Smart-City Drainage System: Sistema IoT de Monitorização de Infraestruturas de Drenagem de Águas Pluviais

Paulo Carvalho

Faculdade de Ciência e Tecnologia

Universidade Fernando Pessoa

Uma tese submetida para obtenção do grau de

Mestre

2022

Resumo

Os sistemas urbanos de drenagem de águas pluviais têm como função permitir o escoamento das águas pluviais, através de caixas de recolha, tipicamente instaladas nas bermas das vias de comunicação. Estas caixas recebem ao longo do ano, não só as águas provenientes das chuvas, mas também detritos que se vão acumulando e que, por vezes, entopem os sistemas de drenagem. A previsão de inundações, resultantes destes entupimentos, pode evitar danos materiais graves e consequentemente evitar despesas avultadas associados à limpeza e reparação de vias e edifícios, públicos e privados.

Neste contexto, o trabalho desenvolvido nesta dissertação foca-se na especificação, desenvolvimento e avaliação de um sistema IoT, usando redes de sensores, que permite a monitorização em larga escala das caixas de recolha de águas pluviais de uma cidade. Estas caixas de recolha são elementos cruciais dos sistemas de drenagem e o objetivo é antecipar a previsão do entupimento destas caixas.

O sistema IoT proposto utiliza uma combinação de arquitecturas IoT (cf. Edge, Fog e Cloud) adequada à monitorização escalável e autónoma de redes municipais de caixas de drenagem. Os nós sensoriais combinam a utilização de vários sensores de modo a permitir efetuar uma monitorização em larga escala de todo o sistema de drenagem, para informar as autoridades autárquicas sobre o estado atual das caixas. O sistema de monitorização permite também detetar possíveis fatores de entupimento e consequentemente gerar alertas de manutenção para prevenir eventuais cheias e todos os prejuízos e incómodos daí decorrentes.

Abstract

The function of urban stormwater drainage systems is to allow rainwater to run off through collection boxes, typically installed on the sides of roads. These boxes receive throughout the year, not only the water from the rains, but also debris that accumulates and sometimes clogs the drainage systems. The forecasting of floods, resulting from these blockages, can prevent serious material damage and consequently avoid huge expenses associated with the cleaning and repair of roads and buildings, both public and private.

In this context, the work developed in this thesis focuses on the specification, development and evaluation of an IoT system, using sensor networks, that allows large-scale monitoring of a city's stormwater collection boxes. These collection boxes are crucial elements of drainage systems and the goal is to be able to anticipate the prediction of clogging of these boxes.

The proposed IoT system uses a combination of IoT architectures (cf. Edge, Fog and Cloud) suitable for scalable and autonomous monitoring of municipal networks of drainage boxes. The sensory motes combine the use of various sensors to enable large-scale monitoring of the entire drainage system to inform municipal authorities about the current status of the boxes. The monitoring system also makes it possible to detect possible clogging factors and consequently generate maintenance alerts to prevent possible flooding and all the resulting damage and inconvenience.

I would like to dedicate this thesis to my loving wife Bárbara ...

Agradecimentos

Esta dissertação representa a concretização de mais um objetivo pessoal concluído. Estes cinco anos permitiram aumentar o conhecimento em diversas áreas, através de muitos professores e colegas, o que permitiu uma evolução pessoal em todos os aspetos, tanto a nível pessoal como profissional.

Um grande agradecimento aos professores e orientadores Dr. Rui Moreira e Dr. Christophe Soares, que mostraram sempre grande disponibilidade para ajudar em diversos aspetos desta dissertação, e acompanharam em todos os passos o desenvolvimento da mesma. Também agradecer a todos os outros professores do curso de Engenharia Informática, que nestes cinco anos contribuiriam para o aumento de conhecimento pessoal e profissional nesta área.

Um obrigado muito especial para a minha esposa, Bárbara Lopes, que me incentivou sempre a prosseguir com os meus sonhos, ajudou, esteve lá nos momentos bons e maus e nunca me fez desistir dos meus objetivos. Também aos pais, irmão, sogros e a restante família, que sempre me motivaram e incentivaram nestes cinco anos de aprendizagem.

Um grande agradecimento a todos os colegas e amigos que tive oportunidade de conhecer nesta vida acadêmcia, e que me acompanharam e ajudaram durante estes cincos anos.

Um grande agradecimento também para a Hardlevel, empresa que deu a possibilidade de evoluir no mundo do mercado de trabalho, em especial ao Dr. Karim Karmali e Dr. Salim Karmali, e aos meus colegas que conheci nesta empresa.

Por fim, à instituição Universidade Fernando Pessoa, que, para além de proporcionar a oportunidade de estudar no curso que amo, incutiu-me os melhores valores e ensinamentos para levar por esta vida fora.

Lista de Siglas

AWS *Amazon Web Services*

CoAP *Constrained Application Protocol*

GPRS *General Packet Radio Service*

GSM *Global System for Mobile Communications*

HTTP *Hypertext Transfer Protocol*

IoT *Internet of Things*

MQTT *Message Queuing Telemetry Transport*

NB-IoT *Narrowband Internet of things*

TCP *Transmission Control Protocol*

LTE *Long Term Evolution*

Contents

List of Figures

List of Tables

Chapter 1

Introdução

Com a chegada do inverno vêm as tempestades e consequentemente as cheias, que provocam normalmente danos materiais nas vias, edifícios e outros bens, bem como obrigam a processos de limpeza dispendiosos.

Espalhadas pela via pública existem caixas de recolha que escoam as águas pluviais por canais subterrâneos em direção aos rios e ao mar. Estes sistemas tiram partido da gravidade para escoar as águas das chuvas. As caixas de recolha de águas pluviais designam-se vulgarmente por bueiros ou caixas coletoras. Os bueiros são caixas mais pequenas, localizadas nas bermas das vias, que se organizam em grupos para receber as águas das chuvas. Existem ainda as caixas coletoras, também localizadas nas bermas das vias, que recebem as ligações dos grupos de bueiros.

A chegada do outono está associada à queda de folhas que enchem muitas das vias públicas. Adicionalmente, os sedimentos e lixo espalhados pela rua são arrastados, com a chegada das primeiras chuvas, para os bueiros, provocando assim possíveis entupimentos nos bueiros e caixas coletoras. Normalmente associados aos entupimentos surgem as cheias, pela falta do escoamento correto da água.

Neste contexto e devido a estes problemas procurou-se neste trabalho estudar e desenvolver um sistema escalável de monitorização que permita informar a autarquia sobre o estado atual das caixas de recolha de águas pluviais e dessa forma antever e prevenir cheias nas vias públicas. Pretende-se, portanto, recolher informação que permita à autarquia conhecer o nível de lixo que existe nas caixas de recolha de modo a desencadear se necessário a sua limpeza e manutenção. Conseguindo prevenir e evitar o entupimento destas caixas, consegue-se também evitar danos materiais e custos adicionais associados à reparação e limpeza de vias e edifícios, resultantes de possíveis cheias.

1.1 Motivação do Trabalho

Este tema despertou interesse devido ao facto de existirem recorrentemente várias notícias nos media sobre cheias, com particular relevo para determinadas zonas das regiões do Porto, Lisboa e Algarve. Muitas destas cheias estão normalmente associadas a uma má gestão da limpeza e manutenção das caixas coletoras. Seria, portanto, importante conseguir ajudar as autarquias a fazer uma verificação constante do estado dos seus sistemas de escoamento e, consequentemente, permitir um melhor planeamento e gestão da sua limpeza e funcionamento. Num esforço inicial para perceber melhor este tema, percebemos que existem já armadilhas que foram desenvolvidas no Brasil para filtrar o lixo à entrada das caixas coletoras. No entanto, estes filtros não possuem inteligência ao ponto de reconhecerem automaticamente se as caixas estão cheias de lixo ou não. Adicionalmente, esta solução contribui para acumular lixo à entrada das caixas, dificultando ainda mais a recolha e escoamento de águas, agravando a possibilidade de cheias.

Este tema é também muito importante no âmbito da empresa onde o autor da dissertação estava integrado e pela qual foi apoiado. A empresa em questão aposta em várias soluções para smart cities, sendo, portanto, outro fator importante para a escolha do tema.

1.2 Problema

As cheias provocadas pelo entupimento dos sistemas de drenagem urbanos são um problema generalizado em todo o mundo, particularmente agravado pelas alterações climáticas. Uma pesquisa bibliográfica preliminar permitiu identificar algumas soluções de monitorização e previsão de cheias, que foram criadas noutros países. Todas estas soluções foram concebidas com o intuito de funcionarem em tempo real na deteção de cheias. Contudo, diferentes países adotam soluções específicas normalmente ajustadas à existência de diferentes arquiteturas nos seus sistemas de drenagem de águas pluviais. Cada país possui tipicamente estruturas próprias e específicas nas suas caixas de recolha. Em Portugal, por exemplo, a arquitetura dos sistemas de drenagem é específica e o sistema tem que se adequar à realidade existente. No momento em que desenvolvemos este trabalho, pelo que conseguimos determinar, ainda não existe nenhuma solução tecnológica em Portugal para a resolução deste problema concreto.

1.3 Objetivos

Pretende-se com este sistema IoT monitorizar e alertar a autarquia para possíveis entupimentos das caixas de drenagem, gerando alertas de deteção de cheias e alertas do nível de lixo presente nestas caixas para facilitar a sua manutenção.

O sistema IoT deverá constituir uma solução escalável e de baixo custo com base numa rede de sensores para permitir a monitorização de caixas de drenagem e alertar para eventuais cheias ou para a presença de lixos nestas caixas de drenagem.

1.4 Restrições

Este é um projeto complexo no qual se previram várias restrições. Por exemplo, uma restrição importante é o facto de todo o sistema de sensores ter de ser IP67 ou IP68 pois, caso aconteçam cheias, os vários componentes não podem ser danificados e terão de ser resistentes para suportar condições adversas que se encontram dentro das caixas de drenagem. Estas restrições aplicam-se tanto aos sensores como às caixas e componentes de hardware do sistema, incluindo microcontroladores, dispositivos de comunicação e baterias, uma vez que estas caixas serão instaladas dentro ou nas proximidades das caixas de drenagem.

Existem também restrições quanto ao tipo de comunicação e alimentação dos sistemas, uma vez que os nós de monitorização vão ser instalados no subsolo. Pelo facto de ser um sistema que não irá ter energia elétrica constante, irá existir a necessidade de um estudo aprofundado sobre a sua autonomia.

Outra restrição importante está relacionada com o facto do sistema ter de ser económico e de fácil instalação, para poder abranger áreas urbanas extensas. Portanto, será necessário que os sensores utilizados sejam de baixo custo e que os nós sejam de fácil instalação nas caixas de drenagem.

1.5 Metodologia

O desenvolvimento do trabalho proposto foi subdividido e realizado em várias fases, abaixo descritas.

- Identificação do problema: Inicialmente é importante identificar e caracterizar os problemas e os fatores que levam ao aparecimento das cheias e que estão relacionados com as caixas de drenagem.

- Revisão Bibliográfica de Soluções Existentes e Ferramentas Úteis: Pesquisar artigos científicos sobre projetos de investigação e também sistemas ou produtos que já se encontrem no mercado com possíveis soluções para o problema. Identificar vantagens e desvantagens das diversas soluções de modo a planear uma abordagem que preencha as necessidades e se adapte ao caso português.

- Conceção e desenvolvimento do sistema: Desenvolver um protótipo do sistema IoT de monitorização da drenagem de águas pluviais. Para tal deverá pesquisar-se e

identificar-se diversos sensores que possam ser integrados na solução proposta e que se coadunem com as condições difíceis que se verificam nas caixas de drenagem, isto é, níveis de lixo presentes, condições de humidade e espaço disponível em cada bueiro (devido a existirem bueiros de diversos tamanhos). A autarquia deverá ter acesso a um dashboard onde visualizará um mapa e/ou uma tabela com o estado de todos os bueiros monitorizados pelo sistema. Serão gerados alertas caso algum bueiro tenha possíveis problemas de entupimento ou má drenagem, de modo a planear a manutenção e evitamento de cheias. Pretende-se também estudar possíveis locais onde se poderão colocar os sensores e a caixa do mote que conterá os microcontroladores e baterias. A decisão do tipo de comunicações a utilizar e como este sistema será alimentado são também questões importantes para o correto planeamento, desenvolvimento e funcionamento do sistema. Pretende-se também realizar diversos testes aos sensores que possam vir a ser usados, bem como aos restantes componentes do sistema. Os testes serão realizados recorrendo a protótipos construídos com o propósito de conseguirmos manter ambientes controlados na caixa de drenagem.

- Realização de um estudo empírico: Depois da montagem de todo o sistema planeiam-se testes aos sensores, comunicações e alimentação do sistema. Prevê-se a realização destes testes tanto em laboratório como em ambientes reais, ou seja, com os nós instalados em condições semelhantes às reais para aferir a resistência e viabilidade dos componentes, bem como da eficiência das comunicações entre os nós do sistema.

1.6 Recursos

Para a realização deste trabalho, em particular do protótipo do sistema IoT de monitorização de águas pluviais, utilizaram-se város componentes de hardware que incluiram:

- Computador pessoal;
- Micro-controladores (cf. Arduíno, Esp32, etc.);
- Gateway (e.g. Raspberry Pi);
- Sensores (e.g. ultrassom, fotoelétrico, etc.);
- Placas de comunicação (e.g. SIM800L);
- Baterias;
- Placas PCB para montar e integrar os componentes de hardware;

- Caixas IP68 para proteção e isolamento do hardware;
- Caixas e tubos para montagem dos protótipos;
- Ferramentas de desenvolvimento (e.g. C/C++, *React*, *Python*);
- Serviços integradores de *Cloud* (e.g. *ThingSpeak*, AWS).

1.7 Estrutura da Dissertação

Esta dissertação de mestrado está organizada em cinco capítulos principais, conforme se descreve abaixo.

No estado de arte, abordam-se alguns temas e conceitos importantes para conhecer o que são *smart cities*, Internet das coisas, tecnologias de comunicação, tecnologias de coordenação e sensores. Exploram-se ainda alguns trabalhos relacionados, desenvolvidos na área de monitorização de drenagem de águas pluviais.

Na especificação do sistema, começa-se por descrever a arquitetura e funcionamento dos sistemas de drenagem de águas pluviais em Portugal, bem como os componentes e processos envolvidos na drenagem. Detalham-se também os requisitos de todo o sistema, bem como também a arquitetura deste. Procura-se justificar também a utilização dos sensores recorrendo a testes e faz-se uma análise de diversas formas de comunicação, para enviar os dados para a *cloud*.

Na implementação do sistema, descreve-se o hardware escolhido e os componentes de comunicação do protótipo. Detalha-se a montagem e implementação de todos os componentes do sistema, nomeadamente os componentes de hardware escolhidos em função de vários testes desenvolvidos. Descreve-se ainda o software desenvolvido para os vários componentes, nomeadamente dos módulos de hardware, do serviço de *cloud* e da aplicação *web*.

Na avaliação do sistema, descrevem-se os testes efetuados e analisam-se resultados de forma a aferir e analisar as diversas funcionalidades, bem como conhecer a autonomia em particular dos componentes de hardware. Procura-se também analisar e justificar de forma sustentada a adequação do sistema à previsão e prevenção de cheias nas vias públicas.

No último capítulo, a conclusão, resume-se todo o trabalho desenvolvido, justificando as fases do desenvolvimento do sistema de drenagem de águas pluviais, procurando enfatizar os principais resultados e contribuições. Procura-se também discutir possíveis objetivos do trabalho futuro.

Chapter 2

Sistemas de Drenagem de Águas Pluviais

Este capítulo apresenta os conceitos tecnológicos de base para a realização desta dissertação, bem como uma revisão de trabalhos relacionados com *Smart-city Drainage Systems*.

2.1 *Smart Cities*

As *smart cities* resultam da aplicação e integração de arquiteturas *Internet of Things* (IoT) nas cidades (Yarlagadda, 2018). A utilização de sistemas de monitorização recorrendo a diversos tipos de sensores, redes sem fios, dispositivos autónomos e aplicações móveis, permitem a recolha de cada vez mais dados sobre as cidades. Estes sistemas oferecem um suporte para o desenvolvimento de novas soluções para problemas de gestão urbana, visando a melhoria da organização e qualidade de vida dos cidadãos.

Os desenvolvimentos de novas tecnologias proporcionam um aumento da interatividade no quotidiano dos cidadãos (Chourabi et al., 2012). As soluções inteligentes otimizam os serviços públicos por meio da gestão analítica das informações sobre onde os recursos estão a ser consumidos. Estes dados possibilitam uma melhor monitorização e gestão por parte do município e permitem que os cidadãos façam uma utilização mais consciente, reduzindo custos operacionais de manutenção e permitindo aumentar o tempo de vida das infraestruturas existentes. Uma *smart city* utiliza a tecnologia e proporciona uma nova forma, mais consciente, de viver na cidade.

2.2 Tecnologias e Sistemas IoT

Atualmente existem diversas definições para o conceito IoT. Neste trabalho consideramos que os sistemas IoT são os que interligam os objetos que estão à nossa volta,

tornando-os mais inteligentes. Referimo-nos a pequenos objetos do nosso dia-a-dia, tais como eletrodomésticos, câmaras, lâmpadas, termostatos, sistemas de som, entre muitos outros, e até mesmo estruturas mais complexas como pontes, estradas e cidades. Os sistemas IoT permitem a conectividade entre todas as coisas que nos rodeiam, tendo em conta, entre outros aspetos, a autonomia dos objetos interligados e a privacidade da informação recolhida (Hassan et al., 2015).

O conceito de IoT foi introduzido em 1999 (Patel et al., 2016), e tem vindo a tornar-se cada vez mais importante devido ao forte crescimento das tecnologias móveis e *Cloud computing*. Existem várias características fundamentais dos sistemas IoT, nomeadamente a interconectividade, heterogeneidade, adaptação dinâmica, escalabilidade, segurança e autonomia dos seus componentes (Patel et al., 2016). As "coisas" que nos rodeiam conseguem comunicar entre si sem precisar da intervenção humana, através da Internet, facilitando a obtenção de dados em tempo real e o seu processamento local ou na *Cloud* (Singh and Singh, 2015).

Os sistemas IoT têm cada vez mais vindo a desempenhar um papel importante na sociedade, em diversos campos tais como a *smart health*, *smart agriculture*, *smart cities*, entre muitos outros (Yarlagadda, 2018). Alguns exemplos de aplicação podem ver-se na monitorização do estado de pacientes, a monitorização dos níveis de lixos nas cidades para uma maior eficiência na recolha destes e a monitorização do solo e recursos hídricos para uma maior eficiência no consumo de água.

Existem inúmeros exemplos de projetos de aplicação IoT. Por exemplo (Nagaraja et al., 2019), propõe um sistema de gestão de agricultura inteligente. Este sistema é composto por dois módulos, um de monitorização de dados e outro de previsão da colheita. Na monitorização de dados é utilizado uma Raspberry Pi 3 e um microcontrolador NodeMCU. Os sensores utilizados são o DHT11, que permite efetuar leituras de temperatura e humidade, e um higrómetro. A previsão da colheita é efectuada com base em dados recolhidos de diferentes origens, incluindo testes laboratoriais e dados sensoriais amostrados nos campos. A previsão é produzida por um modelo de *Machine Learning* (ML) baseado num classificador SVM (*Support Vector Machine*). O protocolo da comunicação entre o microcontrolador e a Raspberry Pi 3 é o *Message Queuing Telemetry Transport* (MQTT). Este sistema também possui uma aplicação *Web* para visualizar os dados das leituras dos sensores.

O projecto (Hori et al., 2010), desenvolvido no Japão, apresenta também um modelo *cloud computing* aplicado na agricultura, mas que pode também ser usado noutros setores, tais como, ambiente, medicina e manutenção. O modelo consiste num *workflow* simples: entrada de dados, armazenamento, visualização, análise e instrução. Este modelo foi testado no terreno para o cultivo de hortaliças e arroz em *Miyazaki* e *Shiba*, respetivamente.

Outro projeto interessante corresponde ao sistema inteligente de gestão de resíduos urbanos proposto em (Kanade et al., 2021). Este sistema utilizou a instalação de sensores

nos caixotes do lixo para a monitorização de resíduos em zonas urbanas. O sistema local é composto por um sensor de ultrassons, um Arduíno com GPS (*Global Positioning System*) e um modem *Global System for Mobile Communications* (GSM), instalados nos caixotes de lixo a monitorizar. Os dados são enviados para a *Cloud* e armazenados numa base dados *Google Firebase*. Os dados de cada caixote podem ser consultados numa aplicação móvel.

Existem também inúmeros sistemas inteligentes de monitorização individual na área da saúde. Por exemplo, o sistema (Ananth et al., 2019) propõe *smartwatches* para ler diferentes parâmetros corporais, tais como *ECG (Electrocardiogram)*, *PPG (Patient Participation Groups)* e *HR (Hear Rate)*. Todos estes parâmetros podem ser utilizados para detetar eventuais problemas cardíacos. Os dados podem ser comunicados via *Bluetooth Low Energy* (BLE), entre o *smartwatch* e o *smartphone*. O *smartphone* fica encarregue de enviar esses dados para o serviço *ThingSpeak*, através de rede celular, onde são processados. Caso seja detetada alguma anomalia, é gerado um alerta por *SMS (Short Message Service)* para o utilizador ou cuidador.

2.2.1 Modelos e Arquitecturas IoT

2.2.1.1 *Cloud Computing*

O termo *cloud computing* é utilizado para caracterizar o armazenamento e processamento de dados que é efectuada em serviços de backend da Internet, ou seja, o termo *cloud* refere-se à Internet e o termo *cloud computing* à computação remota efectuada por serviços disponibilizados na Internet (Soomro and Wahba, 2010). *Cloud computing* refere-se portanto à virtualização de serviços e centros de dados que permitem fornecer aplicações e utilidades remotas (Soomro and Wahba, 2010).

O *cloud computing* move a computação do computador local para a Internet, onde o utilizador não precisa de se preocupar com manutenção e gestão dos recursos. Estes serviços funcionam tipicamente numa perspectiva pagar-para-usar, onde se paga pelos recursos usados da *cloud* (Aazam et al., 2014).

As soluções de *cloud computing* oferecem normalmente quatro categorias de serviços (Aazam et al., 2014):

- *Software as a Service* (SaaS): a aplicação está a correr na Internet, e o utilizador paga pela sua utilização. Tem como vantagem o facto do utilizador não precisar de guardar, instalar e manter a aplicação, sendo tudo gerido pela a empresa que aloja a aplicação.

- *Platform as a Service* (PaaS): a plataforma para a criação de aplicações e serviços fornece todas as ferramentas e recursos que podem ser utilizados com base num custo de utilização.

- *Networks as a Service* (NaaS): são fornecidos recursos ou redes virtuais para os utilizadores aplicarem nos seus serviços.

- *Infrastructure as a Service* (IaaS): são fornece serviços de computação e armazenamento. Tem a vantagem de menor custo, para tarefas de menor custo computacional, uma vez que não são necessárias máquinas com grande poder de processamento. Relativamente aos dados, estes podem estar disponíveis por toda a Internet.

