireless-rf/lora-connect/sx1278

[114] BIBLIOTECA SOFTWARE-SERIAL, ARDUINO, [ACEDIDO: 09/11/2024] LINK: https://docs.arduino.cc/learn/built-in-libraries/software-serial/

[115] BIBLIOTECA PZEM004TV30, ARDUINO, [ACEDIDO: 11/11/2024] LINK: https://docs.arduino.cc/libraries/pzem004tv30/

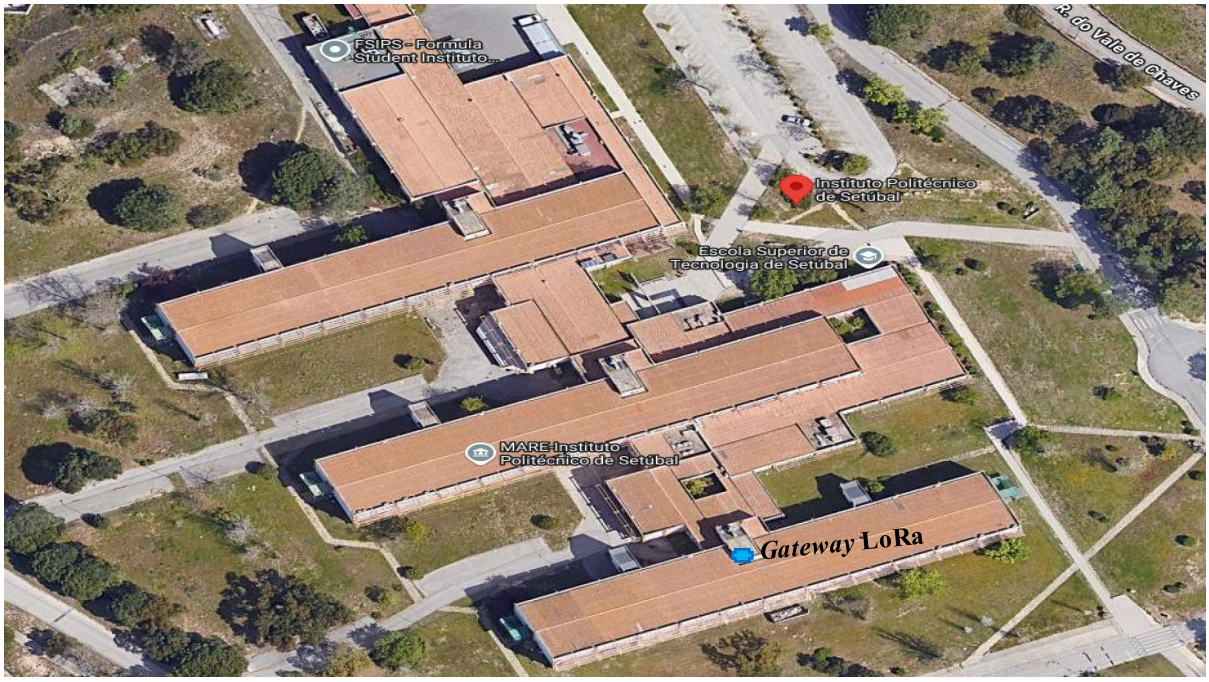
[116] LISTA DE COMANDOS LORA-E5 AT, SEEEDSTUDIO, [ACEDIDO: 11/11/2024] LINK: https://files.seeedstudio.com/products/317990687/res/LoRa-E5+AT+Command+Specification_V1.0+.pdf

[117] DOCUMENTAÇÃO DO MEDIDOR PZEM-004T-100A , [ACEDIDO: 11/11/2024] LINK: https://storage.googleapis.com/mauser-public-images/prod_description_document/2022/199/05f03b5b862b203ff7c40db03f08ed7d_096-8731.pdf