Com o número de número de dispositivos conectados cada vez a aumentar mais, guardar os dados localmente começa a ser cada vez mais insustentável. Também a grande quantidade de dados produzidos não pode ser armazenado nos dispositivos IoT, uma vez que estes são de baixo poder de processamento. Tudo isto é possível com o *cloud computing*, que com a sua integração com os dispositivos IoT, permite que estes dispositivos apenas tenham o trabalho de efetuar as leituras, e depois todo o processamento é feito na *cloud* (Aazam et al., 2014).

O *cloud computing* é um conceito que tem evoluído no tempo, beneficiando das capacidades quase ilimitadas que a *cloud* pode oferecer, tais como, armazenamento e processamento (Bagherzadeh et al., 2020). No entanto, os sistemas IoT não dependem só da *cloud* evoluindo ao longo dos tempos em diversas áreas (Aazam et al., 2014):

- Suporte de protocolos: diversos dispositivos suportam diferentes protocolos de comunicação. Foi necessário um esforço de normalização de protocolos entre os dispositivos IoT e os serviços de *cloud*.

- Eficiência energética: tipicamente um sistema IoT é composto por quatro componentes, sensores/atuadores, processador ou micro-controlador, transmissor e unidade de potência. Não é sustentável que a alimentação dependa de trocas frequentes das baterias. Existe, portanto, a necessidade de conseguir uma boa gestão de energia, recorrendo a energia gerada no ambiente ou colocando os dispositivos em *deepsleep* para aumentar a autonomia.

- Alocação de recursos: um sistema pode recorrer a diferentes tipos de dispositivos, em diversos cenários de aplicação. Assim, a alocação de recursos torna-se um desafio, uma vez que poderá tornar-se complexo prever a quantidade de recursos que cada dispositivo individualmente irá gerar. Deve no entanto realizar-se um estudo de previsão de recursos associados aos cenários de aplicação, de modo a poderem aferir-se os custos necessários.

- Localização do armazenamento de dados: a localização dos dados para o seu armazenamento e processamento também é muito importante, pois queremos que o sistema seja de baixa latência, i.e. minimizar o *delay* na transmissão e receção de dados. Por exemplo, na monitorização em tempo real de séries temporais de dados é importante que a latência seja mínima.

- Segurança e privacidade: quando os dados gerados pelos dispositivos são enviados para a *cloud* é necessário que estejam protegidos. Contudo nem todos os dados recolhidos têm que ir para a *cloud*, podendo ser processados localmente e evitando assim transmitir informação privada ou sensível.

- Comunicação excessiva de dados: quando qualquer dispositivo pode comunicar para a *cloud*, é necessário que todas as comunicações façam sentido e não existam comunicações desnecessárias que gastem recursos indevidamente. Este ponto também é importante na relação com a eficiência energética, uma vez que comunicações exageradas implicam um esgotamento da bateria mais rápido.

2.2.1.2 *Edge Computing*

Segundo *Cao* et al. (Cao et al., 2020), o *edge computing* é diferente do *cloud computing*, que em vez de fazer a sua computação na *cloud*, faz na periferia do sistema, i.e. nos próprios dispositivos de recolha da informação.

Existem diversas definições para *edge computing*. Segundo *Shi and Zhang* (Shi and Zhang, 2019) o *edge computing* é um novo modo de computação cuja execução acontece na periferia da rede. O envio de dados continua a efectuar-se para a *cloud*, contudo na preferia o *edge computing* refere-se ao processamento local de dados nos próprios dispositivos de recolha, entre os dados de origem e o caminho para a *cloud*. Já *Satyanarayanan* (Satyanarayanan, 2017) diz-nos que a aplicação de recursos de computação e armazenamento, tais como pequenas bases de dados e nós locais, na periferia da rede, junto dos dispositivos de recolha de dados, é chamado de *edge computing*.

O *edge computing* transporta parte da computação, armazenamento e recursos para a periferia da rede. Também aplica serviços inteligentes para satisfazer os requisitos de baixa latência e alta largura de banda na rede (Cao et al., 2020).

O *edge computing* apresenta algumas vantagens, entre elas algumas citadas por *Cao* et al. (Cao et al., 2020):

- Processamento de dados e análise em tempo real: o *edge computing* está mais próximo da origem dos dados, permitindo o armazenamento e algumas tarefas de computação no *end node*, minimizando o processo intermédio da transmissão de dados. No geral, o *edge computing* assegura processamento em tempo real local e uma baixa latência.

- Segurança: no *edge computing*, como os dados não precisam de ir na sua totalidade para a *cloud*, permite garantir maior segurança do que no *cloud computing*, por exemplo. O *edge computing* apenas está responsável por dados locais, e mesmo que exista o ataque, apenas afeta o local onde estes estão a ser guardados.

- Baixo custo, consumo de energia e largura de banda: no *edge computing*, como os dados não precisam de ser todos enviados para a *cloud*, os requisitos de largura de banda para a transmissão são menores e de menor custo. O consumo de energia também baixa uma vez que o *edge device* a as necessidades de comunicação para a *cloud* são também menores.

A Tabela 2.1 apresenta uma comparação entre o *cloud computing* e o *edge computing* (Carvalho et al., 2021).

Tabela 2.1: Comparação entre *Edge Computing* e *Cloud Computing*

Características	***Cloud***	***Edge***
Aplicações	Orientado aos dados	Orientado ao utilizador
Arquitetura	Centralizado	Distribuído
Largura de banda	Alto	Baixo
Recursos	Alto	Moderado
Distância aos utilizadores	Longe	Perto
Consumo energético	Alto	Baixo
Latência	Alto	Baixo
Escabilidade	Médio	Alto
Capacidade de armazenamento	Alto	Baixo

Hassan cita alguns projetos que foram desenvolvidos utilizando o *edge computing*. Nas *smart cities* utilizam-se sistemas *edge computing* para controlo de luzes na via pública, monitorização da qualidade de água e ar, explorar diferentes rotas para quando existe um acidente ou catástrofe, rega automática nas cidades, entre outros (Hassan et al., 2018).

Também é normal o uso de *edge computing* em combinação com *cloud computing* (Hassan et al., 2018), para complementar as funcionalidades e valências de projetos IoT, por exemplo, na monitorização de concentração de gás no ar, níveis de água em reservatórios de água, condições de iluminação, humidade de solo, entre outros. Estes projetos de monitorização ambiental são muito importantes em diversas áreas, tais como, agricultura, floresta e qualidade de alimentos.

2.2.1.3 *Fog Computing*

Com a evolução rápida dos sistemas IoT, os sistemas baseados na *cloud* foram sofrendo aperfeiçoamentos na sua arquitectura, para otimizar o processamento de dados e aumentar a eficiência de armazenamento, surgindo outro tipo de computação local designado por *Fog Computing* (Peter, 2015).

A definição de *Fog Computing* foi inicialmente criada pela CISCO (Bonomi and Milito, 2012). Com a criação do *Fog Computing* a computação passou a ser realizada localmente junto dos dispositivos sensores, ou seja, a análise aos dados recolhidos dos sensores

é efectuada na rede dos dispositivos locais, diminuindo assim o tempo de resposta (Peter, 2015).

O *Fog Computing* é um paradigma com capacidades limitadas de computação, armazenamento e serviços de rede local que permite fazer a ponte entre vários dispositivos *end nodes* e a *cloud*. Fornece uma boa solução para aplicações IoT que são particularmente sensíveis à latência na transmissão dos dados (Verma et al., 2016).

Para *Vaquero* e *Rodero-Merino* (Vaquero and Rodero-Merino, 2014), *Fog Computing* refere-se a cenários de interligação de um grande número de dispositivos ubíquos heterogéneos e descentralizados que comunicam e cooperam entre si através da rede local, para efetuar tarefas de armazenamento e processamento sem a intervenção de aplicações externas. Estas tarefas podem ser suportadas por operações e serviços na rede local, num ambiente isolado.

Resumidamente, o *Fog Computing* é uma extensão da *cloud*, mas encontra-se mais perto das "coisas", o que permite o processamento local dos dados. O *Fog Computing* atua como um intermediário entre a camada de *cloud* e *edge*, ou seja, consegue trazer o processamento, armazenamento e serviços de rede para a periferia. Os *Fog Nodes* podem ser instalados em qualquer sítio que tenha uma conexão a uma rede e que possibilitem a receção dos dados dos *edge nodes*. Qualquer dispositivo com capacidade de computação, armazenamento e conexão a uma rede pode ser um *Fog Node*, e.g. *switches*, *routers* e luminárias (*smart lightning*) (Verma et al., 2016).

O *Fog Computing* expande o modelo de *Cloud Computing* para a periferia da rede. Mesmo que o *Fog Computing* tenha muitas semelhanças com o *Cloud Computing*, tais como recursos, mecanismos e atributos, este traz benefícios para os sistemas IoT (Liu et al., 2017). Estes benefícios são, por exemplo:

- Agilidade: permite que as aplicações de *Fog Computing* sejam rapidamente desenvolvidas e implementadas, podendo ser programadas de acordo com as necessidades dos clientes (Cisco et al., 2015).
- Baixa latência: o *Fog Node* tem a capacidade de suportar serviços em tempo real (Peralta et al., 2017).
- Distribuição geográfica escalável: o *Fog Computing* permite fornecer recursos distribuídos e de armazenamento para aplicações de grande dimensão/escala e alcance (Peralta et al., 2017).
- Menor despesa operacional: em vez de enviar todos os dados para a *cloud*, permite o processamento e análise de dados local, poupando largura de banda (Cisco et al., 2015).
- Flexibilidade e Heterogeneidade: o *Fog Computing* permite a colaboração entre os diferentes dispositivos, infraestruturas e serviços locais (Bonomi et al., 2014).

- Escabilidade: o facto de os *Fog Nodes* estarem perto dos dispositivos IoT locais, permite uma maior flexibilidade na cobertura geográfica e número de dispostivios IoT conectados aos serviços (Peralta et al., 2017).

Na Tabela 2.2 apresenta as principais diferenças entre o *Cloud Computing* e o *Fog Computing* (Mouradian et al., 2018).

Tabela 2.2: Diferenças entre *Cloud Computing* e *Fog Computing*

Itens	***Cloud Computing***	***Fog Computing***
Latência	Alta	Baixa
Hardware	Capacidade de armazenamento e computação escalável	Capacidade de armazenamento e computação limitada
Distância entre o cliente e o servidor	Múltiplos saltos	Um salto
Localização dos nós servidor	Na internet	Na periferia da rede local
Ambiente de utilização	Edifício com ar condicionado	Ao ar livre (jardins, via pública)
Implementação	Centralizado	Distribuído
Sensibilização sobre o lugar	Não	Sim

O *Fog Computing* funciona como um intermediário entre a camada de dispostivos IoT e a camada de *cloud*. Traz grande parte do processamento para a periferia da rede local, aumentando a eficiência na transmissão dos dados para a *cloud*. Como os dados são processados em tempo real, aumenta ainda mais esta eficiência (Atlam, Alassafi, Alenezi, Walters and Wills, 2018). O *Fog Computing* oferece uma melhor solução de processamento em tempo real e fornece, às aplicações IoT, maior segurança e eficiência (Ketel, 2017).

A integração de *Fog Computing* nos sistemas IoT traz vários benefícios. Segundo *Atlam, Walters and Wills* (Atlam, Walters and Wills, 2018), o *Fog* suporta comunicação em tempo real entre dispositivos IoT, reduzindo drasticamente a latência destas aplicações. Também permite o suporte a grandes redes de sensores, que é um grande problema em *Cloud Computing*, sendo que estes sensores têm vindo a ser cada vez mais.

O *Fog Computing* consegue fornecer várias maneiras para ultrapassar as limitações das arquiteturas de computação existentes, que dependem da computação em *cloud* e em dispositivos IoT (Chiang and Zhang, 2016).

Bittencourt propôs uma arquitetura de *Fog computing* para uma aplicação IoT que fornece alocação e processamento em tempo real, especialmente apropriada para *smartphones* e carros inteligentes na cidade (Bittencourt et al., 2017).

2.2.2 Tecnologias de Comunicação Sem Fios

Temos assistido a um forte desenvolvimento na vertente das comunicações máquina para máquina. Múltiplos casos de uso na utilização de sensores e atuadores têm vindo a despertar grande interesse na integração deste tipo de comunicações sem fios. O rápido desenvolvimento dos sistemas IoT promoveu também a necessidade de uma evolução das tecnologias e redes de comunicação sem fios, facilitando assim a transferência de dados entre as "coisas" e a Internet (Wang et al., 2017).

2.2.2.1 NB-IoT

O reconhecimento da importância da IoT levou o 3GPP (*3rd Generation Partnership Project*), a introduzir algumas características chave na sua última atualização (Rel-13) (Wang et al., 2017). Esta atualização permitiu ao *Narrowband Internet of things* (NB-IoT) usar novas operações (Ratasuk et al., 2016), tais como:

- Melhoria da cobertura interior: atingir uma cobertura do *General Packet Radio Service* (GPRS) de 20db. Também um aumento da taxa de *uplink* e *downlink* de 160bps é suportado no nível de aplicação.
- Suporte a um número massivo de dispositivos de baixo débito: o objetivo é suportar, pelo menos, 52457 diapositivos num setor da rede celular. Este número foi determinado usando 40 dispositivos por casa (Tanim, 2015).
- Reduzir a complexidade: reduzir a complexidade da implementação de aplicações IoT.
- Melhoria da eficiência energética: providenciar capacidade de bateria para 10 anos.
- Latência: ter no máximo uma latência de 10 segundos.

O NB-IoT é uma nova tecnologia de acesso de rádio 3GPP e foi criada sobre a infraestrutura existente do *Long Term Evolution* (LTE) (Li et al., 2018).

Os principais objetivos do NB-IoT são, assegurar que os dispositivos sejas de baixo custo (até 5$), baixa latência de *uplink* (inferior a 10 segundos), até 40 dispositivos conectados em simultâneo e uma vida de bateria estimada em 10 anos, se existir um envio diário de 200 *bytes*, em média (Li et al., 2018). Uma das grandes vantagens também é a flexibilidade de instalação, que permite ao operador de NB-IoT usar uma pequena porção do espetro disponível da rede. O NB-IoT foi desenhado para atingir ótimos resultados nas redes GSM, GPRS e LTE (Wang et al., 2017).

O NB-IoT possui três modos de instalação: no primeiro, operação em banda, em que o NB-IoT é implementado numa onda portadora LTE; no segundo, operação em banda-guarda, onde o NB-IoT é implementado na banda de guarda do LTE; no terceiro, operação

independente, em que o NB-IoT é usado no lugar de uma ou várias ondas portadoras GSM (Li et al., 2018).

Como o NB-IoT foi desenhado para trabalhar sobre as funcionalidades atuais do LTE, é possível reutilizar o mesmo hardware e partilhar o mesmo espetro, sem que existam problemas. Resumidamente, o NB-IoT adota o modo de funcionamento em que simplesmente se conecta a uma rede LTE. Isto faz com que seja de fácil implementação (Ratasuk et al., 2016). Isto permite que todos os serviços de rede como autenticação, segurança, política, localização e carregamento, sejam completamente suportados (Ratasuk et al., 2016).

Em (Ratasuk et al., 2016), efetuaram-se vários testes nos 3 modos de utilização (operação em banda, operação em banda-guarda e operação independente). Os resultados confirmam que todos os objetivos do NB-IoT foram alcançados.

2.2.2.2 ESP-NOW

O ESP32 é um microcontrolador, desenhado e produzido pela *Espressif System*. Oferece diversas capacidades de comunicação, tais como, *WiFi* e *Bluetooth*. É um microcontrolador de baixo consumo e baixo custo. Ao longo dos anos, o ESP32 tem vindo cada vez mais a receber novas funcionalidades. Uma destas funcionalidades foi o ESP-NOW, um protocolo *peer-to-peer* (P2P), desenvolvido em 2018 (Hoang et al., 2019).

Existem três modos de funcionamento do ESP-NOW: *unicast*, *broadcast* e *multicast*. No modo *unicast* e *multicast* é necessário o emparelhamento, confirmação e existe uma limitação de até 20 diapositivos. No modo *broadcast* não existem estas limitações (Hoang et al., 2019).

O ESP-NOW trata-se de um protocolo de comunicação *WiFi*, que não necessita de conexão à Internet. A comunicação é feita através de uma frame específica, onde depois é transmitido de um dispositivo *WiFi* para outro sem conexão (Systems, 2021). O ESP-NOW suporta algumas funcionalidades tais como (Systems, 2021):

- Distância de envio de 320m a 500m.
- Comunicação *unicast* encriptada e desencriptada.
- Dispositivos emparelhados encriptados e desencriptados.
- Limite de *payload* de 250 *bytes*.
- A função de *callback* pode informar a camada de aplicação se a transmissão foi um sucesso ou falhou.

Resumidamente, o ESP-NOW é um protocolo de comunicação rápido, que pode ser usado para trocar pequenas mensagens, entre microcontroladores ESP32 (Systems, 2022).

Hoang desenvolveu um sistema de comunicações descentralizado, de baixo custo, para ser utilizado em edifícios. Este sistema tira partido da viabilidade em curto alcance, e o facto de ser descentralizado, não comprometendo o sistema caso algum nó tenha algum problema e não consiga efetuar comunicações (Hoang et al., 2019).

2.2.2.3 LoRa

O LoRa é uma tecnologia de comunicação sem fios que utiliza um espetro de 1GHz não licenciado, utilizada em comunicações de grandes distâncias. O termo LoRa refere-se à capacidade de grande alcance da tecnologia, devido ao seu espetro alargado (Sinha et al., 2017).

O LoRaWAN utiliza uma arquitetura em estrela em que as *gateways* encaminham mensagens entre os nós e uma central de rede . Numa rede LoRaWAN, os nós não estão associados a uma gateway específica, ou seja, os dados transmitidos pelo nó é tipicamente recebido por múltiplas *gateways*. Cada *gateway* encaminha os pacotes através de outro tipo de rede (celular, WiFi, etc). Todas as *gateways* estão conectadas a um servidor central, via conexão *IP (Internet Protocol)*. Este servidor é quem filtra os pacotes duplicados de diferentes *gateways*, verifica a segurança, envia *ACKs (Acknowledgements)* para as *gateways* e envia o pacote para o servidor da aplicação específico (Sinha et al., 2017).

A maior vantagem da utilização do LoRa é que esta consegue cobrir uma cidade inteira, com apenas uma *gateway*. Por exemplo, na Bélgica, um país com uma área de aproximadamente $30500km^2$, a rede LoRa apenas necessita de apenas sete estações (Sinha et al., 2017).

O LoRa é considerado uma tecnologia barata. Usa uma topologia em estrela em vez de *mesh*, reduz a complexidade de *hardware* e não é necessário uma grande infraestrutura (uma estação consegue conectar até 1000 dispositivos). Apesar do LoRa ser uma tecnologia mais barata, comparativamente ao *NB-IoT*, esta requer mais *hardware* o que torna o custo do sistema mais elevado (Muteba et al., 2019).

2.2.3 Protocolos de Comunicação IoT

2.2.3.1 MQTT

O MQTT foi criado pela IBM (*International Business Machines Corporation*), como um protocolo leve para troca de mensagens numa arquitetura *publish/subscriber*, baseado em *brokers*. É um protocolo criado para comunicações *machine-to-machine* (M2M), usando o protocolo *Transmission Control Protocol* (TCP)/IP. O MQTT é usado em dispositivos que possuem baixa memória e baixo poder de processamento. O seu *payload* está limitado a 256MB (*MegaBytes*), o que leva a que este protocolo seja leve e possa ser usado em redes instáveis (Nastase, 2017).

No MQTT, quem gere todas as mensagens entre quem publica e os subscritores é o *broker*. Este está instanciado do lado do servidor, e trata de encaminhar as atualizações para todos os clientes, ou seja, os clientes não comunicam entre si, mas sim com o *broker*. Cada mensagem MQTT possui um tópico, organizado num modelo de árvore, onde os clientes podem publicar e subscrever. Quando é publicada uma mensagem (dados ou comandos), o *broker* trata de encaminhar essa mensagem para todos os clientes que estão subscritos ao tópico da mensagem. Este protocolo também foi desenhado para efetuar comunicações assíncronas, e também fornece transferências confiáveis, escolhendo um dos três tipos de mecanismos, que também são chamados de QoS (*Quality of Service*) (Dinculeană and Cheng, 2019). Estes três tipos de mecanismos (Soni and Makwana, 2017) são:

- QoS0 (no máximo uma vez): a mensagem é enviada uma vez e não existe garantia da entrega desta.
- QoS1 (pelo menos uma vez): os dados são enviados pelo menos uma vez e é possível enviar a mensagem mais que uma vez.
- QoS2 (exatamente uma vez): a mensagem é enviada exatamente uma vez usando *4-way handshaking*.

A arquitetura do MQTT pode ser dividida em dois componentes principais, o cliente e o *broker*. O modo de funcionamento está descrito na Tabela 2.3 (Soni and Makwana, 2017).

Tabela 2.3: Componentes principais do MQTT

Cliente	***Broker***
Pública mensagens para os utilizadores interessados	Aceita pedidos do cliente
Subscreve em tópicos de interesse para receber mensagens	Recebe mensagens publicadas pelos utilizadores
Anula a subscrição dos tópicos de interesse	Processa pedidos de publicações e subscrições dos utilizadores
Desliga-se do *broker*	Depois de receber mensagens do publicador, envia para os utilizadores interessados

O MQTT foi desenvolvido para ser utilizado em dispositivos de baixo poder de armazenamento e processamento. Existem diversas áreas onde o MQTT foi utilizado, entre elas a área da saúde, energia e redes sociais (Lampkin et al., 2012).

A simplicidade e código aberto do MQTT permite que este protocolo seja adequado para ambientes limitados, como vemos muitas vezes no IoT, onde existe baixo poder de processamento, memória e largura de banda (Soni and Makwana, 2017).

As ceifeiras têm com função debulhar as colheitas de cereais e armazenar no seu contentor de carga. Este contentor quando se encontrar cheio necessita de ser descarregado para transportadoras que se encontram no local, para a ceifeira continuar com o seu trabalho. Para poupar tempo neste processo, é necessário que o descarregamento seja feito em tempo real e enquanto está a processar a ceifa dos cereais. Para isto pode-se utilizar o protocolo MQTT, que permite comunicações em tempo real da localização da ceifeira debulhadora e o nível de enchimento do seu tanque. Estas mensagens são transmitidas para um *broker* central, onde depois é comunicada para a ceifeira debulhadora como sendo o consumidor (Bauer and Aschenbruck, 2017).

2.2.3.2 CoAP

O *Constrained Application Protocol* (CoAP) é um protocolo de transferência de dados leve baseado em REST (*Representational state transfer*), utilizado em sistemas IoT devido ao seu baixo consumo energético (Bandyopadhyay and Bhattacharyya, 2013). Este protocolo é muito parecido ao *Hypertext Transfer Protocol* (HTTP), mas para evitar o *overhead* do TCP, utiliza o UDP (*User Datagram Protocol*). Este protocolo funciona em duas camadas lógicas: no topo as camadas de pedido/resposta, que são responsáveis pela redundância e consistência das mensagens, e camada de mensagens em baixo, que é responsável pela conectividade e comunicação (Al-Haidari and Alqahtani, 2020).

O CoAP é um protocolo do tipo cliente-servidor, que faz uso de um modelo de troca de mensagens. Pode enviar vários tipos de pedidos (*GET, POST, PUT e DELETE*), tal como o HTTP, para um servidor usando o URL (*Uniform Resource Locator*) do servidor. O CoAP adota o protocolo UDP (*User Datagram Protocol*) para ser utilizado em conexões de baixa largura de banda e em dispositivos de baixo poder de computação (Al-Haidari and Alqahtani, 2020).

As áreas de aplicação do CoAP (Bandyopadhyay and Bhattacharyya, 2013) são tipicamente com sensores (IoT), em projetos de energia inteligente, automação de edifícios, entre outros.

2.2.3.3 HTTP

O *HTTP ((Hyper Text Transport Protocol)* é um protocolo de mensagens *web*. Suporta a arquitetura *RESTful Web* de pedido/resposta e usa *URI (Universal Resource Identifier)*, em vez de tópicos. O servidor envia dados via *URI* e o cliente recebe esses dados através deste. Usa o *TCP (Transmission Control Protocol)* como protocolo de transporte e *TLS/SSL (Transport Layer Security/Secure Sockets Layer)* para segurança (Naik, 2017).

O protocolo HTTP envia pequenos pacotes para o servidor para se conectar. Isto pode levar a um alto uso de tráfego o que pode causar a um elevado uso de recursos de rede e, consequentemente, levar a *delays* (Wukkadada et al., 2018).

2.2.3.4 MQTT vs CoAP vs HTTP

Para comparação entre os dois protocolos (MQTT e CoAP) considerados para este projeto, (Bandyopadhyay and Bhattacharyya, 2013) fez a comparação entre estes protocolos.

Tabela 2.4: Diferenças entre MQTT e CoAP

Características	**MQTT**	**CoAP**
Arquitetura	Cliente/*Broker*	Cliente/*Broker* ou Cliente/Servidor
Abstração	Publicador/Subscritor	Pedido/Resposta ou Publicador/Subscritor
Tamanho do *Header*	2 *Byte*	4 *Byte*
Tamanho da mensagem	Máximo de 256MB	Pequena para caber num datagrama IP
Semânticas	Conectar, desconectar, publicar, subscrever, anular subscrição, fechar	*Get, post, put, delete*
Protocolo de transporte	TCP	UDP, SCTP
Segurança	TLS/SSL	DLTS/IPSec
Consumo de energia	Maior	Menor
Latência	Maior	Menor
Largura de banda	Maior	Menor

Analisando a Tabela 2.4, podemos ver que existem algumas diferenças entre o MQTT e o CoAP. As suas vantagens e desvantagens dependem do tipo de projeto que se utilizar. Em projetos onde exista uma grande quantidade de comunicações, e seja necessário um grande *payload*, o MQTT parece ser uma melhor solução. Também o facto do MQTT possuir TCP permite que as mensagens sejam entregues com fiabilidade, e assim adotar mecanismos para não existir perda de mensagens.

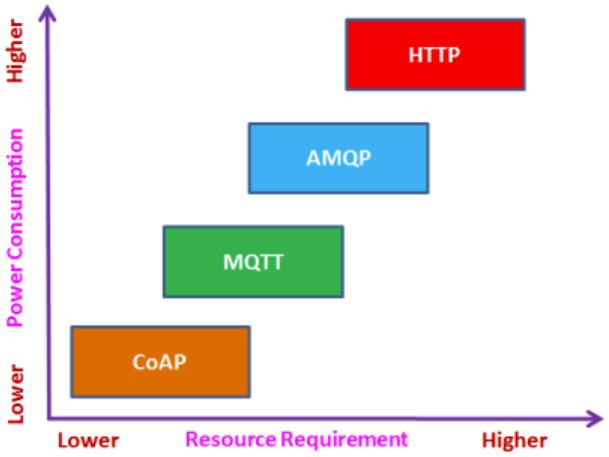

Figura 2.1: Consumo energia vs Utilização de recursos

Fonte: (Naik, 2017)

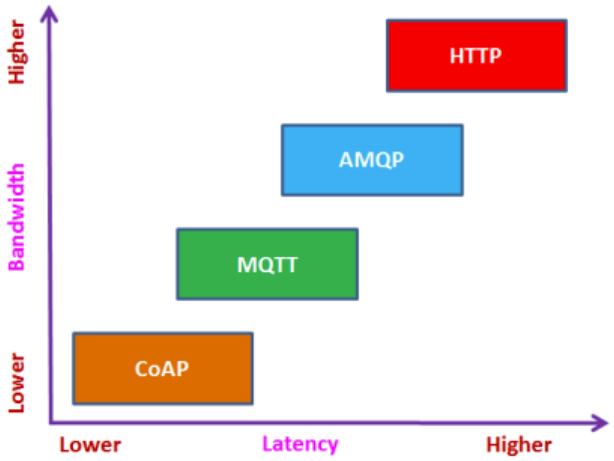

Figura 2.2: Largura de banda vs Latência

Fonte: (Naik, 2017)

Relativamente ao HTTP este foi imediatamente descartado devido a vários fatores. O alto consumo (ver Figura 2.1) e alta latência (ver Figura 2.2) fizeram com que este protocolo fosse descartado para este sistema. Como o objetivo deste sistema é que este fosse de baixo consumo e com uma alta taxa de sucesso no envio de dados a utilização deste protocolo não ia ser a mais adequada, visto termos melhores soluções.

2.2.4 Tecnologias de Sensorização de Sistemas de Água

Nesta subsecção abordam-se diversos sensores que foram estudados para potencial aplicação no *Smart-city Drainage System*. Alguns destes sensores foram utilizados em

projetos afins, enquanto outros foram identificados através de uma pesquisa em sites de revendedores especializados.

2.2.4.1 Sensor de fluxo de líquido

Figura 2.3: Sensor de Fluxo

Fonte: `https://www.dfrobot.com/product-1517.html`

Com o sensor de fluxo[1] conseguimos determinar a quantidade de fluxo de líquido que atravessa o sensor. É um sensor com um baixo custo e baixo consumo energético.

Os componentes principais deste sensor são, o íman, a roda da turbina (ou pás), e o sensor que deteta o efeito *hall*, como demonstra a Figura 2.4. A corrente de líquido faz com que a roda e o íman girem criando assim uma rotação no campo magnético. O sensor que deteta o efeito *hall* é acionado, respondendo com valores digitais altos e baixos (pulsos). Por cada volta, o volume de líquido que atravessa o sensor é uma certa quantidade, assim como o número de *outputs* devolvidos pelo sensor de efeito *hall*. Concluindo, se contarmos o número de vezes que o sensor devolve uma resposta, é possível calcular o fluxo do líquido.

Um Sensor de efeito *hall* é um transdutor que, quando sob a aplicação de um campo magnético, responde com uma variação na sua tensão de saída.

Poderíamos colocar este sensor na entrada e saída da caixa de drenagem/coletora, e em dias chuva, analisar o fluxo de água para saber se existe algum tipo de problemas de drenagem.

[1] `https://www.dfrobot.com/product-1517.html`

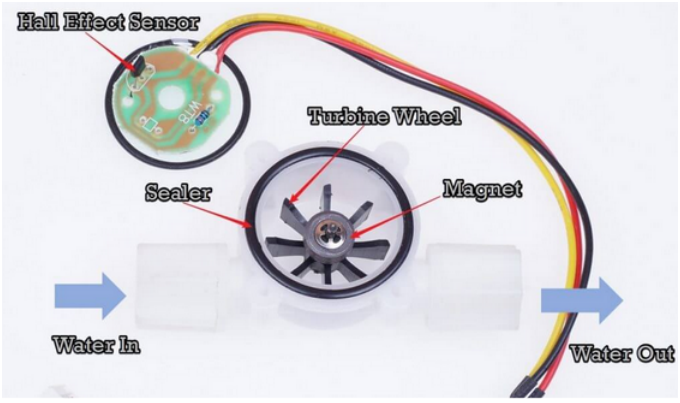

Figura 2.4: Composição interna do sensor de fluxo

Fonte: https://wiki.seeedstudio.com/Water-Flow-Sensor/

2.2.4.2 Sensor de Pressão de Água

Figura 2.5: Sensor de Pressão de Água

Fonte: https://www.dfrobot.com/product-1675.html

O sensor de pressão de água[1] é composto por um material de silício mono cristalino, e quando é exercida força sobre este material, existe uma alteração infinitesimal e uma mudança a nível eletrónico da estrutura do átomo interno, o que levará a uma grande alteração na resistividade (*Factor H Mutation*) e, consequentemente, na resistência. Este efeito físico é um efeito piezoresistivo. O efeito piezoresistivo é uma alteração da resistividade elétrica de um semicondutor ou metal quando é aplicada tensão mecânica.

A escolha deste sensor esteve no facto de este ter certificação IP68, o que proporciona uma grande resistência a água e lixo, e à possibilidade de conseguirmos ler a pressão de água.

A ideia seria ter este sensor no tubo ou fundo da caixa coletora/drenagem e efetuar as leituras de pressão de água. Com isto poderíamos relacionar uma maior pressão ao facto

[1] https://www.dfrobot.com/product-1675.html

de a caixa estar com dificuldades em drenar a água.

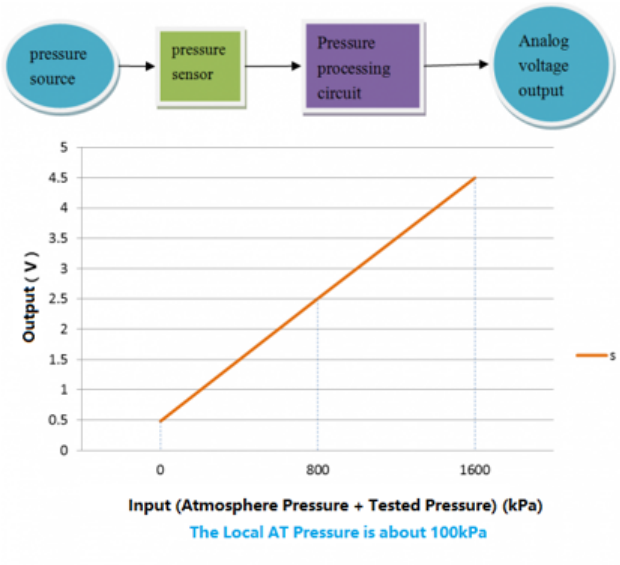

Figura 2.6: Relação entre *output* do sensor e a pressão

Fonte: `https://wiki.dfrobot.com/Gravity__Water_Pressure_Sensor_SKU__SEN0257`

Na Figura 2.6 podemos ver a relação entre o *output* do sensor e a pressão a que este está sujeito. O sensor funciona da forma em que é colocado num local sobre certa pressão, lê o valor, processa através de um circuito interno, e finalmente representa o valor sob a forma de um *output* de corrente em *volts*. Quanto maior for a *output* do sensor, maior será a pressão a que este está sujeito.

2.2.4.3 Sensor de Ultrassons

Figura 2.7: Sensor de Ultrassons

Fonte: `https://www.dfrobot.com/product-1935.html`

O sensor de ultrassons[1] contém tipicamente dois transdutores ultrassónicos, em que um atua como transmissor e o outro como recetor. O transdutor transmissor emite impulsos sonoros para o meio e é recebido pelo transdutor recetor. A partir da diferença do tempo em que o impulso foi emitido e o que foi recebido, conseguimos calcular a distância dos obstáculos.

A ideia para o uso deste sensor seria detetar o nível de lixo presente na caixa coletora/-drenagem. Com a altura da caixa e o cálculo da distância do sensor poderíamos saber a que nível de lixo ou água a caixa teria no momento da leitura. É um sensor que tem certificação IP67, baixo custo, baixo consumo e o seu *blind zone* é relativamente baixo (3cm), o que leva a considerar a sua escolha para uso no sistema.

Este sensor também foi utilizado num projeto de monitorização de resíduos urbanos (Kanade et al., 2021). O propósito deste sensor era para efetuar as leituras do nível de resíduos que se encontravam nos caixotes do lixo nas zonas urbanas. Esta leitura era efetuada através do tempo em que a onde ultrassónica era enviada do sensor até ao tempo em que era recebida pelo mesmo (ver Figura 2.8). Esta diferença de tempo permitia calcular a distância a que o sensor se encontrava dos resíduos, ou seja, menor o tempo, maior quantidade de lixo se encontrava no caixote.

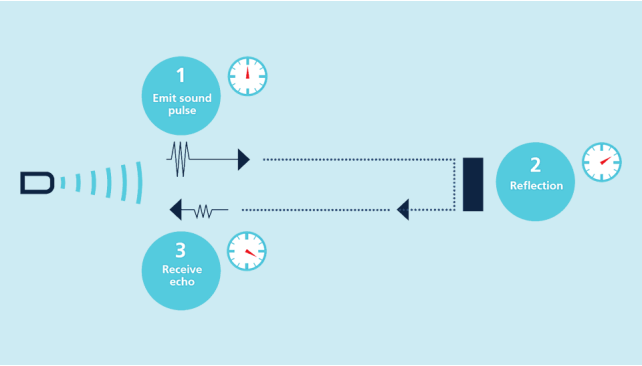

Figura 2.8: Princípio do sensor de ultrassons

Fonte: `https://www.microsonic.de/en/support/ultrasonic-technology/principle.htm`

[1] `https://www.dfrobot.com/product-1935.html`

2.2.4.4 Sensor Capacitivo Sem Contacto

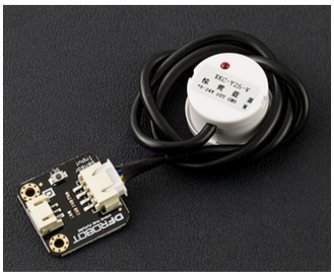

Figura 2.9: Sensor Capacitivo Sem Contacto

Fonte: https://www.dfrobot.com/product-1493.html

O sensor capacitivo sem contacto[1] é capaz de detetar a presença de líquidos condutores (por ex. água) sem contacto (ver Figura 2.10). Utiliza o princípio da capacitância para saber se existe líquido do outro lado da superfície. A capacitância é a grandeza escalar que mede a quantidade de energia que um condutor elétrico consegue armazenar.

O uso deste sensor incide sobretudo no uso para a deteção do nível de água na caixa coletora/drenagem. A utilização deste sensor iria ser na parte de fora da caixa, visto ser um sensor não resistente à água. É um sensor de baixo custo e de baixo consumo.

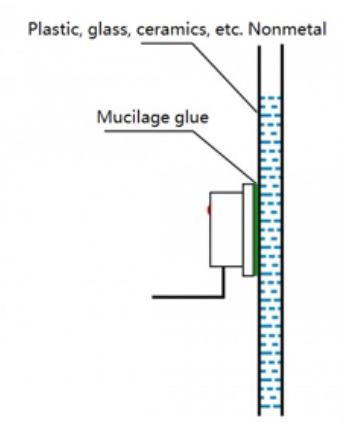

Figura 2.10: Modo de funcionamento do Sensor Capacitivo Sem Contacto

Fonte: https://wiki.dfrobot.com/Non-contact_Liquid_Level_Sensor_XKC-Y25-T12V_SKU__SEN0204

[1] https://www.dfrobot.com/product-1493.html

2.2.4.5 Sensor Fotoelétrico de Nível de Água

Figura 2.11: Sensor Fotoelétrico de Nível de Água

Fonte: https://www.dfrobot.com/product-1470.html

O sensor fotoelétrico de nível de água[1], tal como o capacitivo sem contacto, permite a deteção do nível de água presente na caixa coletora/drenagem. É um sensor resistente, baixo custo e baixo consumo.

O sensor usa o princípio tradicional ótico. Quando a água entra em contacto com o sensor este retorna 1 (existência de água), caso contrário retorna 0. É resistente à corrosão e aguenta a altas pressões e temperaturas.

A ideia seria utilizar este sensor na parte interna da caixa, num certo nível de emergência, em que a deteção positiva de água seria considerada um alerta para a caixa coletora/-drenagem.

2.2.4.6 Sensor Capacitivo de Humidade do Solo

Figura 2.12: Sensor Capacitivo de Humidade do Solo

Fonte: https://www.dfrobot.com/product-2054.html

[1] https://www.dfrobot.com/product-1470.html

O sensor capacitivo de humidade do solo[1], também usa a capacitância para a deteção de presença de água no ambiente. Ao contrário do sensor sem contacto (cf. seccção 2.2.4.4), este tem de estar em contacto com o material. Como o nome indica, é um sensor de deteção de água no solo, que é usado para determinar se o solo precisa de ser regado ou não.

O caso de uso no sistema seria para a deteção da quantidade de lixo e água depositados na caixa coletora/drenagem. É um sensor com baixo consumo, resistente à água, anticorrosivo e de baixo custo.

O sensor dispõe de uma Tabela 2.5 com referência a intervalos de valores para a quantidade de humidade que se encontra no solo, dependendo dos valores de leitura do sensor.

Tabela 2.5: Relação qualidade do solo com valores de leitura do sensor

Tipo de solo	Mínimo	Máximo
Seco	380	570
Húmido	190	380
Muito húmido	0	190

Este sensor é utilizado num projeto de agricultura inteligente (Nagaraja et al., 2019), onde é utilizado pela sua capacidade de deteção da humidade do solo. É utilizado para ativar um motor de rega que, quando o solo se encontra abaixo de um certo *threshold*, este ativa o motor para regar o solo.

2.2.4.7 Sensor de TDS

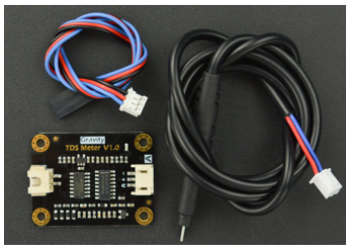

Figura 2.13: Sensor de TDS

Fonte: `https://www.dfrobot.com/product-1662.html`

O sensor de TDS[2] (*Total Dissolved Solids*) funciona através da emissão de impulsos elétricos e a sua futura leitura. O *output* vai-se traduzir em ppm (partes por milhão) que corresponde ao número de partículas presentes no líquido condutor. Se não existir

[1] `https://www.dfrobot.com/product-2054.html`
[2] `https://www.dfrobot.com/product-1662.html`

líquido o sensor retorna 0ppm. Dependendo do *output*, podemos tirar conclusões acerca da qualidade da água que se encontra para leitura. A Figura 2.14 demonstra a qualidade da água dependendo do *output* do sensor.

Este sensor, no contexto do sistema, pode permitir a deteção de descargas ilegais numa caixa de drenagem específica. É um sensor resistente a lixos e água, baixo custo e de baixo consumo energético.

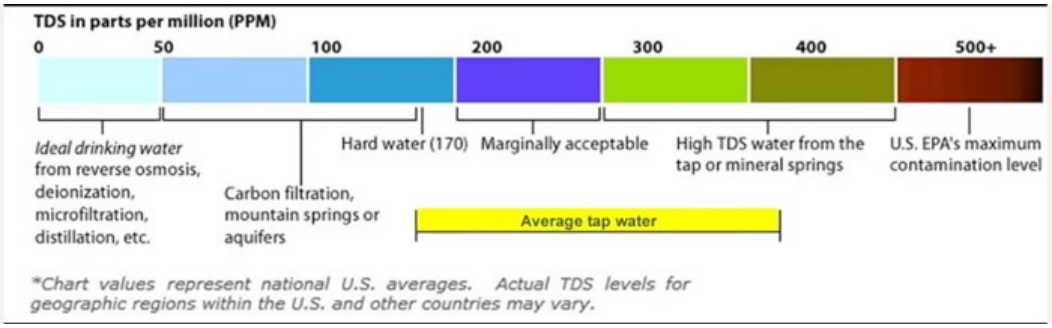

Figura 2.14: Qualidade da água referente ao output do sensor de TDS

Fonte: `https://www.dfrobot.com/product-1662.html`

2.3 Sistemas de Drenagem de Águas Pluviais

A finalidade dos sistemas de drenagem de águas pluviais é recolher e transportar as águas provenientes de chuvas e tempestades (Pedroso et al., 2013). Estes sistemas são importantes para garantir a correta drenagem das águas e assim evitar estragos na via pública e edifícios, onde consequentemente, a autarquia poderia ter de pagar indemnizações sobre estes estragos.

Os sistemas de drenagem podem classificar-se em dois tipos: naturais e artificiais/urbanos. Nos sistemas de drenagem naturais a água escoa naturalmente e não precisa de mão humana para existir este escoamento. Pelo contrário, os sistemas artificiais/urbanos são aqueles criados por mão humana, onde no passado não existia escoamento possível das águas pluviais (Pedroso et al., 2013).

As redes de drenagem de águas pluviais são constituídas, essencialmente, pela rede de coletores e órgãos acessórios (Pedroso et al., 2013). A rede de coletores corresponde ao conjunto de canalizações, que tem como função a condução de águas pluviais até aos seus pontos finais de drenagem, tais como, mares ou rios. Os órgãos acessórios são compostos pelos dispositivos de entrada e por câmaras de visita. Os dispositivos de entrada são aqueles que estão nas bermas das estradas, vulgarmente designados por bueiros ou caixas de drenagem. As câmaras de visita, também chamadas de caixas coletoras, são as caixas que fazem as ligações aos dispositivos de entrada, ou seja, são caixas intermédias que permitem operações de manutenção e limpeza.

A pesquisa de trabalhos relacionados permitiu identificar várias soluções IoT de monitorização para diferentes arquiteturas de drenagem de águas pluviais em diversos países, mas nenhuma solução que se utilizasse em Portugal.

Um dos projetos menciona uma solução implementada na Flórida (Estados Unidos) (Chang and Guo, 2006), que consiste num sistema com 3 módulos: um sensor de ultrassons para monitorizar o nível de água, uma câmara para gravar vídeos e um módulo de processamento de dados. Este sistema tem a vantagem de poder monitorizar o nível de água com o sensor de ultrassons e permitir gravações de vídeo para deteção de cheias. As desvantagens inerentes a esta solução estão relacionadas com o custo e as dificuldades de instalação no terreno.

Outro projeto propõe uma solução implementada em *Barranquilla* (Colômbia) (Cama-Pinto et al., 2016) baseada num sistema IoT composto por nós espalhados pela cidade. Cada nó possui sensores de temperatura, humidade e um sensor de pressão atmosférica. Possui ainda um módulo de transformação de energia solar em elétrica. Usa a tecnologia XBee para as comunicações entre os nós. A vantagem deste sistema reside no baixo custo e na capacidade de fazer previsões, com base nas variáveis atmosféricas, sobre a possibilidade de precipitação como causa inundações repentinas. As desvantagens estão associadas com o facto de o sistema não estar instalado nas caixas de drenagem, logo não conseguir detetar uma cheia diretamente do local exato onde esta está a ocorrer. O sistema também não permite determinar o estado em que se encontram estas caixas de drenagem relativamente ao nível de lixos que possam causar entupimentos.