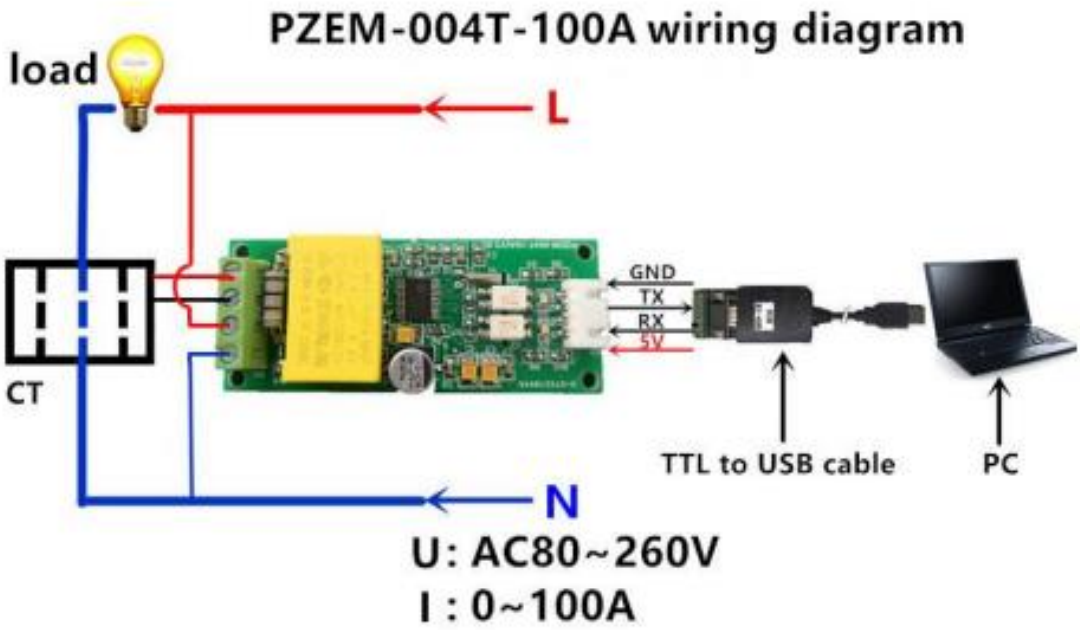
Anexo

Este anexo é composto em um total de três partes. No anexo 1 encontra-se o ponto onde o *gateway LoRa* foi instalado e no anexo 2 está ilustrado o esquema de ligações do medidor PZEM-004t-100A. Por último, no anexo 3, encontra-se o código para a programação do sensor *LoRa* desenvolvido neste trabalho.

Anexo 1: Ponto de instalação do gateway LoRa - visão no mapa da ESTSetúbal/IPS, com a marcação do ponto onde o *gateway* LoRa foi instalado. Este encontra-se no telhado da escola, acima do bloco F.



Anexo 2: Esquema de ligação do PZEM-004t-100A.



Picture 4.2 PZEM-004T-100A wiring diagram

Anexo 3: Código para o sensor desenvolvido para a leitura de consumos energéticos.

```
#include <SoftwareSerial.h>
#include <PZEM004Tv30.h>
#include <LowPower.h>
#define APPKEY "C53F64126BF6C5976155076CEF798775"
#define DATA_RATE "0"
#define AUTH_METHOD "LWOTAA"
#define PORT "1"

// LoRa Pin 10 RX e Pin 11 TX
SoftwareSerial Serial_Lora(10, 11);
// PZEM Pin 5 RX e Pin 6 TX
PZEM004Tv30 pzem(5, 6);

/**
* Configurações iniciais
* Serial - PZEM, SoftwareSerial - LoRa
*/
void setup() {
  Serial.println("CONFIGURATION: Start.");
  Serial.println("CONFIGURATION: #### PZEM Module ####");
  if(pzem.readAddress() != NAN) {
    Serial.println("CONFIGURATION: PZEM Module Ready.");
  } else {
    Serial.println("CONFIGURATION: PZEM Module ERROR: No Available Address");
  }
  Serial.begin(9600);
  Serial.println("CONFIGURATION: #### LoRa-E5 Module ####");
  Serial_Lora.begin(9600);
  while(!setupLoraDevice());
  Serial.println("CONFIGURATION: LoRa-E5 Module Ready.");
  Serial.println("CONFIGURATION: Done.\n");
  pzem.resetEnergy();
}
```

```
void loop() {
  readPowerMetrics();
  LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF);
}

/**
* Esta função vai enviar comandos AT para o módulo LoRa
* @request - este é o comando AT que o módulo de transmissão vai executar
* @expectedResponse - é a resposta expectável do módulo depois de executar o comando AT
*/
void sendATCommand(String request, String expectedResponse) {
  Serial_Lora.println(request);

  while (Serial_Lora.available())  {
    String response = Serial_Lora.readStringUntil('\n');
    Serial.println("Serial_Lora:  " + response);

    if (response.indexOf(expectedResponse) != -1) {
      break;
    }
  }
}

/**
* Esta função vai configurar o módulo LoRa-E5, garantido que fique operacional para comunicar com o servidor TTN
* É feito um reset inicial para garantir que todas as definições sejam feitas corretamente, após isso é configurado
* o modo de autenticação com servidor TTN para LWOTAA, a appkey da aplicação TTN e feito o pedido de join para a mesma.
* Se o pedido de join falhar, o led vermelho é ligado
*/
boolean setupLoraDevice() {
```

```
    sendATCommand("AT+RESET", "OK");
    delay(3000); // Garantir 3s para que todas as configurações sejam limpas
    sendATCommand("AT+MODE=" + String(AUTH_METHOD), "OK");
    delay(1000); // Garantir 1s para que a alteração seja aplicada
    sendATCommand("AT+KEY=APPKEY,\"" + String(APPKEY) + "\"", "OK");
    delay(1000);
    sendATCommand("AT+PORT=\"" + String(PORT) + "\"", "OK"); // Configurar Data Rate para SF9 BW125kHz
    delay(1000);
    sendATCommand("AT+ADR=OFF", "OK"); // Configurar Data Rate para SF9 BW125kHz
    delay(1000);
    sendATCommand("AT+DR=" + String(DATA_RATE), "OK"); // Configurar Data Rate para SF9 BW125kHz
    delay(1000);
    sendATCommand("AT+ID", "OK");
    delay(1000);
    sendATCommand("AT+JOIN", "OK");

    if(isDeviceConnectedToTTN()) {
      Serial.println("CONFIGURATION: Device joined TTN");
      return true;
    } else {
      Serial.println("CONFIGURATION: Failed to join TTN");
      return false;
    }
  }

  /**
  * Esta função vai retornar um boolean a indicar se o sensor está conectado ao servidor TTN
  * Vai esperar 30s para receber uma mensagem de resposta do servidor +EVT:JOINED, se receber antes da janela de tempo terminar
  * então retorna true
  * Caso contrário retorna false
```

```
*/
bool isDeviceConnectedToTTN() {
  unsigned long startTime = millis();
  String response = "";

  while (millis() - startTime < 30000) {
    if (Serial_Lora.available()) {
      String response = Serial_Lora.readStringUntil('\n');

      if (response.indexOf("+JOIN: Network joined") != -1) {
        return true;
      }
    }
  }
  return false;
}

/**
* Esta função vai enviar comandos para o medidor PZEM para ler cada uma das métricas
* Se o valor obtido do sensor para uma das métricas for "NAN" então a variável dessa métrica é inicializada com o valor
* de omissão "0"
* Ao fim de obter todas as métricas, vai ser construido um payload em formato JSON para ser enviado para a aplicação TTN
*/
void readPowerMetrics() {
  float voltage = pzem.voltage();
  if (!isnan(voltage)) {
    Serial.println(voltage);
    Serial.println("Tensão:  " + String(voltage) + "V");
  } else {
    voltage = 0;
    Serial.println("Error  reading voltage");
  }
```

```
float current = pzem.current();
if (!isnan(current)) {
  Serial.println("Corrente: " + String(current) + "A");
} else {
  current = 0;
  Serial.println("Error reading current");
}

float power = pzem.power();
if (!isnan(power)) {
  Serial.println("Potência: " + String(power) + "W");
} else {
  power = 0;
  Serial.println("Error reading power");
}

float energy = pzem.energy();
if (!isnan(energy)) {
  Serial.println("Energia: " + String(energy) + "kWh");
} else {
  energy = 0;
  Serial.println("Error reading energy");
}

float frequency = pzem.frequency();
if (!isnan(frequency)) {
  Serial.println("Frequencia: " + String(frequency, 1) + "Hz");
} else {
  frequency = 0;
  Serial.println("Error reading frequency");
}

float pf = pzem.pf();
if (!isnan(pf)) {
```

```
    Serial.println("Fator  De Potência: " + String(pf, 1));
   } else {
    pf = 0;
    Serial.println("Error  reading power factor");
   }

   float power_apparent = voltage * current;
   float power_reactive = power_apparent * sin(acos(pf));

   String measurement  = String(voltage) + "," + String(current) + "," + String(power) + "," + String(power_reactive) + "," + String(power_apparent) + "," + String(energy) + "," + String(frequency) + "," + String(pf);
   Serial.println(measurement);
   sendATCommand("AT+CMSG=\"" + measurement + "\"", "OK");
  }
```