Em *Manila* (Filipinas) (Garcia et al., 2015) foi criado um sistema para a deteção de cheias e alerta de condutores sobre caminhos alternativos, para assim evitar as cheias. Cada nó do sistema utiliza um sensor de pressão e um pluviómetro. Utiliza também um painel fotovoltaico para a produção de energia que alimenta o sistema. As vantagens estão associadas ao baixo custo e à autonomia energética do sistema. As desvantagens, tal como no caso anterior, referem-se ao facto do sistema não ser instalado nas caixas de drenagem de águas pluviais, nem informar sobre a quantidade de lixo presente nestas.

Por fim, em *Nakhon Si Thammarat* (Tailândia), foi criado um sistema para a deteção de cheias em tempo real (Sunkpho and Oottamakorn, 2011). Este sistema consiste em 2 módulos principais, um que funciona nas áreas onde se pretende detectar a chuva e outro, instalado no centro de controlo. Em cada zona de deteção é usado um pluviómetro para medir a intensidade e quantidade de chuva e um dispositivo ultrassónico, designado por *STARFLOW*, para medir o nível e velocidade da água. As vantagens deste sistema estão associadas ao facto de ser permitir identificar a possibiliade de ocorrência de eventuais cheias em tempo real. As desvantagens prendem-se, mais uma vez, com o custo elevado e com o facto de não ser instalado nas caixas coletoras/drenagem e, por isso, não ser capaz de detetar o respetivo nível de lixo e possibilidade de entupimento do sistema de drenagem.

2.4 Conclusão

Neste capítulo foram abordados diversos tópicos importantes para a realização desta dissertação. Começou-se por abordar o conceito de *smart cities* e a sua evolução ao longo dos anos, bem como a sua interligação com a utilização de sistemas IoT. Realçou-se ainda um dos aspetos principais das *smart cities* relacionado com estas possibilitarem um melhor estilo de vida e segurança aos seus cidadãos.

A utilização e evolução de sistemas IoT foi também abordada, procurando caracterízar-se os aspetos fundamentais e características destes sistemas. Foram também identificados e analisados alguns exemplos de projetos IoT aplicados à monitorização de cidades inteligentes. Dentro do tópico IoT, procurou-se detalhar diversos aspetos.

A subsecção de tecnologias de comunicação abordou diversos tipos de arquiteturas e protocolos de comunicação e permitiu-nos concluir que a solução mais adequada a este trabalho seria a utilização de uma arquitetura híbrida de *Cloud computing* com *Fog computing* e *Edge computing*. Nesta solução utilizam-se *Gateways* locais que recebem os dados recolhidos pelo sensores dos vários *Edge devices*. Cada *Fog device*, tipicamente acoplado a uma luminária, processa e armazena os dados para os enviar posteriormente para a *Cloud*, onde residem todos os dados de todas as *Gateways* espalhadas pelo cidade ou mesmo pais.

Na subsecção de tecnologias de coordenação, abordaram-se os protocolos de comunicação com a *Cloud*, nomeadamente dois em particular: (MQTT e CoAP). Sendo estes os principais protocolos utilizados em sistemas IoT, fez sentido estudarmos ambos para a utilização neste sistema. Concluímos que o MQTT poderia ser o protocolo mais adequado para o sistema proposto, uma vez que se trata de um protocolo eficiente, que utiliza TCP e assim oferece maior fiabilidade no envio e receção de dados. Para além disso este protocolo oferece um modelo assíncrono de comunicação (*publish/subscribe*) adequado à transmissão dos dados entre as várias camadas da arquitetura.

A subsecção de sensores, identificou e descreveu vários sensores que poderiam ser utilizados no sistema proposto nesta dissertação. A escolha destes sensores baseou-se na revisão bibliográfica de vários projetos e em sites especializados na venda de sensores. Foram considerados sensores que pudessem ser utilizados em ambientes e condições atmosféricas adversas e que fizessem sentido na monitorização de sistemas de drenagem de águas.

Por fim procurou-se abordar os sistemas de drenagem de águas pluviais no contexto Português, referindo a sua importância, constituição e arquiteturas. Foram também descritos e analisados diversos trabalhos realizados no âmbito de sistemas IoT de monitorização de drenagem de águas pluviais em todo o mundo. Até ao momento não identificamos trabalhos realizados em Portugal, contudo foi possível perceber que a drenagem de águas pluviais é um tema que preocupa cada vez mais os responsáveis pela gestão urbana,

não só em Portugal, mas também um pouco por todo o mundo. Percebemos que existem alguns esforços que procuram soluções para se combater este problema, mas as alterações climáticas irão potenciar cada vez mais soluções deste género.

Chapter 3

Especificação do Smart-city Drainage System

Este capítulo começa por descrever a estrutura genérica dos sistemas de drenagem de águas pluviais que se encontram instalados na via pública atualmente. Depois passa aos cenários de aplicação em que podemos utilizar o sistema de monitorização de drenagem de águas pluviais. Com base nestes cenários foi possível fazer o levantamento de requisitos funcionais, não funcionais e do sistema (hardware e software). Posteriormente apresenta-se a arquitetura do sistema proposto, detalhando-se os seuss componentes e respetivas trocas de mensagens. Por fim, apresenta-se uma análise de todos os sensores que foram considerados para o protótipo do sistema proposto. Descreve-se em particular a lógica de funcionamento dos sensores e a sua potencial utilização na monitorização das caixas de drenagem.

3.1 Estrutura do Sistema de Drenagem de Águas Pluviais

O sistema de drenagem de águas pluviais, de forma genérica, é composto por caixas de drenagem e caixas coletoras. As caixas de drenagem, também chamados de bueiros, recebem as águas que provêm das chuvas. Estes bueiros estão todos conectados a um tubo principal que recebe toda a água proveniente dos bueiros. Para cada conjunto de bueiros existem caixas coletoras (*manholes*), que estão ligadas ao tubo principal. Estas caixas estão encarregues de receber toda a água proveniente dos bueiros e encaminhá-la até à próxima caixa coletora ou onde a água vai desaguar (cf. rio, afluente, etc.).

Na Figura 3.1 podemos ver um esquema de um bueiro unitário. Como mencionado anteriormente, cada bueiro recebe as águas da chuva provenientes, tanto da casa como da via pública. Posteriormente, as águas coligidas são encaminhadas para um tubo principal. Importante salientar que a rede do sistema de águas pluviais está separado da rede de

águas de esgoto (fora do âmbito desta dissertação).

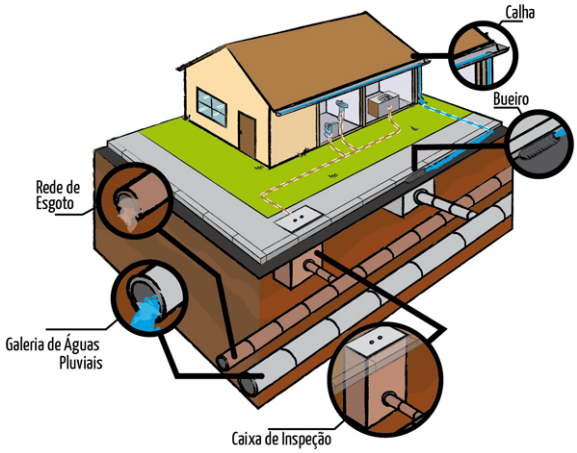

Figura 3.1: Arquitetura do Sistema de Drenagem de Águas Pluviais Unitário

Fonte: `https://site.sabesp.com.br/site/imprensa/noticias-detalhe.aspx?secaoid=65&id=5054`

Na arquitetura geral do sistema de drenagem de águas pluviais, como mostra a Figura 3.2, existem diversos grupos de bueiros que estão conectados ao tubo principal. A água é recebida pelos bueiros que, por ação da gravidade faz com que esta entre no tubo principal e escoe até à caixa coletora. As setas na figura indicam a direção que a água leva em todo o processo de drenagem. A caixa coletora pode estar conectada a outra caixa do mesmo tipo, com outros grupos de bueiros pelo meio, ou pode ser uma caixa final para escoar toda a água coligida.

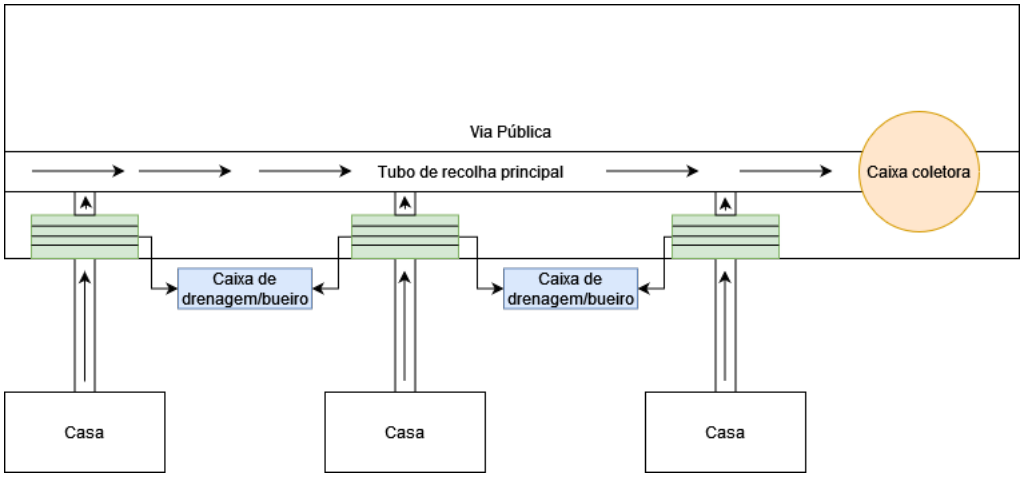

Figura 3.2: Arquitetura do Sistema de Drenagem de Águas Pluviais Geral

Fonte: Própria

3.2 Cenários de Aplicação

Neste trabalho pretende-se proceder à recolha de dados, através de sensores colocados em bueiros e caixas coletoras. Estes dados deverão permitir perceber se existe alguma obstrução ou possível entupimento no sistema de drenagem.

O sistema de monitorização pretende ser utilizado para prevenir que ocorram cheias, alertando a autarquia sobre potenciais anormalias que se encontrem em certas zonas públicas, e assim prevenir danos na via pública. Cada vez mais as *smart cities* têm sido um tema de muita importância. Tornar as cidades mais inteligentes, consequentemente mais seguras é uma prioridade dos dias de hoje. Este sistema poderá contribuir para prevenir a ocorrência de eventuais catástrofes na via pública, enquadrando-se por isso nos cenários de monitorização das *smart cities*.

Os bueiros e caixas coletoras são locais com condições físicas adversas, sujeitas à passagem de muita água e também de muitos detritos e sugidade. Consequentemente, a solução tecnológica de monitorização tem de ter considerar requisitos tanto a nível da precisão dos sensores como também de resistência do hardware colocado nestas caixas.

3.3 Requisitos do Sistema

3.3.1 Requisitos funcionais

Na Tabela 3.1 estão representados os requisitos funcionais (RF) do sistema. Na coluna Componentes apresentam-se os dispositivos ou serviços do sistema associado ao respectivo requisito. O dispositivo *Edge* refere-se ao nó de monitorização, o *Fog* representa a *Gateway* e a *Cloud* corresponde aos serviços de *backend* implementados na *Amazon Web Services* (AWS).

Tabela 3.1: Requisitos Funcionais (RF)

ID	Componente	Descrição
RF1.1	*Edge*	Deve ser capaz de medir a quantidade de lixo e água que se encontra depositado no bueiro ou caixa coletora.
RF1.2	*Edge*	Deve ser capaz de enviar os dados recolhidos dos sensores para o dispositivo *Fog*.
RF1.3	*Edge*	Deve ser capaz de receber dados relativos à previsão metereológica provenientes dos dispositivos *Fog*.
RF1.4	*Edge*	Deve ser capaz de alterar o seu modo/ciclo de funcionamento.
RF2.1	*Fog*	Deve ser capaz de receber periodicamente os dados dos dispositivos *Edge* locais.

RF2.2	*Fog*	Deve ser capaz de guardar localmente todos os dados recebidos dos dispositivos *Edge* locais.
RF2.3	*Fog*	Deve ser capaz de processar e sincronizar os dados recolhidos dos dispositivos *Edge* com o serviço de *Cloud*.
RF2.4	*Fog*	Deve ser capaz de obter e enviar a previsão metereológica para os dispositivos *Edge* associados.
RF3.1	*Cloud*	Deve ser capaz de receber os dados provenientes dos dispositivos *Fog*.
RF3.2	*Cloud*	Deve ser capaz de guardar os dados transmitidos pelos dispositivos *Fog* remotamente.
RF3.3	*Cloud*	Deve ser capaz de apresentar todos os dispositivos *Edge* e *Fog* existentes, num mapa.
RF3.4	*Cloud*	Deve ser capaz de mostrar alertas numa página *Web*.
RF3.5	*Cloud*	Deve ser capaz de registar novos dispositivos *Fog*.
RF3.6	*Cloud*	Deve ser capaz de registar novos dispositivos *Edge* e associá-los a um dispositivo *Fog*.

3.3.2 Requisitos não funcionais

Na Tabela 3.2 representam-se os requisitos não funcionais (RNF) do sistema. Tal como na tabela anterior, a coluna Componente apresenta o dispositivo associado ao requisito descrito.

Tabela 3.2: Requisitos Não Funcionais (RNF)

ID	Componente	Descrição
RNF1.1	*Edge*	Deve ser de baixo custo.
RNF1.2	*Edge*	Deve ter baixo consumo energético, garantindo autonomia 6 meses.
RNF1.3	*Edge*	Deve ser de fácil instalação nos bueiros e caixas coletoras.
RNF1.4	*Edge*	Deve possuir elevada resistência a detritos e água.
RNF2.1	*Fog*	Deve estar constantemente ligado a uma fonte de energia elétrica.
RNF2.2	*Fog*	Deve poder ser instalado numa luminária.
RNF3.1	*Cloud*	Deve oferecer um bom mecanismo de segurança.
RNF3.2	*Cloud*	Deve ser escalável.
RNF3.3	*Cloud*	Deve oferecer interfaces intuitivas e fácil usabilidade.

3.4 Arquitetura do Sistema

A arquitetura do sistema está representada na Figura 3.3. Esta arquitetura é composta por três camadas: os dispositivos *Edge* correspondentes aos nós de monitorização, os dispositivos *Fog* locais, instalados nas luminárias e o serviço de *Cloud* alojado na AWS. Os nós de monitorização utilizam sensores, são alimentados por baterias e utilizam microcontroladores ESP32. Estes nós de monitorização comunicam os dados obtidos dos sensores para a *Gateway* através do protocolo ESP-NOW. A *Gateway* será instalada nas luminárias de modo a terem energia elétrica constante e estarem ao alcance de vários dispositivos *Edge* instalados nos bueiros das vias públicas.

A *Gateway* possui um microcontrolador ESP32 e uma Raspberry Pi. O ESP32 será responsável por receber os dados dos nós de monitorização via ESP-NOW e entregá-los à Raspberry Pi, via porta série. A Raspberry recebe estes dados, processa-os e guarda-os numa base de dados local. Estes dados irão ser posteriormente sincronizados com os serviços da *Cloud*. A Raspberry Pi funciona, portanto, como *Fog Device*. A tecnologia de comunicação usada entre a *Gateway* e a *Cloud* é NB-IoT, sobre a qual são transmitidos os dados usando uma ligação MQTT.

Na *Cloud* os dados são recebidos por um *broker* MQTT e posteriormente processados e guardados numa base de dados. O *broker* utilizado é o *Mosquitto*, e permite gerir as mensagens entre as *gateways* e a API de *backend*. As *gateways* publicam mensagens via *broker* e a API consume essas mensagens e processa os dados a armazenar na base de dados central.

A API baseia-se na tecnologia *Python Flask*, e tem como funcionalidades: consumir as mensagens publicadas no *broker*, guardar os dados na base de dados e disponibilizar informação para ser visualizada na aplicação *web* (cf. *website*).

A base de dados é relacional e recorre à tecnologia *PostgreSQL*. Serve para guardar todos os dados recolhidos nos nós de monitorização, os dados referentes aos dispositivos *Edge* e *Fog*, bem como dos utilizadores registados para se autenticarem e cederem ao *website*.

Por fim, a aplicação *web* utiliza a tecnologia *React* e funciona como um *dashboard* para visualização de toda a informação do sistema. Permite, por exemplo, consultar as informações de todas as *gateways*, como também dos nós de monitorização associados. Permite ainda consultar os dados de todos os nós de monitorização e exportar estes dados para um ficheiro csv. Disponibiliza também uma página *web* de alertas referente, por exemplo, a possíveis cheias ou níveis baixos de bateria dos nós.

Todos os componentes da *Cloud* funcionam num servidor da AWS e baseiam-se em *containers Docker* que permitem uma maior facilidade na sua criação e gestão.

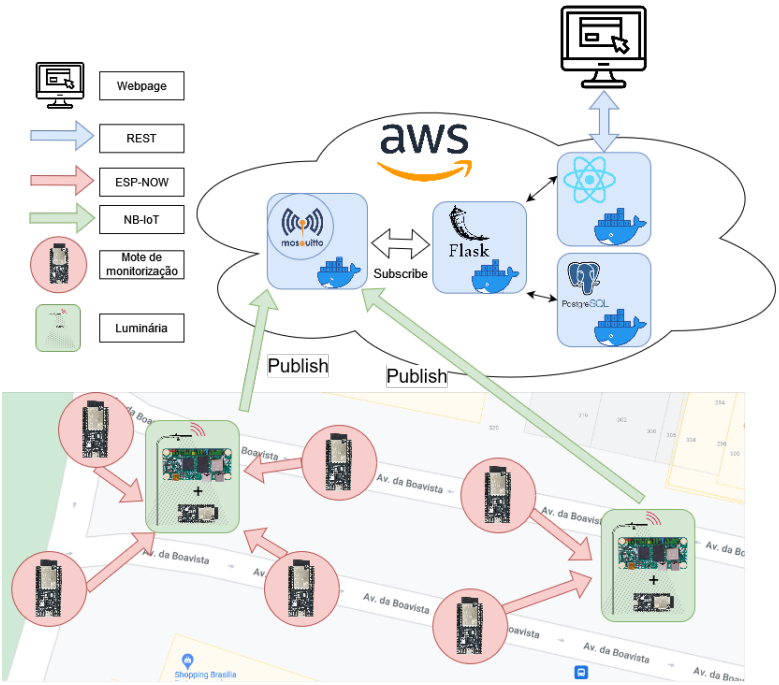

Figura 3.3: Arquitetura do Sistema

Fonte: Própria

3.5 Validação dos Sensores Seleccionados

O processo de revisão de possíveis sensores a utilizar no *Smart-city Drainage System*, permitiu identificar algumas alternativas que melhor se coadunavem aos requisitos do sistema identificado anteriormente, por forma a permitir a monitorização das caixas coletoras e de drenagem. Este processo de revisão baseou-se em serviços *online* especializados na venda de sensores, em trabalhos relacionados e em sensores pré-analisados pelas equipas da empresa Hardlevel.

Os requisitos para a escolha destes sensores foram: ter certificação IP67 ou IP68, alta resistência a condições adversas, serem de baixo consumo, terem baixo custo e terem uma boa precisão nas suas leituras.

3.5.1 Análise do sensor de fluxo de líquidos

Para aferir a eficácia deste sensor, foram efectuados testes em laboratório com ambientes controlados. Inicialmente o sensor foi calibrado para a leitura de fluxos de água, uma vez que o sensor permite leituras de diversos líquidos com diferentes densidades. Foi portanto necessário proceder à calibração para o líquido que queríamos medir.

Executaram-se vários ensaios com diferentes quantidades de água. Definiu-se um *ground truth* baseado na medição de fluxos de água recorrendo a equipamento especial-

izado de laboratório (tais como gobelé, para medir a quantidade de líquido), para comparar com as leituras do sensor, permitindo assim aferir a precisão do sensor. Efetuaram-se testes ao sensor, recorrendo a diversos ensaios com diferentes quantidades de água (equivalentes ao *ground truth*) para comprovar a precisão do sensor.

Na Tabela 3.3, podemos ver os resultados de quatro ensaios feitos com o sensor de fluxo, onde se apresentam três cálculos: quantidade real, deteção do sensor e o erro. Todos as leituras foram registados com a unidade de medida mililitros.

Tabela 3.3: Resultados obtidos nos testes do sensor de fluxo

Ensaios	**Quantidade real (mL)**	**Deteção do sensor (mL)**	**Erro (mL)**	**Erro (%)**
1	500	578	78	13
2	1000	1036	36	4
3	1500	1480	20	1
4	2000	1938	72	3

A Figura 3.4 apresenta um gráfico que mostra os resultados das medidas obtidas pelo sensor em comparação com as medidas reais. No eixo das abcissas temos o número do ensaio, em que podemos ver que foram efetuados quatro ensaios. No eixo das ordenadas temos a quantidade em mililitros da leitura do sensor. Na linha azul temos o *ground truth* com as medidas reais, e na linha laranja as leituras obtidas pelo sensor.

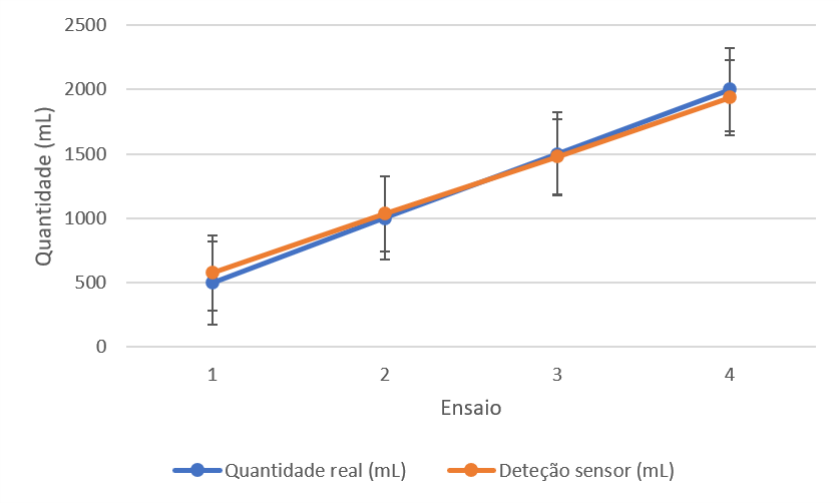

Figura 3.4: Gráfico comparativo entre as leituras do sensor de fluxo versus leituras reais

Fonte: Própria

Analisando a Tabela 3.3 e a Figura 3.4 podemos concluir que o sensor de fluxo oferece uma precisão assinalável, minimizando o erro na determinação do fluxo de água. As leituras deste sensor nos testes realizados demonstram que pode ser uma solução para determinar o fluxo de água que atravessa as caixas de drenagem.

Um dos principais problemas identificados com este sensor é a sua instabilidade quando em contacto com detritos. As pás móveis podem ficar obstruídas, alterando assim as leituras dos valores de fluxo, diminuindo assim a precisão das leituras. Como os bueiros e caixas coletoras são locais propícios a muitos detritos, este sensor pode sofrer alguma instabilidade nas suas leituras.

A utilização deste sensor pode também agravar as obstruções provocadas pelos detritos dentro do tubo, levando à obstrução da passagem da água e, consequentemente, à possibilidade de cheias. A difícil instalação deste sensor nos bueiros/caixas coletoras, foi outro dos factores que levou à sua exclusão para a utilização no sistema. Contudo o estudo deste sensor permitiu-nos ter uma melhor percepção sobre o seu modo de funcionamento e procurar uma solução alternativa para a leitura do fluxo dentro dos tubos de drenagem.

3.5.2 Análise ao sensor de pressão de água

Foram efetuados testes com o sensor de pressão para aferir a sua eficácia na determinação da quantidade de água a encher um bueiro. Para isso foi construído um protótipo de testes para este sensor. A Tabela 3.4 identifica as dimensões do protótipo do bueiro. Os testes efetuados desenrolaram-se com uma pressão de água constante. Recolheram-se leituras a cada segundo. A Figura 3.6 mostra a introdução de água no bueiro de testes.

Tabela 3.4: Dimensões do protótipo

	Dimensões (cm)
Comprimento	55,5
Largura	39
Altura	28,5

O sensor foi colocado na entrada do tubo que conecta ao bueiro, para depois fazer leituras de possíveis entupimentos. Também foi colocado um tampão para simular a obstrução no bueiro. O protótipo encontra-se visível na figura 3.5. Os resultados esperados deveriam demonstrar um aumento de pressão ao longo do tempo, devido a uma acumulação constante de água resultante do entupimento.

Figura 3.5: Protótipo de bueiro para medição da pressão da água em função do entupimento

Fonte: Própria

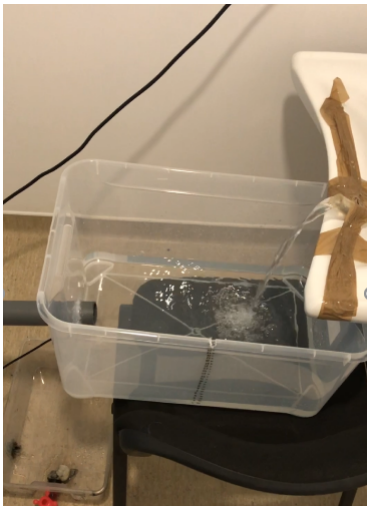

Figura 3.6: Testes no protótipo de bueiro com a introdução de água

Fonte: Própria

Como se vê na Figura 3.7, as leituras do sensor de pressão são muito instáveis. Mesmo desenhando uma linha de tendência podemos verificar que a partir desta não conseguimos tirar grandes conclusões. Com o bueiro cheio de água, o valor de pressão pouco difere dos registos com menos água. Estes resultados devem-se ao facto da granularidade das leituras de pressão do sensor ser muito baixa (entre 0-1.6 Mpa). Como o protótipo não tem espaço para grandes quantidades de água (tal como o bueiro), as leituras do sensor são instáveis e praticamente indiferenciáveis, por se tratarem de pequenos valores de pressão.

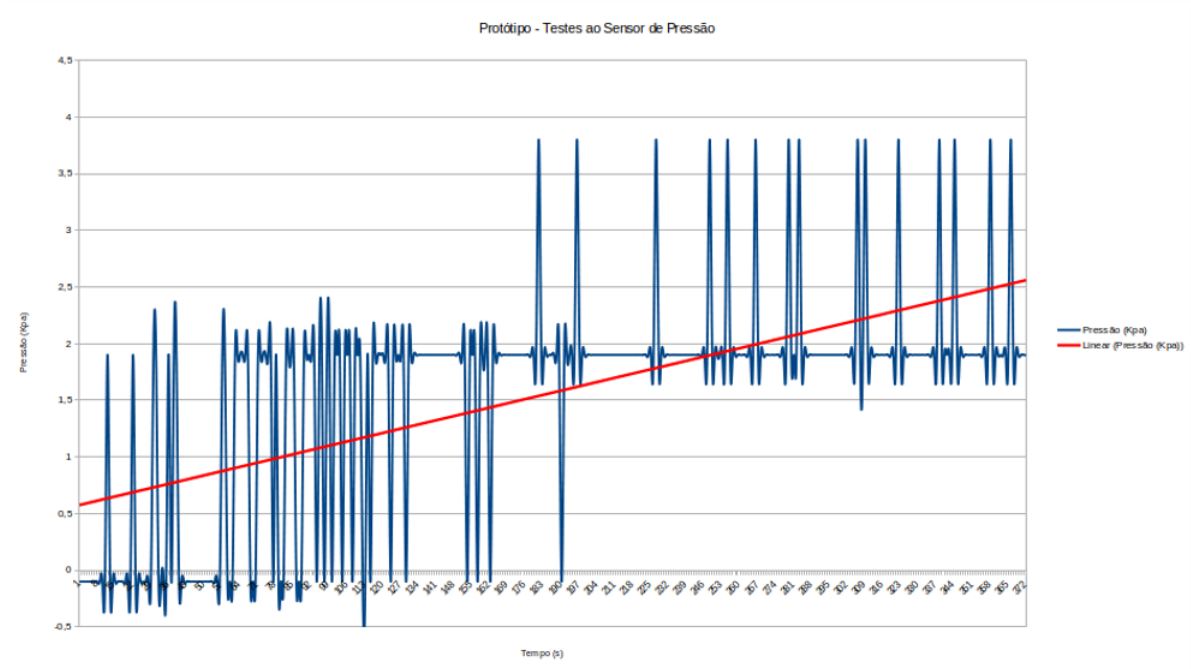

Figura 3.7: Gráfico de medidas do sensor de pressão

Fonte: Própria

Por se tratar de um sensor cujos resultados são bastante instáveis e inapropriados à pressão que pode existir num bueiro, decidiu-se afastar a utilização deste sensor no sistema de monitorização. As complicações de instalação deste sensor (tubo ou fundo do bueiro), contribuiram também para que não fosse utilizado.

3.5.3 Análise ao sensor de ultrassons

Para testar o sensor de ultrassons foi também utilizado o protótipo com as dimensões referidas na Tabela 3.4. Foi desenhada uma régua na parte lateral do protótipo, como mostra a Figura 3.8, utilizando uma régua real. Esta régua serve como referência para comparar distâncias à água medidas com o sensor. Este foi colocado na tampa do protótipo, virado para baixo como mostra a Figura 3.9. Para efetuar as leituras o sensor emite um sinal em direção à água, que é refletido e regressa ao sensor. O tempo de ida e volta permite calcular a distância a que o sensor se encontra da água, e portanto aferir o nível de água no bueiro.

A unidade de medida utilizada para registar a distância foi o centímetro (cm). As medidas foram efetuadas desde os 21cm até aos 4cm. A cada 1cm que se enchia de água, era efetuada uma leitura no sensor.

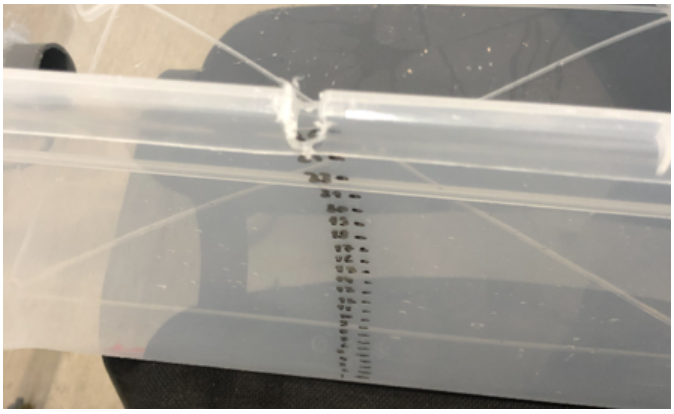

Figura 3.8: Régua no protótipo do bueiro

Fonte: Própria

Figura 3.9: Sensor de ultrassons montado no protótipo do bueiro

Fonte: Própria

Analisando os resultados apresentados na Figura 3.10, podemos verificar que o sensor de ultrassons oferece uma precisão extrema. Comparando as medidas obtidas pela régua (*ground truth*) com as medições efetuadas pelo sensor, podemos comprovar que o erro deste é mínimo ou nulo. Um dos objetivos deste teste era saber se a água comprometia as leituras do sensor. Analisando os resultados, podemos ver que isto não foi um problema.

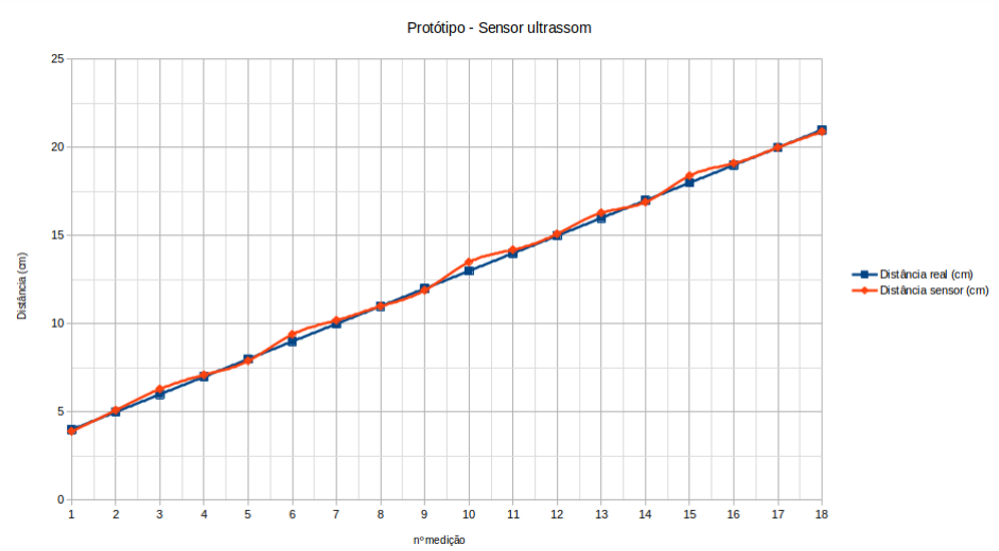

Figura 3.10: Gráfico de resultados do teste do sensor de ultrassons

Fonte: Própria

Concluindo, o sensor de ultrassons é um componente importante para ser usado no sistema. É um sensor com uma baixa taxa de erro, resistente à água e detritos e que pode ser facilmente instalado num bueiro. Com este sensor podemos detetar o nível de água ou lixo que se encontra no bueiro.

3.5.4 Análise ao sensor capacitivo sem contacto

Este sensor é binário, ou seja, deteta a presença ou ausência de um líquido condutor. Os testes efetuados recorreram a três amostras de material depositado no bueiro. Como mostra a Figura 3.11, estas amostras foram de areia húmida, água com terra e água apenas. As amostras serviram para simular diferentes tipos de depósitos dentro do bueiro, e para aferir se as leituras eram precisas, mesmo com detritos envolvidos.

Figura 3.11: Amostras de depósitos para teste

Fonte: Própria

Os testes foram efetuados encostando o sensor na parte de fora das amostras, registando as leituras em cada meio. Para uma maior precisão, foram efetuados vários ensaios para comprovar os resultados. Em cada ensaio, o sensor era colocado sob uma amostra diferente, registando-se de seguida os resultados das leituras do sensor.

Analisando os resultados registados na Tabela 3.5, podemos verificar que o sensor, estando em contacto com um líquido condutor, mesmo em pouca quantidades, consegue detetar a presença deste líquido nas amostras. Com estes resultados, podemos dizer que este sensor oferece uma elevada precisão na deteção de líquidos condutores.

Tabela 3.5: Resultados do teste do sensor capacitivo sem contacto

Ensaio	**Ar**	**Água**	**Água + Terra**	**Areia húmida**
1	0	1	1	1
2	0	1	1	1
3	0	1	1	1
4	0	1	1	1
5	0	1	1	1

Podemos concluir que o sensor capacitivo sem contacto é um sensor com uma elevada precisão nas suas leituras, sendo assim uma possível aposta para o sistema de monitorização. A desvantagem deste sensor é a dificuldade de instalação na caixa de drenagem/-coletora. Como este não é resistente à água, não pode ser instalado dentro destas caixas. Assim, a grande dificuldade da instalação nas caixas e a sua fraca resistência, levou à sua exclusão para como solução para o sistemação de monitorização.

3.5.5 Análise ao sensor fotoelétrico

Nos testes efetuados com o sensor fotoelétrico, foi usado o mesmo conjunto de amostras que foram usadas no sensor capacitivo sem contacto visíveis na Figura 3.11. Os testes foram efetuados da mesma forma, mas desta vez este sensor foi mergulhado em cada amostra. Efetuaram-se as leituras do sensor e os resultados foram registados.

Fazendo uma análise aos resultados visíveis na Tabela 3.6, permite-nos verificar que estes foram semelhantes aos do sensor capacitivo sem contacto. Ambos os sensores usam o princípio binário de deteção de líquidos condutores.

Tabela 3.6: Resultados do teste do sensor fotoelétrico

Ensaio	**Ar**	**Água**	**Água + Terra**	**Areia húmida**
1	0	1	1	1
2	0	1	1	1
3	0	1	1	1
4	0	1	1	1
5	0	1	1	1

Concluindo, o sensor fotoelétrico é um sensor com uma enorme precisão na deteção de líquidos condutores. É um sensor com resultados muito parecidos com o sensor capacitivo sem contacto, mas oferece uma maior resistência, facilidade de instalação e menor custo. Este sensor pode ser uma boa solução para a deteção do nível de água dentro de bueiros e caixas coletoras, trazendo diversas vantagens para o sistema.

3.5.6 Análise ao sensor capacitivo de humidade de solo

Os testes efetuados com o sensor capacitivo de humidade de solo, foram os mesmos que os aplicados aos sensores capacitivo sem contacto e fotoelétrico (ver Figura 3.11). Da mesma forma que o sensor fotoelétrico foi mergulhado nas amostras, este sensor capacitivo foi mergulhado nas amostras de depósitos, durante cinco segundos, e de seguida registaram-se os valores.

Analisando a Tabela 3.7 e o gráfico da Figura 3.12, podemos tirar algumas conclusões acerca deste sensor. É importante salientar que este sensor é muito estável, ou seja, no momento em que se introduzia na amostra, este estabilizava na leitura correspondente, sem variações nos valores apresentados. De acordo com os resultados podemos dizer que este sensor consegue identificar com precisão a presença de água, usando o valor da capacitância. Contudo, entre amostras diferentes que envolvem água com terra ou areia, pudemos verificar que a sua diferenciação é difícil.

Todos os valores lidos pelo sensor, corresponderam à tabela de valores deste, o que demonstra uma grande precisão. Por exemplo, o valor para uma amostra sem a presença de água rondou os 513-514. Consultando a tabela podemos ver que estamos perante um

ambiente seco. Os valores obtidos numa amostra com água diminuem, pois apresentam menor capacitância resultante do ambiente mais húmido.

Tabela 3.7: Resultados do teste ao sensor capacitivo de humidade de solo

Ensaio	Ar	Água	Água + terra	Areia húmida
1	513	262	253	284
2	513	262	253	284
3	514	262	253	285
4	514	261	253	286
5	513	261	253	286

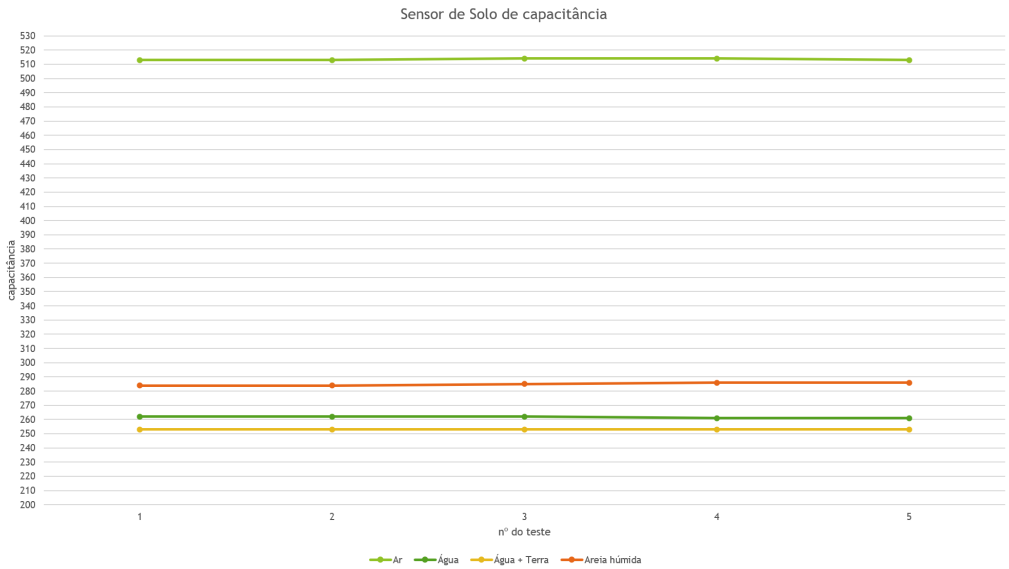

Figura 3.12: Gráfico do teste do sensor capacitivo de humidade de solo

Fonte: Própria

Concluindo, tratando-se de um sensor que é resistente à água, de fácil instalação e com resultados consistentes nos testes, pode ser considerado para utilização no sistema de monitorização. É um sensor que permite a deteção do nível de água dentro do bueiro/-caixa coletora. Tem a vantagem de permitir a deteção da quantidade de partículas quando mergulhado em água. Isto pode dizer-nos se naquele local existe muita passagem de detritos ou não. Este sensor tem a desvantagem, em relação ao sensor fotoelétrico, de ter um preço bastante mais elevado, indo contra um dos requisitos do sistema.

3.5.7 Análise ao sensor de TDS

As amostras referidas na Figura 3.11 serviram também para os testes do sensor de TDS (*Total Dissolved Solids*). Este sensor deteta o nível de partículas em ppm (partes por

milhão) do líquido para o qual se está a efetuar a leitura. O método utilizado foi o mesmo que no sensor capacitivo de humidade solo. Também este sensor possui uma escala (ver Figura 2.14) na qual cada valor de saída do sensor representa uma grandeza da qualidade da água. Quanto mais partículas forem detetadas no sensor, em pior estado se encontra a água.

Analisando os resultados da Tabela 3.8 e o gráfico da Figura 3.13, podemos ver que, tal como o sensor capacitivo de humidade de solo, a distinção entre o contato com o líquido e o ar é diferenciada. A particularidade e vantagem deste sensor está na distinção entre as amostras. No ar o valor é de 0 ppm, pois não está em contacto com nenhum líquido condutor. Na água o valor é de 104 ppm, ou seja, consultando a tabela, vemos que é *Hard water*. Como foi usada água de torneira, este valor é aceitável dentro dos parâmetros de leitura do sensor. Na amostra de água com terra, vemos que o valor sobe consideravelmente. A terra faz com que a água fique mais impura devido às suas partículas. Na areia húmida este valor desce consideravelmente para valores de água normal. Isto deve-se ao facto de a areia ser um agente de filtragem e o tamanho das partículas ser de maior dimensão.

Tabela 3.8: Resultados do teste ao sensor de TDS

Ensaio	**Ar**	**Água**	**Água + terra**	**Areia húmida**
1	0	104	166	31
2	0	104	166	33
3	0	104	162	35
4	0	104	164	37
5	0	104	150	37

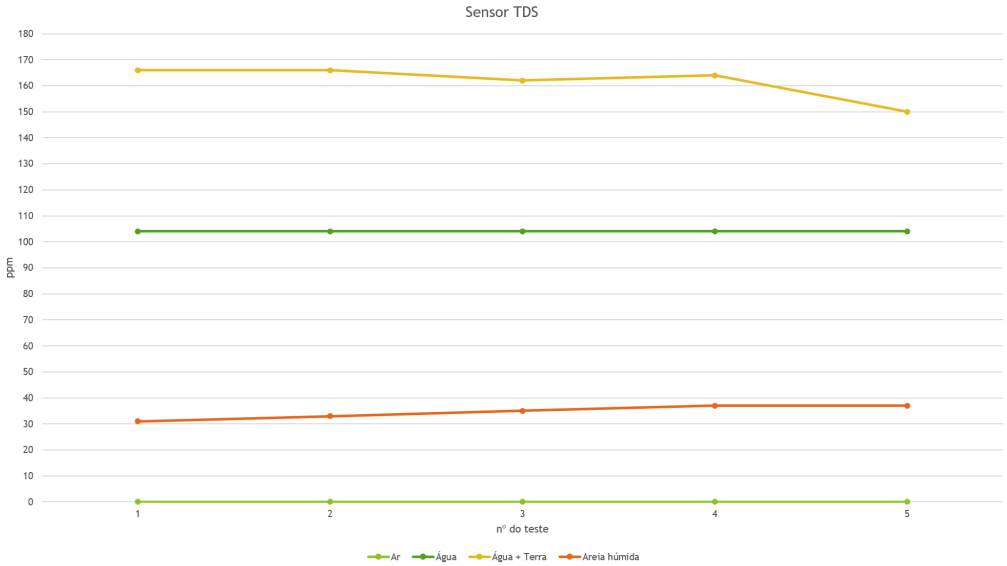

Figura 3.13: Gráfico do teste ao sensor de TDS

Fonte: Própria

Podemos concluir que o sensor de TDS (*Total Dissolved Solids*) é um possível candidato a ser usado no sistema. Para além dos seus resultados terem sido com muita precisão, é um sensor que acresce algumas vantagens comparando com o sensor capacitivo de humidade de solo. Uma destas vantagens é o facto de permitir verificar se estamos perante uma descarga ilegal sobre o sistema de drenagem. Este é um sensor de baixo custo, baixo consumo, fácil instalação no sistema e resistente à água e detritos. Trata-se de um sensor com resultados muito promissores. Para o caso de uso em questão, o sensor fotoelétrico é uma melhor escolha para ser utilizado no sistema devido ao seu menor custo. Como o foco desta dissertação não são as descargas ilegais, decidimos descartar este sensor.

3.5.8 Comparação dos sensores

A Tabela 3.9 resume os resultados de todos os sensores estudados e testados. Analisando a tabela podemos ter uma percepção de quais sensores podem ser usados no sistema e quais devem ser descartados.

Dentro desta gama de sensores temos dois que se destacam, o sensor de ultrassons e o sensor fotoelétrico. Com o sensor de ultrassons temos vantagens relacionadas com a sua elevada precisão e resistência. O sensor fotoelétrico presenta também várias vantagens relacionadas com o baixo custo, resistência e alta precisão.

Com o sensor de ultrassons podemos resolver o problema da deteção de lixo e água depositados no bueiro/caixa coletora. O sensor fotoelétrico permite detetar se está a ocor-

rer alguma cheia. Caso o sensor de ultrassons esteja bloqueado por algum lixo acumulado, o sensor fotoelétrico oferece também uma solução de recurso.

Tabela 3.9: Comparação dos sensores analisados

Sensor	Custo	Consumo	Resistência	Eficácia	Problemas
Fluxo de líquido	Baixo	Baixo	Internamente resistente à água/no exterior não é muito resistente	Alta em líquido sem detritos Péssima em contacto com detritos	Em contacto com detritos pode provocar obstruções no tubo de drenagem Difícil instalação
Pressão de água	Alto	Alto	Resistente a água e detritos	Muito instável a baixas pressões	Baixo nível de precisão devido ao espaço de leituras do sensor ser elevado
Ultrassons	Baixo	Baixo	Resistente a água e detritos	Alta precisão em leituras de lixo e nível de água/baixo blindzone	Certos tipos de lixo (folhas) podem obstruir as leituras do sensor
Capacitivo sem contacto	Baixo	Baixo	Pouco resistente, apenas pode ser utilizado fora da caixa	Alta precisão para detetar água através de obstáculos	Difícil instalação Fraca resistência a condições adversas

Fotoelétrico	Baixo	Baixo	Ponta do sensor é resistente a água, detrito, corrosão, pressão e altas temperaturas	Alta precisão na deteção da presença de líquido	Instalação na caixa de drenagem
Capacitivo de humidade do solo	Alto	Baixo	Resistente a àgua, detritos e anticorrosivo	Fraca precisão na diferenciação de diferentes soluções	Preço mais elevado que o resto da família dos sensores Não diferencia lixo seco do ar
TDS	Baixo	Baixo	Resistente a àgua, detritos e anticorrosivo	Alta precisão para saber a qualidade da água	Com o tempo as agulhas do sensor podem enferrujar Não diferencia lixo seco do ar

3.6 Validação de Tecnologias de Comunicação

Um dos pontos importantes para o correto funcionamento do sistema, é como os nós vão comunicar com a *Gateway*, e como esta vai comunicar para a *Cloud*. Como os nós de monitorização não vão ter acesso a corrente elétrica permanente, irá ser necessário encontrar uma forma de comunicação que seja de baixo consumo. Como a *Gateway* irá estar ligada constantemente à energia elétrica, através das luminárias, isto não será um problema. Os nós de monitorização vão estar no subsolo, logo será necessário a utilização de uma tecnologia que garanta cobertura e comunicações sem falhas.

Para minimizar o consumo de energia, foi previamente determinado que as comunicações não seriam contínuas em cada ciclo *deepsleep*. O nó de monitorização vai recolher dados diversas vezes, e só depois de um certo número de leituras vai fazer a comunicação

para a *Gateway*. Esta também só sincroniza os dados com a *Cloud* de forma periódica.

Foram identificadas diversas tecnologias de comunicação com potencial de utilização nos cenários de monitorização do nosso sistema. As secções abaixo analisam o potencial de utilização destas tecnologias.

3.6.1 GPRS

A primeira opção incidiu sobre a tecnologia GPRS. O estudo desta tecnologia deveu-se ao facto da empresa Hardlevel a utilizar já em diversos dos seus sistemas. Foram efetuados vários testes pelas equipas da Hardlevel, demonstrando que esta tecnologia é adequada para as comunicações entre os dispositivos *Fog* e a *Cloud*. Foi importante considerá-la particularmente para o envio de dados entre a *Gateway* e os serviços de *backend* na *Cloud*.

O módulo de comunicação escolhido foi o SIM800L. Este módulo apesar de ser *Power Hungry* para um sistema IoT em que o principal foco é a poupança de energia, é usado na empresa Hardlevel. Os testes efetuados demonstraram ser capaz de efetuar as comunicações sem problema. Sendo um módulo *Power Hungry*, podemos dizer que não é uma boa solução para os nós de monitorização. Contudo, este módulo é adequado para a *Gateway*, visto que esta está constantemente ligada à rede elétrica. Este módulo irá efetuar a conexão à Internet, via rede celular, e permitir estabelecer uma ligação com o *broker* MQTT para o envio de informação.

As suas principais vantagens são a facilidade de instalação, elevada taxa de fiabilidade nas comunicações e rapidez de envio. As suas desvantagens estão relacionadas com o consumo energético, pelo facto de oferecer uma tecnologia 2.5G datada (existindo actualmente opções 4G e 5G) e possuir alguns problemas nos envios se a rede estiver muito congestionada.

Figura 3.14: Módulo de comunicação SIM800l

Fonte: `https://www.makerfabs.com/sim800l-minimum-system-gprs-gsm.html`

A tecnologia GPRS é utilizada num projeto anterior de monitorização de resíduos urbanos (Kanade et al., 2021), cujo objectivo é efetuar o envio da informação dos sensores

para a *Cloud* através da rede celular. Neste projeto, que decorre há vários anos, tem demonstrado ser bastante eficaz e fiável nas comunicações.

3.6.2 NB-IoT

O módulo SIM800L apresenta algumas desvantagens, relacionadas com o consumo, pelo facto de ser uma tecnologia celular mais antiga (2.5G) e sofrer de problemas com o congestionamento de rede celular. Procurou-se portanto uma nova tecnologia de comunicações para o envio dos dados para a *Cloud*, uma vez que na Hardlevel já se considerava a utilização de NB-IoT, desenvolvendo-se alguns testes para o efeito.

O NB-IoT possui vantagens tais como ser de baixo custo, ter um baixo consumo energético, ser de fácil instalação, não sofrer dos problemas da congestionamento de rede, ter uma cobertura de longo alcance, baixa latência, funcionar em redes desde do 2G até ao 5G e oferecer uma elevada taxa de transmissão de dados. Trata-se de uma tecnologia recente, direcionada para os sistemas IoT e com uma comunidade bastante ativa.

O módulo escolhido para testes foi o Quectel BG95 M3 (ver Figura 3.15), uma vez que já estava a ser utilizado em testes na empresa Hardlevel. Este módulo já se encontra incorporado numa placa PCB (*Printed circuit board*) própria para ser integrado em sistemas IoT desenvolvidos na empresa. O facto de já se terem efetuado testes na empresa com este módulo, influenciou a escolha desta tecnologia.

Figura 3.15: Módulo de comunicação Quectel BG95 M3

Fonte: `https://etmiot.com/products/cellular-oem-modules/etm-adb-bg95-cat-m1-nb2-module/`

3.6.3 MQTT

Existiu a necessidade de encontrar um protocolo de comunicação eficiente, fiável e assíncrono para efetuar o envio dos dados da *Gateway* para a *Cloud*. O protocolo escolhido foi o MQTT, devido a ser um protocolo de comunicação leve e desenhado para dispositivos IoT.

É um protocolo de mensagens leve para dispositivos IoT que está otimizado para redes TCP/IP. Oferece um modo de funcionamento *Publish/Subscriber*, baseado num *bro-*

ker (intermediário) para facilitar a troca de mensagens entre *Publishers* e *Subscribers*. Neste sistema o *Publisher* será a *Gateway* irá sincronizar com os dados da base de dsdos local com o *broker* hospedado na AWS. O *Subscriber* irá ser o serviço de *backend* também hospedado na AWS. O serviço irá consumir as mensagens enviadas pela *Gateway* e guardar os dados na base de dados da *Cloud*.

3.6.4 ESP-NOW

Para a comunicação entre os nós de monitorização e a *Gateway* foi considerada a tecnologia ESP-NOW. Esta tecnologia tem a grande vantagem, em relação a outras tecnologias (e.g. LoRa), de não precisar de hardware externo que faça a tradução para o protocolo TCP/IP. Isto faz com que se reduzam consideravelmente os custos do sistema, uma vez que esta tecnologia já se encontra implementada em quase todos os microcontroladores ESP32.

A tecnologia ESP-NOW possui várias vantagens, tais como:

- Disponível nos microcontroladores ESP32: oferece um solução embebida de menor custo porque não precisa de qualquer módulo extra para as comunicações, como é o caso das tecnologias baseadas em Arduino e afins;
- Baixo consumo: oferece um menor consumo aos nós de monitorização em comparação com tecnolgias similares (e.g. WiFi);
- Facilidade de implementação: os dados podem ser transmitidos diretamente entre dispositivos similares usando apenas os endereços MAC ou por broadcast;
- Cobertura considerável: distância de transmissão elevada (320m a 500m) em comparação com tecnolgias similares;
- Disponibilidade de comunicação em mesh: os nós ESP-NOW podem encaminhar pacotes entre eles para atingirem a *Gateway*.

A tecnologia ESP-NOW permite a comunicação directa sem fios entre múltiplos dispositivos ESP32. Utiliza uma conexão muito parecida com a comunicação sem fios *low power* 2.4GHz. Para efetuar o envio é necessário saber o endereço MAC (*Media Access Control*) do recetor para o emparelhamento. Depois do emparelhamento a conexão é segura e ponto-a-ponto, sem ser necessário o *handshake*. Outra vantagem é que podemos ter uma conexão unidirecional ou bidirecional.

A desvantagem deste protocolo é o seu menor alcance, em comparação ao LoRa por exemplo, e o *payload* máximo ser de apenas 250 *bytes*. Para contornar esta limitação foi considerada para este sistema uma topologia em estrela (ver Figura 3.16). Cada nó de monitorização (*Master*) irá comunicar diretamente com a *Gateway* (*Slave*). A topologia

em estrela também resolve o problema de sincronização porque a *Gateway* irá estar sempre ligada à energia, logo os nós podem acordar a qualquer momento, enviar os dados, confirmar a receção com êxito e voltar a adormecer.

Figura 3.16: Comunicação masters-slave via ESP-NOW

Fonte: `https://randomnerdtutorials.com/esp-now-esp32-arduino-ide/`

3.7 Conclusão

Neste capítulo foi possível especificar todo o *Smart-city Drainage System*. Começou-se por abordar a estrutura dos sistemas de drenagem de águas pluviais que encontrámos em Portugal. Descreveu-se como todo este sistema está implementado, os seus componentes e como todo o processo de drenagem funciona. Nos cenários de aplicação do sistema de monitorização, esplanaram-se os problemas e soluções que este sistema poderia resolver.

Posteriormente, introduziram-se os requisitos do sistema de monitorização. Descreveram-se os requisitos funcionais e não funcionais específicos dos vários componentes ou camadas do sistemas (cf. nós de monitorização, *Gateway* e *Cloud*).

Na arquitetura do sistema descreveu-se como iriam organizar-se os componentes de todo o sistema, tanto ao nível de hardware, como software e comunicações. Os nós de monitorização irão estar colocados na via pública dentro de caixas de drenagem e caixas coletoras. Estes nós irão ser compostos por sensores, baterias e pelo microcontrolador. Este tem a função de processar todos os dados dos sensores e enviar via ESP-NOW para a *Gateway*. As luminárias irão alojar as *Gateways* com alimentação permanente. As *Gateway* serão compostas por um microcontrolador ESP32 que irá receber os dados dos nós de monitorização via ESP-NOW (cf. secção 3.6.4). O ESP32 envia depois os dados via porta série para a Raspberry PI e esta irá guardar e sincronizar com a *Cloud*. Esta sincronização será feita via MQTT (cf. secção 3.6.3) e irá utilizar a tecnologia de comunicação NB-IoT (cf. seccção 3.6.2). A escolha da infraestrutura da *Cloud* incidiu sobre a AWS. É

uma plataforma de fácil utilização, que permite a criação de máquinas virtuais de forma gratuita. Esta máquina funcionar com vários *containers* para o servidor, base de dados, *webpage* e *broker*. A divisão de containers *docker* permite que esta implementação não tenha um *single point of failure*. Por exemplo, caso o container da *webpage* falhe, todo o sistema iria continuar a funcionar normalmente. A *framework* escolhida para o servidor foi o *Flask*. Esta *framework* para desenvolvimento em *Python* é de fácil aprendizagem e rápida. A base de dados escolhida foi o *PostgresSQL*, que é também de fácil aprendizagem e eficiente no armazenamento de dados. A tecnologia escolhida para a *webpage* foi o *React*. Este é conhecido pela sua rápida aprendizagem e pelo seu desempenho. A grande quantidade de bibliotecas disponíveis, também foi outro fator a considerar para a utilização desta *framework*.

Os testes dos sensores identificados e descritos no capítulo anterior, permitiram aferir os sensores que poderiam ser utilizados no sistema de monitorização. O sensor de fluxo de líquidos teve bons resultados, mas não poderia ser considerado por existir grande probabilidade de as pás deste sensor bloquearem o tubo de drenagem. Os resultados dos testes do sensor de pressão de água não foram muito animadores, uma vez que o sensor, quando sujeito a pressões muito baixas, tinha resultados inconstantes. O sensor de ultrassons teve bons resultados nos testes efetuados. Este sensor teve valores de leitura muito próximos ou equivalentes à referência obtida com a régua. Por ser também um sensor muito resistente a condições adversas, foi escolhido para o sistema. O sensor capacitivo sem contacto demonstrou bons resultados nos testes efetuados, mas a sua difícil instalação nas caixas de drenagem e caixas coletoras levou a descartar a sua utilização. O sensor fotoelétrico, com a mesma finalidade do sensor capacitivo sem contacto, teve bons resultados. Este sensor ao contrário do capacitivo, tem uma maior facilidade de instalação e é mais resistente, foi por isso considerado para o sistema. O sensor capacitivo de humidade de solo também apresentou bons resultados, mas o seu preço mais elevado eliminou a sua utilização. Por último, foi testado o sensor de TDS. Este sensor para além de ter tido ótimos resultados, acrescenta a possibilidade de deteção de descargas ilegais. Este não foi escolhido para o sistema pelo facto de ter um preço mais elevado que os restantes nesta categoria. Como a deteção de descargas ilegais não era o foco desta dissertação, foi também um fator para este não ser escolhido.

Por fim, foi efectuado um estudo das tecnologias e protocolos de comunicação adequados à utilização no sistema de monitorização. A tecnologia de comunicação escolhida, para efetuar comunicações entre a *Gateway* e a *Cloud* foi o NB-IoT. Esta tecnologia tem vantagens de suportar todas as comunicações celulares disponíveis e ser de baixo consumo. Para protocolo de comunicação foi escolhido o MQTT. Este é um protocolo leve, desenhado para sistemas IoT, de baixo consumo e que demonstra ser bastante eficiente . Para as comunicações entre os nós de monitorização e a *Gateway*, escolheu-se a tecnologia ESP-NOW, que não requer módulos de hardware extra associados aos microcon-

troladores ESP32. Esta solução fica a um preço bastante reduzido em comparação com outras tecnologias de comunicação (e.g. LoRA). O ESP-NOW apresenta baixo consumo energético e demonstra ser capaz de assegurar comunicações de curto/médio alcance. O ESP-NOW também permite comunicações *mesh*, ou seja, podemos utilizar comunicações entre nós de monitorização, no encaminhamento de pacotes para a *Gateway*.

Chapter 4

Implementação do Smart-city Drainage System

Este capítulo aborda a implementação do *Smart-city Drainage System*. Descreve-se todo o processo envolvido na integração dos componentes de hardware para os nós de monitorização e *gateways*, com respectiva montagem e programação do protótipo. Procura-se também justificar as tecnologias utilizadas na implementação dos componentes, mais especificamente no servidor, *broker*, base de dados e *website*. Fornecem-se ainda detalhes da lógica por detrás de todo o software desenvolvido.

4.1 Sensores Utilizados na Monitorização

Inicialmente, foi necessário realizar um estudo sobre quais os sensores disponíveis no mercado que pudessem ser úteis às necessidades de monitorização do nosso sistema. Na análise dos sensores existentes, foram tidos em consideração diversos aspectos, mas em particular o seu preço, adequação física às condições extremas do âmbiente onde iriam ser instalados e grandesas medidas que pudessem ser úteis à monitorização de condutas e caixas de recolha de águas pluviais.

Foi considerada a utilização do sensor de ultrassons (cf. seccção 2.2.4.3) que poderia permitir medir o nível de lixo que se encontra nos bueiros/caixas de drenagem. Tratando-se de um sensor de baixo custo, baixo consumo, de fácil instalação e resistente à água e detritos, poderia ser uma boa escolha para o sistema. Este sensor permite portanto registar o nível de lixo dos bueiros, ao longo do tempo, e permiti assim fazer uma avaliação histórica das quantidades de lixo para planear a manutenção atempada das caixas de drenagem.

Outro sensor também considerado para o sistema foi o sensor fotoelétrico (cf. seccção 2.2.4.5). Este sensor permite aferir o nível de água (em fluxo ou residual) que se entra ou passa nos bueiros/caixas de drenagem. Podemos cruzar informação deste sensor com o

sensor de ultrassons para melhor aferir a quantidade de água no bueiro/caixa de drenagem, e assim avisar a autarquia de um possível problema naquele local. Sendo um sensor com características de preço, facilidade de instalação e resistência à água, semelhantes ao sensor de ultrassons, fazem dele outra boa escolha para utilização no sistema proposto.

4.2 Tecnologias de Comunicação

Um dos aspetos mais importantes para o *Smart-city Drainage System* são as tecnologias de comunicação sem fios usadas entre os nós de monitorização, a *Gateway* e a *Cloud*. Decidiu-se utilizar a tecnologia de comunicação sem fios ESP-NOW entre os nós e a *Gateway*, bem como a tecnologia NB-IoT, juntamente com o protocolo MQTT, entre a *Gateway* e a *Cloud*.

O ESP-NOW permite a comunicação entre os nós de monitorização e a *Gateway* e também entre os próprios nós. Resumidamente, esta tecnologia foi escolhida devido a estar em quase todos os microcontroladores ESP32, ter baixo consumo, facilidade de implementação, cobertura considerável e disponibilidade da comunicação em mesh.

Para a comunicação física entre a *Gateway* e a *Cloud*, a tecnologia de comunicação escolhida foi o NB-IoT. Trata-se de uma tecnologia que suporta GPRS, GSM e LTE e tem vindo a ser atualizada. Esta tecnologia oferece baixa latência e não está tão dependente do congestionamento de tráfego na rede celular. O módulo de hardware utilizado é o Quectel BG95 M3 e oferece um baixo consumo quando comparado, por exemplo, com outros módulos GPRS.

Sobre a ligação física NB-IoT utilizou-se o protocolo MQTT para efetuar trocas de mensagens assíncronas entre a *Gateway* e a *Cloud*. Este protocolo é vulgarmente utilizado em sistemas IoT devido à sua eficiência e versatilidade nos modos de operação disponibilizados.

4.3 Montagem do Protótipo

Nesta secção aborda-se o procedimento de montagem do protótipo, nomeadamente os componentes de hardware da *gateway* e dos nós de monitorização.

As caixas onde se montaram os componentes de hardware para o nó e para a *gateway*, não ofereciam proteção IP68. Isto porque na altura não existiam em stock caixas IP68. Contudo, uma vez que se tratava de um protótipo, o tipo de caixa utilizado não iria ser um problema. Na solução final seria necessária a utilização de caixas IP68 para proteger os componentes de hardware.

O protótipo do *Gateway* possuía uma WiPy(ESP32), uma Raspberry PI 4 e um módulo de comunicação Quectel BG95-M3. Estes componentes de hardware foram soldados

numa *Printed circuit board* (PCB) para facilitar a organização das ligações. Foi também criada uma ligação via porta série entre a WiPy e a Raspberry Pi 4, para o envio dos dados recebidos via ESP-NOW na WiPy. A escolha destes componentes de hardware (WiPy e Raspberry) deveu-se à sua versatilidade e por existirem em stock.

Num produto final, poder-se-ia trocar a WiPy por um microcontrolador ESP32 com menor consumo e em vez de uma Raspberry Pi 4, podia-se utilizar uma Raspberry Pi Zero com menor consumo e melhor aproveitamento do espaço. Na Figura 4.1 vêem-se duas imagens da mesma *Gateway*, uma com a WiPy e a Raspberry afastadas e outra com as placas empilhadas na sua organização final.

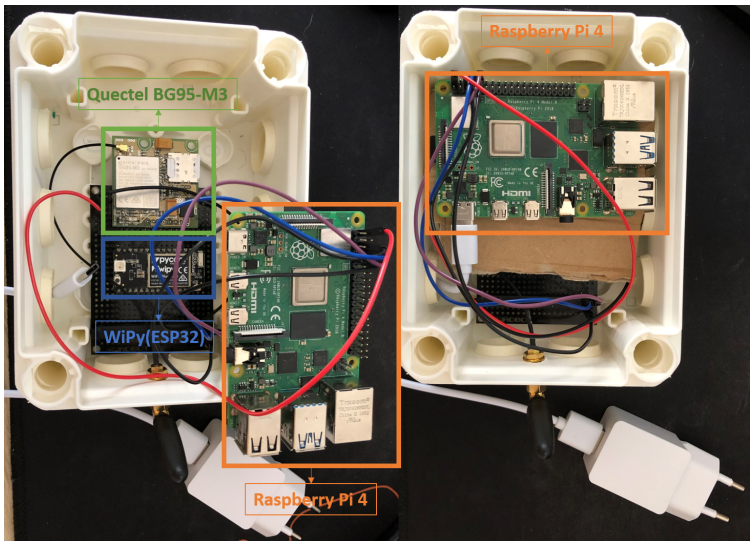

Figura 4.1: Montagem da Gateway contendo a WiPy e Raspberry PI

Fonte: Própria

Foram também criados dois protótipos dos nós de monitorização (ver Figura 4.2). Cada um destes possuía uma WiPy, duas baterias de 3.7v de 2100mah ligadas em série e dois sensores, um sensor de ultrassons e um sensor fotoelétrico. Foi usada uma WiPy devido à mesma situação da gateway, devido ao stock. Também seria possível aumentar o comprimento dos fios de ambos os sensores, mas para efeitos de protótipo deixou-se estar os fios com o comprimento de raiz. O sistema foi montado com apenas duas pilhas porque no momento da sua montagem não havia beterias disponíveis no mercado. Como existiam apenas quatro pilhas disponíveis, foram utilizadas duas em cada nó.

As duas pilhas foram ligadas em série para aumentar a capacidade do pack. Tal como na *gateway* todos os componentes foram ligados a uma *Printed circuit board* (PCB) para facilitar a montagem do hardware, incluindo os sensores.

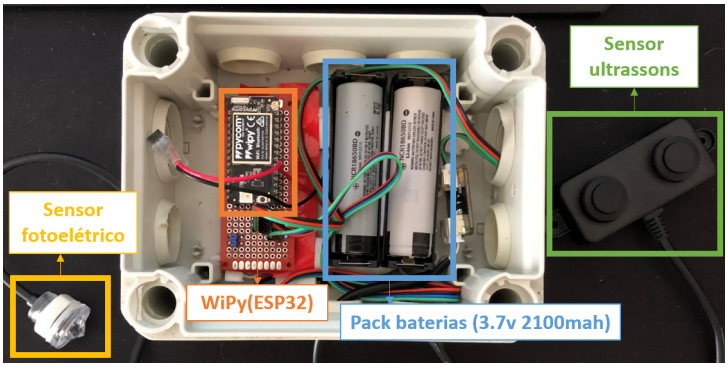

Figura 4.2: Montagem do Nó de Monitorização - *Edge device*

Fonte: Própria

4.4 Desenvolvimento do Software

Foi necessário o desenvolvimento de software para os nós de monitorização (WiPy), *gateway* (Raspberry Pi 4), servidor (*Python*) e *webpage* (*React*). Nos subcapítulos seguintes vamos demonstrar como esta implementação foi feita.

4.4.1 Microcontroladores e Gateway

O desenvolvimento de software para o microcontrolador WiPy (usado nos nós de monitorização), foi concebido para efetuar a leitura e envio da informação para a *gateway*, via ESP-NOW.

Na Figura 4.3, podemos ver o fluxograma do funcionamento dos nós de monitorização. A lógica geral do modo de funcionamento destes nós, foi implementada com o objectivo de poupança energética. Assim, definiram-se ciclos para periodicamente acordar/adormecer o sistema de modo a permitir uma maior autonomia e vida útil da bateria. Sendo a autonomia energética um dos principais objectivos a atingir, existiu a necessidade de o sistema apenas estar acordado quando necessário. Num ciclo de actividade era efectuada a leitura dos sensores e o envio da informação para *gateway*. Definiram-se ciclos de actividade alternativos, nos quais o sistema efetua várias leituras e só posteriormente, num dos ciclos, efetuava o envio para a *gateway*. Esta forma de funcionamento permitiu aumentar ainda mais a poupança de bateria e a autonomia dos nós.

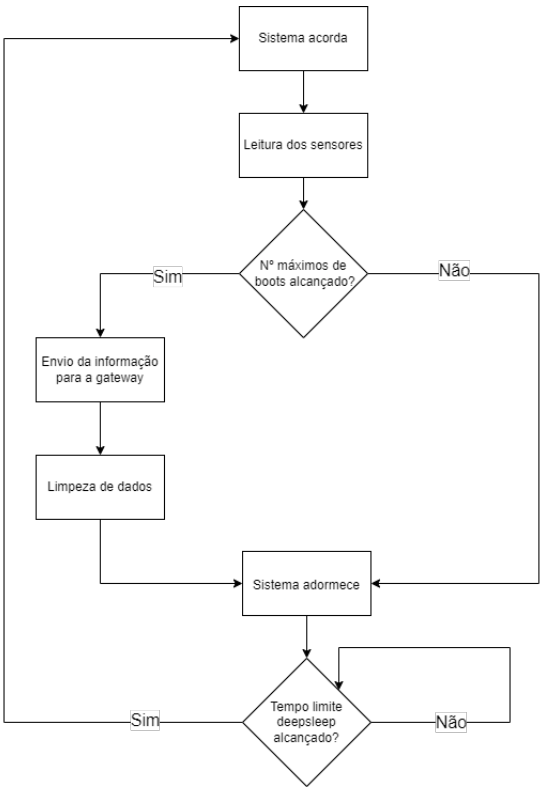

Figura 4.3: Fluxograma de funcionamento dos nós de monitorização

Fonte: Própria

O nó de monitorização possui inicialmente duas variáveis que definem de quanto em quanto tempo o sistema acorda, e o número de vezes até efetuar o envio para a *gateway*. Possui um id gerado no registo do nó de monitorização no *website* e um endereço mac da *gateway* para onde enviar os dados via ESP-NOW. Finalmente, possui também quatro variáveis RTC (*Real Time Clock*): o número de inicialização e três *arrays* de dados, bateria, sensor ultrassons e sensor fotoelétrico. Estas variáveis RTC (*Real Time Clock*) permitem que, mesmo que o sistema adormeça, os dados lidos até ao momento não sejam perdidos.

O *code snippet* apresentado abaixo é do nó de monitorização. Pode-se ver as variáveis RTC (*Real Time Clock*) utilizadas. Também está apresentado um método geral do modo de funcionamento do nó. Este método apresenta a leitura dos sensores, inserção dos dados nos *arrays* e incremento do número do *boot*. Também possui uma verificação que, caso esteja no *boot* limite configurado, empacota os dados e envia para a *gateway*.

```
// variables to save on sleep
RTC_DATA_ATTR int bootNumber = 0;
RTC_DATA_ATTR double ultrasound[MAX_BOOTN];
RTC_DATA_ATTR int photoeletric[MAX_BOOTN];
RTC_DATA_ATTR int battery[MAX_BOOTN];
```

```
void read_and_send_data()
{
  Serial.println("Reading sensors...");
  readSensorsData();
  insert_data();
  if (bootNumber == MAX_BOOTN - 1)
  {
    Serial.println("Sending to Master...");
    setupAndSendData();
  }
  else
  {
    ++bootNumber;
  }
}
```

Também foi desenvolvido código para a Raspberry Pi 4 (*gateway*) para efetuar a receção dos dados dos nós de monitorização e efetuar análise, tratamento e o seu envio para a *Cloud*. Foi criada uma API (*Python Flask*) e uma base de dados (*PostgresSQL*), para ter informação local acerca dos nós de monitorização ligados à *gateway* e os dados relativos a estes dispositivos. Tanto a API como a base de dados foram instanciadas em *docker*. Quando novos dados são recebidos na *gateway*, esta fica encarregue de sincronizar estes dados com a *Cloud*. Esta sincronização é feita através de mensagens MQTT, em que os novos dados são publicados num *broker* instanciado na *Cloud*. De seguida o serviço da *Cloud* consome as mensagens do *broker* e insere-as na base de dados.

No *code snippet* abaixo podemos indentificar duas partes importantes do código do microcontrolador ESP32 que se encontra na *gateway*. Na primeira parte do código tem-se o formato da mensagem que é recebida pela *gateway*. Esta mensagem é constituída por quatro strings de dados, cada uma com um tamanho pré-configurado. Na segunda parte do código temos um exemplo do *setup* do ESP32 para escutar novas mensagens dos nós através do ESP-NOW.

```
typedef struct struct_message {
    char thing_id_data[5];
    char photoeletric_data[20];
    char ultrasound_data[50];
    char battery_data[20];
} struct_message;
struct_message myData;
```

```
void setup() {
  Serial.begin(115200);
  rasp.begin(115200, SERIAL_8N1, SERIAL_RX, SERIAL_TX);
  delay(100);

  WiFi.mode(WIFI_STA);
  delay(100);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    ESP.restart();
  } else {
    Serial.println("ESP-NOW initialized!");
  }
  // callback for ESP-NOW
  esp_now_register_recv_cb(OnDataRecv);
  Serial.println("Waiting for messages...");
}
```

A WiPy que se encontra conectada via porta série com a Raspberry Pi, tem a função de receber os dados por ESP-NOW dos nós de monitorização, e entregá-los ao serviço que está instanciado na Raspberry Pi. Foi necessário desenvolver código para a WiPy que se encontra na gateway, para receber estes dados, tratá-los e escrevê-los na porta série para a Raspberry Pi.

Foi criado um script em Python para escutar a porta série em tempo real, receber os dados e enviá-los para um serviço alojado localmente. Este script foi instanciado como serviço no sistema operativo da Raspberry Pi, para ser executado em background.

No *code snippet* abaixo podemos ver o código do *script* que está a ser executado através de um serviço na Raspberry Pi 4. Inicialmente é feita uma conexão à internet, utilizando a rede celular, através da *board* de NB-IoT acoplada à Raspberry. Este serviço está à escuta da porta série por novos dados do ESP32. Quando recebe novos dados, trata-os e envia diretamente para a API que está a ser executada em *container Docker* na Raspberry.

```
wipy = serial.Serial()
wipy.port = "/dev/ttyAMA0"
wipy.baudrate = 115200
wipy.setDTR(False)
wipy.setRTS(False)
wipy.open()
```

```
# connect to gprs
os.system("sudo pon rnet")
os.system("sudo route add default ppp0")

with wipy:
    time.sleep(0.1) # wait for serial to open
    if wipy.isOpen():
        try:
            while True:
                # receive
                while wipy.inWaiting() == 0:
                    pass
                if wipy.inWaiting() > 0:
                    data = str(wipy.readline().decode("utf-8"))
                    wipy.flushInput() # remove data after reading
                    data = data.replace("\n", "")
                    # make local request
                    requests.post("https://0.0.0.0:5000/thingData",
                                  json=data, verify=False)
        except KeyboardInterrupt:
            print("KeyboardInterrupt has been caught.")
```

4.4.2 Serviços de Cloud

Os serviços de *backend* da *Cloud* foram alojados num servidor da AWS. Desenvolveu-se um servidor em *Python* usando a *framework Flask*, uma base de dados *PostgresSQL*, um *website* em *React* e um *broker* MQTT (*Mosquitto*). A escolha da infraestrutura AWS deveu-se sobretudo à facilidade de criação de máquinas virtuais (EC2), ao seu desempenho e por ser gratuita durante um ano. Todos os componentes de software alojados na *Cloud* foram instanciados em *docker* para uma maior facilidade de gestão e dinamismo de recursos do sistema.

O broker MQTT era uma instância do *Mosquitto*, funcionando como intermediário de comunicação entre as *gateways* e o servidor. Este *broker* permite distribuir a informação entre os componentes de uma maneira eficiente, escalável e extremamente leve.

O serviço de *backend* foi desenvolvido em *Python* utilizando a *framework Flask*. Trata-se de uma *framework* com muitas vantagens como, por exemplo, flexibilidade, compatibilidade com tecnologias recentes, escalável, desempenho e rápida implementação. O servidor tem conexão direta com a base de dados, que foi desenvolvida em *PostgresSQL*. O *PostgresSQL* tem a vantagem de permitir uma rápida implementação, ser *open-source* e

com bastante suporte. Foi também usado o *SQLAlchemy*, que é uma biblioteca de mapeamento objeto-relacional (ORM), facilitando a interação entre o código em *Python (Flask)* e a base de dados *PostgresSQL*. Esta biblioteca traduz as classes *Python* em tabelas na base de dados relacional e converte automaticamente as funções em declarações SQL. O servidor também oferece *callbacks* que, na receção de novos dados originários do *broker*, recebem estes dados e os guarda na base de dados.

Por fim, foi desenvolvido um *website* em *React*. Trata-se de uma *framework* em *Javascript*, que tem vindo cada vez mais a ser utilizada, por ser eficiente, flexível, oferecendo componentes reutilizáveis e com uma curva de aprendizagem rápida. Na Figura 4.4 podemos ver como todas estas tecnologias foram integradas na *Cloud*.

Figura 4.4: Tecnologias utilizadas nos serviços Cloud

Fonte: Própria

4.4.3 Aplicação Web

O desenvolvimento da aplicação *Web* foi necessário para a visualização e gestão das *gateways* e nós de monitorização, por parte da autarquia. Esta aplicação possui quatro páginas principais: Mapa, Dispositivos, Operações e Alertas.

Na página Mapa (ver Figura 4.5), temos acesso a todas as *gateways* e nós de monitorização, através da sua localização (latitude e longitude). É possível interagir com o mapa, em que, ao pressionar num ícone dos nós de monitorização é possível consultar as últimas dez comunicações desse dispositivo. Essas informações incluem dados acerca dos sensores de ultrassons e fotoelétrico e também dados relativos ao nível de bateria. Todas estas informações são demonstradas através de gráficos.

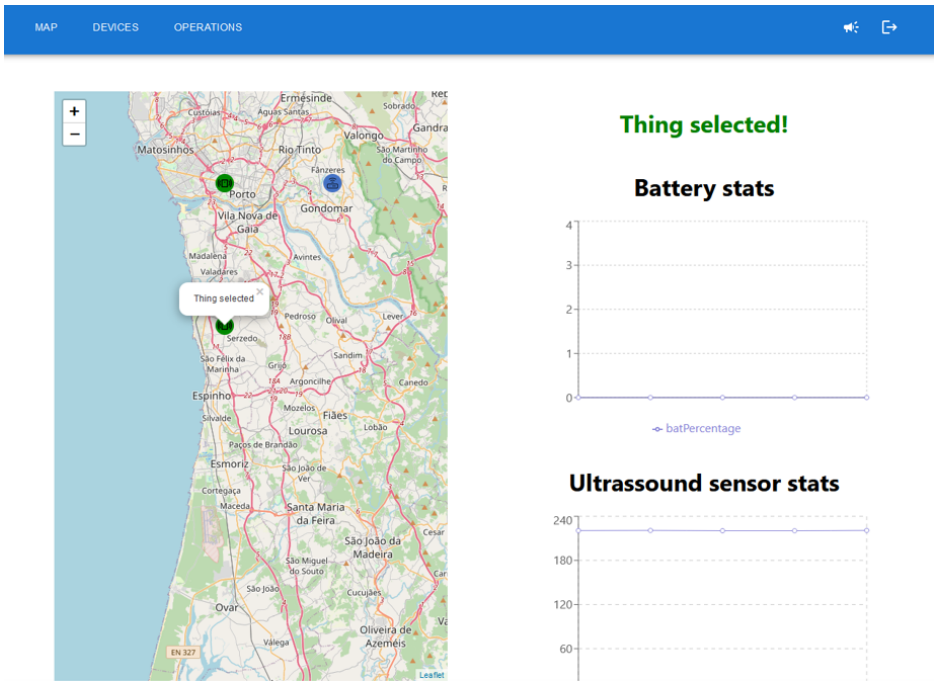

Figura 4.5: Mapa com Gateways e nós de Monitorização

Fonte: Própria

Na página Dispositivos (ver Figura 4.6), podem ver-se todas as *gateways* e os nós de monitorização associados a estas, através de duas tabelas informativas. Na primeira tabela temos acesso às *gateways* e podemos efetuar operações de editar, que permite editar a posição da *gateway* e eliminar, que permite eliminar a *gateway* e todos os nós associados a esta. Possui também uma funcionalidade que mostra, numa segunda tabela, todos os nós de monitorização que estão associados a uma *gateway* específica. A tabela com os nós de monitorização também permite efetuar as mesmas operações de editar e eliminar que temos na tabela de *gateways*. Esta tabela tem uma função extra que permite consultar os dados relacionados ao nó de monitorização, através de outra tabela. Esta tabela possui suporte para paginação, para uma maior comodidade a consultar todos os dados. Também possui a função de exportar todos os dados relacionados com o nós de monitorização para um ficheiro Excel.

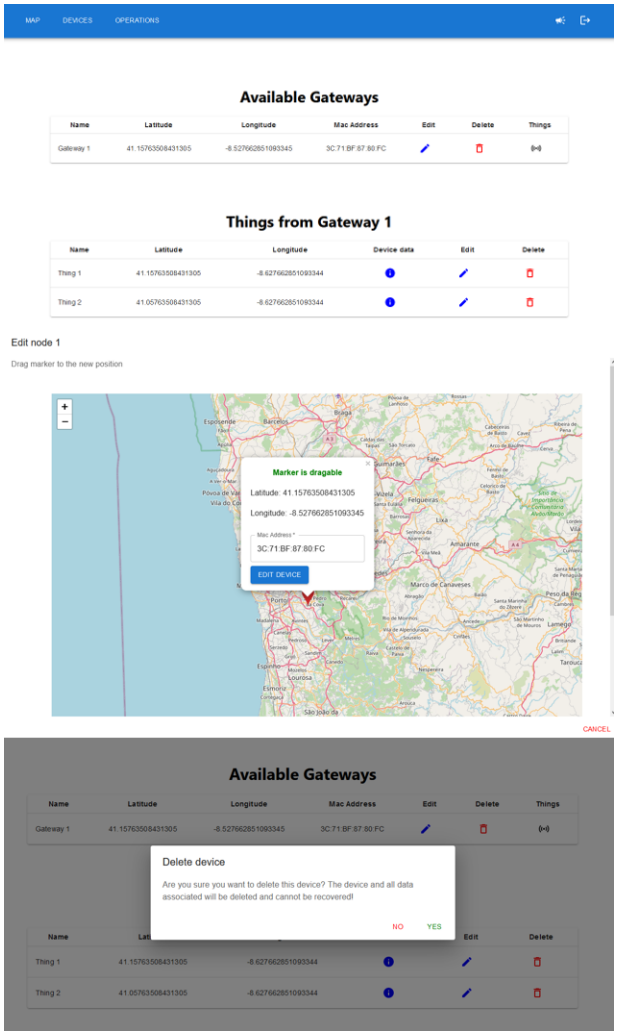

Figura 4.6: Página Dispositivos com operações sobre Gateways e nós de Monitorização

Fonte: Própria

A página Operações (ver Figura 4.7) permite adicionar *gateways* e nós de monitorização através de um mapa. Este mapa possui um marcador, que pode ser deslocado através da sua localização, e que, dependendo do dispositivo a adicionar, muda também o formulário. Caso seja para adicionar uma *gateway*, o formulário requer a introdução do endereço MAC associado a esta. Se for um nó de monitorização então este necessita de ser associado a uma *gateway* existente.

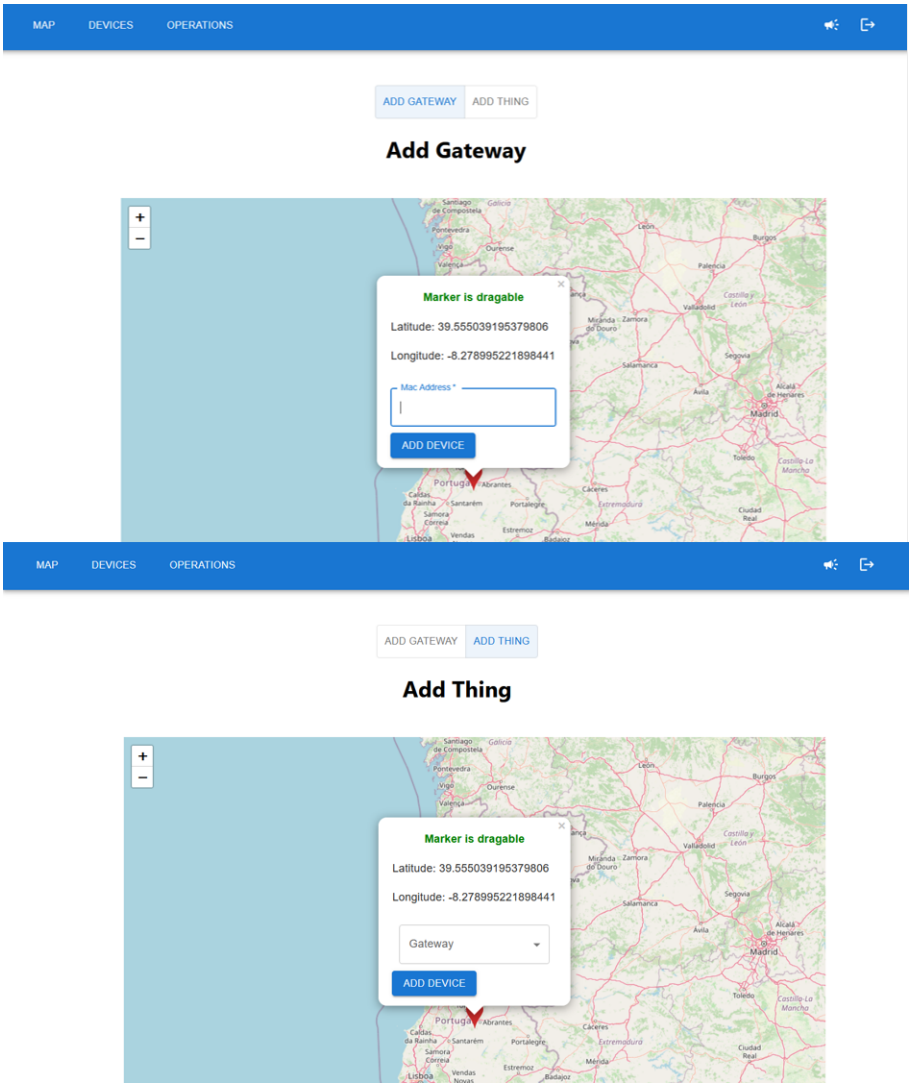

Figura 4.7: Adicionar Gateways e Nós de Monitorização

Fonte: Própria

Por fim, a página de Alertas permite consultar todos os nós de monitorização que estejam a ter algum tipo de problema. Dentro destes problemas podemos, por exemplo, ter alertas sobre o nível de bateria baixo (<20%) e possíveis cheias. As possíveis cheias podem ser detetadas através da informação relativa aos sensores. Por exemplo, caso o sensor fotoelétrico detete água, ou então se a leitura do sensor de ultrassons for de muito próximo ao sensor. Isto permite que a autarquia consulte os locais que possam estar a ter problemas através da sua localização.

4.4.4 Previsão Meteorológica

Para melhorar a eficiência energética dos nós de monitorização, foi pensado utilizar um sistema de previsão meteorológica, para modificar dinamicamente os valores

do número de leituras e o ciclo e período de comunicações dos nós, consoante a previsão metereológica. Desta forma foi possível controlar o número de vezes que os nós ligam e os momentos em que efetuam as comunicações. Sendo estas duas situações que fazem drenar mais a bateria, a previsão meteorológica poderá contribuir para aumentar a autonomia dos nós.

Para adaptar esta funcionalidade ao sistema, foi importante pensar em algumas fases importantes que irão ser descritas de seguida. Na primeira fase tivemos de pensar numa forma de efetuar as leituras das previsões meteorológicas e, em que local isto iria acontecer. Chegamos à conclusão de que estas leituras iriam ser feitas através de um *script python*, consultando um site fidedigno de previsões meteorológicas[1]. Relativamente ao local, selecciona-se a localização das *gateways*. Como estas se encontram na zona dos nós de monitorização, podemos usar a sua latitude e longitude para definir o local da previsão metereológica e assim efetuar a leitura e comunicação aos nós de monitorização.

Na Figura 4.8 podem ver-se as diferentes zonas de Portugal em função dos graus de precipitação. Esta imagem é importante pois permite pensar-nos nos diversos modos de funcionamento que iremos ter. Caso os nós de monitorização estejam situados no norte, iremos ter que efetuar mais leituras e comunicações do que os nós na zona centro e sul. Com isto podemos prever que a drenagem de bateria seja maior nos nós situados mais a norte.

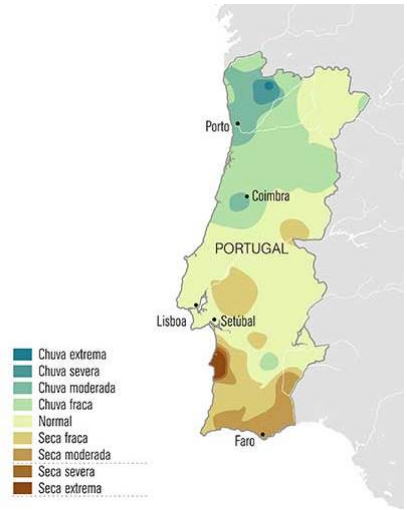

Figura 4.8: Zonas de precipitação em Portugal - Novembro 2019

Fonte: https://nationalgeographic.pt/natureza/actualidade/2450-secas-mais-longas-e-intensas-em-portugal

Numa segunda fase tivemos de decidir como iríamos comunicar esta previsão aos nós de monitorização, para estes atualizarem os seus valores de período de leituras e comunicações. Para isso foi utilizada também a tecnologia de comunicação ESP-NOW.

[1] https://openweathermap.org/

Os nós de monitorização enviam os dados recolhidos para a *gateway*, e esta responde com uma mensagem de sucesso e com um ID do modo de funcioamento que os nós devem adoptar. Visto que a *gateway* já se encontra com as previsões meteorológicas, esta pode decidir qual o modo que os nós de monitorização devem utilizar no seu funcionamento.

Inicialmente foram pensadas duas situações importantes para o sistema. Na Tabela 4.1 podemos ver os diferentes cenários pensados para a alteração dos tempos de comunicações e leituras. No primeiro modo, com uma grande probabilidade de chuva, o sistema irá efetuar leituras de 10 em 10 minutos, e efetuar comunicações de hora em hora. No segundo modo, com um pequena probabilidade de chuva, o sistema irá efetuar leituras de 60 em 60 minutos, e irá fazer comunicações de 12 em 12 horas.

Tabela 4.1: Descrição das diferentes situações de previsão metereológica

Situação	**Descrição**	**Probabilidade de chuva**	**Período de comunicações**	**Período de leituras (min)**
1	Grande probabilidade de chuva	>=30%	6	10
2	Pouca probabilidade de chuva	<30%	6	60

Na Figura 4.9 podemos ver um esquema de como irá ser a arquitetura do sistema com o modo de previsão meteorológica. Os dados irão ser armazenados na *gateway*, que irá correr um script a cada 24 horas para atualizar os seus dados. Quando o nó de monitorização comunica com a *gateway*, esta responde com o modo correspondente, para o nó atualizar os seus valores.

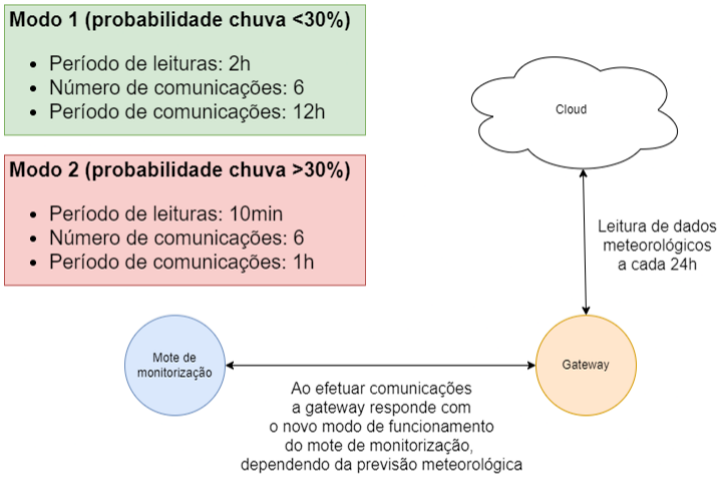

Figura 4.9: Modo de funcionamento dos nós em função da previsão meteorológica

Fonte: Própria

Foi também efetuada uma análise sobre quais os meses em que é mais provável ocorrerem chuvas. Na Figura 4.10 podemos consultar esses dados. Podemos ver que entre os meses de maio e setembro, existe um período de maior seca, o que pode permitir ao sistema estar mais em repouso, pois não necessita de efetuar tantas leituras e comunicações por não existir precipitação. Entre janeiro a abril e outubro a dezembro podemos ver que a probabilidade de precipitação é maior, logo o sistema necessita de estar mais alerta a cheias. Esta imagem possui percentagens de probabilidade de chuva durante todo o ano de 2022. Este ano foi de muita seca, porque não existiu grandes chuvadas, ou seja, iria ser um ano em que o sistema iria estar com leituras e comunicações mais dispersadas.

Figura 4.10: Dados de 2022 para níveis de percipitação em Lisboa

Fonte: `https://ecoativo.pt/2021/12/estruturas-de-retencao-de-agua/`

Com esta implementação vamos permitir ao sistema ter uma maior eficiência energética, e vamos garantir que o sistema gasta os seus recursos de uma maneira mais útil sem existir a necessidade de os desperdiçar em situações desnecessárias. Consequentemente garantimos também com maior segurança a deteção de cheias, pois o sistema irá estar com o maior número de leituras e comunicações em situações de maior probabilidade de chuvas. Existe também uma vantagem nesta implementação que permite personalizar os tempos e modos de operação dos nós de monitorização.

4.5 Conclusão

Neste capítulo foi possível fazer uma descrição geral de todo o processo de implementação do *Smart-city Drainage System*. Começou-se por decidir quais os sensores que iriam ser utilizados no sistema. Tratou-se de uma etapa muito importante por ter várias condicionantes relativas ao uso do sistema. Decidiu-se utilizar o sensor de ultrassons e o sensor fotoelétrico, por serem sensores resistentes, de baixo consumo e que permitiam efectuar leituras do nível de detritos e água nos bueiros.

Na fase de escolha das tecnologias de comunicação, decidiu-se que se usaria o ESP-NOW e o NB-IoT em conjunto com o MQTT, por oferecerem tecnologias e protocolos direcionados sistemas IoT. Os módulos de comunicação foram o ESP32 que suporta ESP-NOW incorporado no microcontrolador e não precisa de hardware extra; o módulo Quectel BG95 M3, que suporta NB-IoT.

Depois de decididos os sensores, módulos e protocolos de comunicação, passau-se à montagem dos componentes de hardware. Foi desenvolvida uma placa PCB (*Printed circuit board*) com todos os componentes do nós de monitorização, bem como da *Gateway*. Foi ainda criado um suporte para a colocação das pilhas. Foram montados dois protótipos dos nós e um da *gateway*, para a realização de todos os testes.

Na fase de desenvolvimento de software, foi implementado todo o código ncessário para os componentes da *Cloud*, *Gateway* e nós de monitorização. Começou-se por desenvolver o código para os nós de monitorização e *gateway*. Nos nós, o código permite as leituras dos sensores e respectivo armazenamento e envio para a *Gateway*. Na *Gateway* foi necessário desenvolver o código para a Raspberry Pi e para o microcontrolador ESP32. Na raspberry Pi instanciou-se um servidor local para receber os dados encaminhados da ESP32. Foram ainda desenvolvidos dois módulos, um para estabelecer a conexão NB-IoT e dar Internet à Raspberry PI e outro para encaminhar os dados da porta série da Raspberry PI para o servidor local.

Na *Cloud* estruturaram-se diversos serviços, entre eles, o serviço de *backend* para receber os daddos das *Gateways*, a base de dados, o *broker* e a aplicação *web* de visualização de dados e alertas. Decidiu-se utilizar contentores *docker* para iniciar e gerir estes serviços. Com o *docker* tornaram-se todos os serviços independentes e autónomos, i.e., conseguiu-se gerir os contentores de forma independente e de maneira mais fácil. Para isto, foi necessário a criação de uma máquina *Linux* na plataforma AWS, para alojar todos os serviços.

O desenvolvimento da aplicação *web* tornou-se fundamental para que os possíveis gestores da autarquia pudessem visualizar e analisar informação recolhida pelos nós de monitorização, bem como visualizar, inserir e editar geograficamente novos nós e *Gateways*. A aplicação *Web* foi desenvolvida em *React*.

Foi também implementada e incorporada no sistema uma reconfiguração dinâmica do modo de funcionamento dos nós de monitorização adaptada à previsão meteorológica. Esta funcionalidade é importante para o sistema, uma vez que permite adaptar os ciclos de funcionamento dos nós e assim ganhar aumentar a poupança de bateria e com isso melhorar a autonomia energética dos nós. Esta funcionalidade foi desenvolvida recorrendo a *scripts Python* na *Gateway*, que em resposta à transmissão de dados por porte dos nós, enviam de volta um ID referente ao modo de funcionamento que os nós devem adoptar.

Chapter 5

Avaliação do Smart-city Drainage System

Neste capítulo avalia-se o *Smart-city Drainage System* desenvolvido. Começa-se por descrever o planeamento dos testes que posteriormente foram efetuados no protótipo. Posteriormente analisam-se os resultados obtidos nos testes e apontam-se as conclusões a que foi possível chegar.

5.1 Testes às Comunicações e Consumo do Sistema

Para aferir a eficácia das comunicações e o consumo energético do protótipo foram planeados diversos testes. Os testes às comunicações procuram avaliar eventuais problemas nas ligações entre os nós de monitorização e a *Gateway*, bem como como entre a *Gateway* e a *Cloud*. No primeiro caso usando comunicações ESP-NOW e no segundo caso usando MQTT sobre ligações NB-IoT.

Relativamente ao consumo energético, foram planeados testes para estimar o tempo de vida útil das baterias. Como se trata de um sistema IoT, no qual os nós irão ser instalados em zonas de difícil acesso e sem alimentação local, é necessário que exista uma elevada autonomia, para evitar a sua manutenção recorrente.

Os testes foram planeados com a utilização de uma *Gateway* e dois nós de monitorização. A *Gateway* estaria ligada à energia elétrica (simulando a alimentação através de a uma luminária), não precisando por isso de baterias. Os dois nós de monitorização possuíam um pack de duas pilhas cada um, de 3.7v e 2100mah cada uma, ligadas em série. Todas as pilhas seriam carregadas a 100% antes da realização dos testes.

Foram considerados dois cenários de comunicação para cada um dos nós de monitorização (ver Tabela 5.1). No primeiro cenário, as leituras e comunicações seriam feitas num menor espaço de tempo, por exemplo, o período das leituras seria de dez minutos e

a comunicação para a *Gateway* seria feita passadas seis comunicações, ou seja, de hora em hora. No segundo cenário as comunicações e leituras seriam feitas num espaço de tempo mais alargado, por exemplo, o período de leituras seria de duas horas e as comunicações para a *Gateway* seriam feitas passado seis leituras, ou seja, de doze em doze horas. Em ambos os cenários, os nós de monitorização estavam a uma distância aproximada de 200m da Gateway. Em ambos os casos haviam obstáculos entre os nós e a Gateway, i.e., várias paredes para simular dificuldades de cobertura similares às encontradas quando os nós forem instalados nos bueiros. O ambiente de testes foi um apartamento, em que um dos nós iria estar próximo à *gateway*, e o outro numa garagem no subsolo. Este nó no subsolo iria estar distanciado três pisos da *gateway*. Estes testes também iriam contemplar os testes às comunicações entre os nós-*gateway* e *gateway-cloud.* Como sabíamos o intervalo de cada comunicação em cada nó, foi-nos possível saber o número de comunicações que o nó deveria efetuar num certo espaço de tempo. No final do teste só tínhamos de comparar o número de comunicações previstas com o de efetuadas.

Tabela 5.1: Descrição dos testes de consumo energético

	Nó 1	**Nó 2**
Bateria inicial (v/%)	4.2v / 100%	4.2v / 100%
Período de leituras (minutos)	10	120
Período de comunicações (horas)	1	12

5.2 Realização dos Testes

Nesta secção descrevem-se os resultados de todos os testes efetuados ao sistema. Como veremos, os resultados dos testes, desenvolvidos em duas fases, permitiram identificar e corrigir alguns problemas, por forma a aprimorar o sistema.

Todos os dados dos testes foram enviados para um servidor alojado na AWS. Esses dados foram guardados numa base de dados postgresSQL, e foram acedidos através uma aplicação web que também se encontra alojado no servidor. O website permite consultar os dados e exportá-los para um ficheiro CSV. Para este estudo, exportaram-se os dados e foi utilizada a ferramenta Excel para a criação dos gráficos correspondentes.

5.2.1 Testes da Fase 1

Numa primeira fase, foram realizados testes com o objetivo de registar o funcionamento dos nós durante um mês completo. Pretendia-se conhecer os gastos médios de bateria em dois cenários distintos como também a taxa de sucesso das comunicações. Um nó iria ter uma grande quantidade de comunicações e outro um número inferior de

comunicações (cf. seccção 5.1). Estes testes permitiram recolher informação sobre o funcionamento do sistema em dois casos extremos, por exemplo, em cenários de seca (em que o sistema não precisa de estar sempre a comunicar) ou em cenários de muita chuva (em que é importante saber, com curtos espaços de tempo, os níveis de enchimento dos bueiros e caixas de drenagem).

Na Tabela 5.2 temos uma descrição das condições em que os testes foram efetuados, como também os resultados de cada um dos nós de monitorização. Os nós são identificado como "Thing 1" e "Thing 2".

Tabela 5.2: Condições e resultados da fase 1 dos testes

	Thing 1	**Thing 2**
Inicio	23/05/2022	23/05/2022
Fim	16/06/2022	16/06/2022
Dias de comunicações	24	24
Total de comunicações previstas	3420	288
Total de comunicações efetuadas	3420	288
Taxa de sucesso das comunicações	100%	100%
Bateria Final	0%	0%

Como podemos verificar na Tabela 5.2, os resultados não foram propriamente os esperados. Ambos os nós de monitorização drenaram as baterias em 24 dias. O nó de monitorização 1 efectuou um maior número de comunicações em relação ao nó 2. Seria portanto espectável que o mote 2 tivesse um período de autonomia maior do que o mote 1.

Relativamente à taxa de sucesso de comunicações, os resultados foram muito animadores. Todas as comunicações entre mote-*gateway* e *gateway-cloud* foram efetuadas com 100% de taxa de acerto. Isto comprova que as tecnologias e protocolos de comunicação utilizados têm excelentes resultados no sistema.

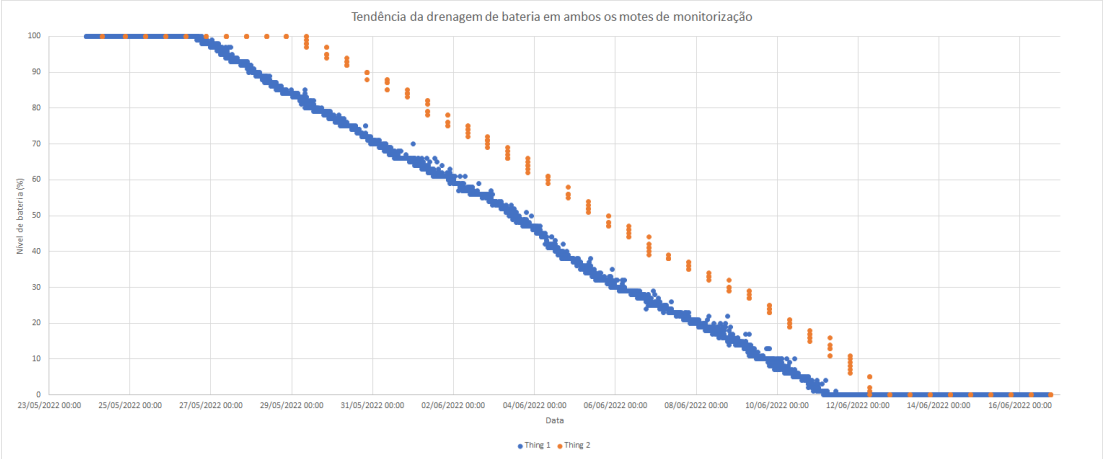

Figura 5.1: Fase 1 - Tendência da drenagem de bateria dos nós de monitorização

Fonte: Própria

Na Figura 5.1 podemos ver a tendência de drenagem de bateria em ambos os nós de monitorização. No eixo das ordenadas temos a percentagem de bateria dos nós de monitorização. No eixo das abcissas, temos a data em que a comunicação da carga da bateria foi feita. Cada um dos pontos apresentados no gráfico representa a carga da bateria medida em cada um dos períodos de comunicações efectuados pelos nós de monitorização. Como se pode verificar o mote 1 apresenta uma maior população de pontos do que o mote 2, ou seja, o mote 1 efetuou um maior número de leituras e comunicações em relação ao mote 2.

Como se pode verificar na Figura 5.1, ambos os nós acabaram por apresentar um declive muito idêntico do nível de drenagem de bateria, mesmo estando sujeitos a um número de comunicações diferente. Procurou-se averiguar a razão para este facto e chegou-se à conclusão que a causa da drenagem das baterias dos nós estava nos sensores, porque estavam directamente ligados às baterias e, portanto, sempre a gastar energia. Este teste foi efetuado com um multímetro, medindo a corrente que alimentava os sensores, mesmo quando o microcontrolador se encontrava em *deepsleep*. Este problema foi identificado e corrigido, passando a alimentação dos sensores a ser efectuada via pins do microcontrolador ESP32. Assim, quando o sistema entrava em *deepsleep*, os pins eram desligados, abrindo o circuito de energia dos sensores. Desta forma, os sensores paravam de drenar bateria. Esta melhoria foi não só feita na parte de hardware, mas também no código do mote. Mesmo em *deepsleep* os pins do ESP32 podem continuar a drenar energia, portanto foi necessário tornar implícito em código essa alteração.

5.2.2 Testes da Fase 2

Estes testes foram efetuados em condições identicas à fase anterior, com os dois nós

de monitorização ligados à mesma *Gateway*. A Tabela 5.3 resume as condições e os resultados obtidos.

Tabela 5.3: Condições e resultados da fase 2 dos testes

	Thing 1	**Thing 2**
Inicio	24/06/2022	24/06/2022
Fim	24/07/2022	24/07/2022
Dias de comunicações	30	30
Total de comunicações previstas	4422	372
Total de comunicações efetuadas	4422	372
Taxa de sucesso das comunicações	100%	100%
Bateria Final	71%	88%

Como podemos verificar na Tabela 5.3, os resultados são bem melhores que no teste anterior. O mote 1, que efetuou um maior número de leituras e comunicações, acabou o período e testes com uma menor percentagem de bateria final em relação ao mote 2. Isto deveu-se ao facto de o mote 1 ter acordado mais vezes e, consequentemente ter ligado mais vezes os sensores e ter efectuado mais comunicações, em relação ao mote 2.

O mote 1 terminou o período de testes com um valor final de bateria de 71%, depois de ter feito um total de 4422 comunicações. Este valor parece-nos aceitável uma vez que o mote foi sujeito a um grande número de leituras e comunicações.

O mote 2 finalizou o período de testes com uma percentagem de bateria de 88%, depois de ter feito 372 comunicações. Este mote efectuou um menor número de leituras e comunicações, o que lhe permitiu gastar menos bateria e, consequentemente, perspectivar uma maior autonomia em relação ao mote 1.

Em ambos os nós a taxa de sucesso de comunicações manteve os 100%, o que representa um ótimo resultado.

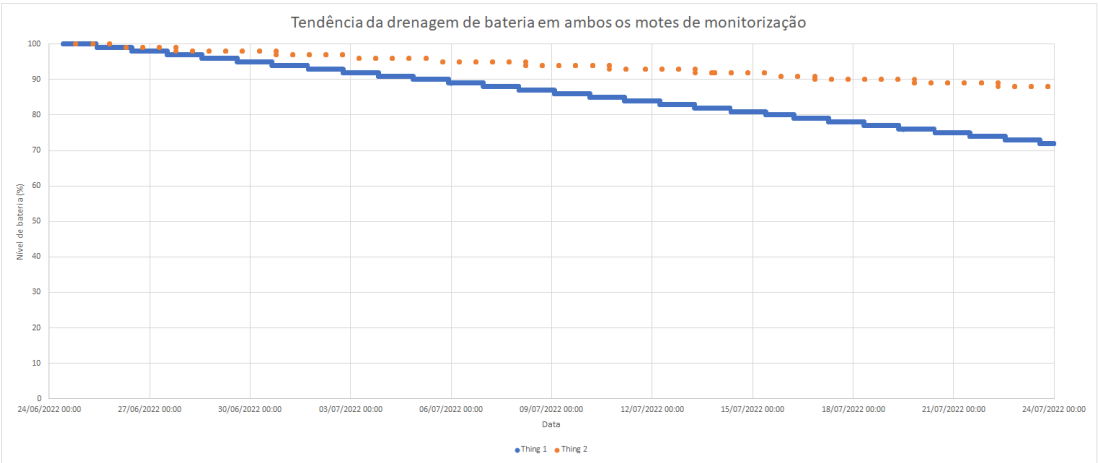

Figura 5.2: Fase 2 - Tendência da drenagem de bateria dos nós de monitorização

Fonte: Própria

Analisando a Figura 5.2, podemos ver que o declive na drenagem da bateria do mote 1 é mais acentuado que o do mote 2. Pode-se concluir assim que a autonomia do mote 2 será superior ao do mote 1.

5.2.3 Previsão da Autonomia dos nós

Procurou-se efectuar uma previsão da duração da bateria para ambos os nós de monitorização, com base nos dados recolhidos nos testes anteriores. Para uma maior facilidade no cálculo desta previsão, foi utilizado a ferramenta Previsão disponível no *Excel*. Recorrendo a esta ferramenta foi-nos possível desenhar os gráficos de previsão.

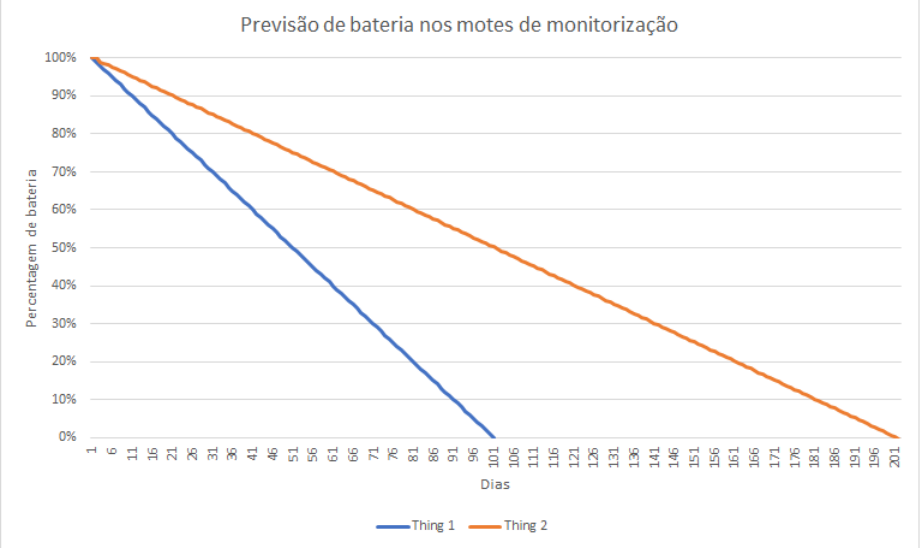

Figura 5.3: Previsão da carga da bateria em ambos os nós de monitorização

Fonte: Própria

Analisando a Figura 5.3 podemos verificar que a previsão de descarga da bateria do mote 1 é bastante superior à do mote 2. Desta forma a autonomia do mote 1 (aproximadamente 100 dias) será metade da do mote 2 (aproximadamente 200 dias). Isto deve-se ao facto do mote 1 efetuar mais leituras e comunicações e, consequentemente, alimentar os sensores um maior número de vezes.

Nestes testes com dois nós, procurou-se comparar duas situações extremas: i) o mote 1 funcionando em ciclos mais curtos para situações de monitorização com muita chuva, ou seja, quando há necessidade de fazer leituras e comunicações com uma maior periodicidade; ii) o mote 2 funcionando em ciclos mais espaçados para situações de monitorização com pouca chuva ou até mesmo de seca, ou seja, quando o mote não necessita de efetuar tantas leituras e comunicações, podendo encontrar-se no estado de *deepsleep* durante um maior espaço de tempo.

5.3 Conclusão

Neste capítulo foi possível descrever os testes que permitiram avaliar os modos de funcionamento e autonomia dos nós do sistema de drenagem de águas pluviais.

Foram simulados dois cenários onde iriam ser feitos os testes aos nós do sistema. Um cenário para situações de monitorização com uma quantidade baixa de leituras e comunicações e outro cenário para uma quantidade elevada de leituras e comunicações. Estes dois cenários permitiram avaliar e prever a autonomia do sistema em situações de monitorização com pouca e muita chuva, respetivamente. Os testes permitiram analisar não só a autonomia do sistema, como também avaliar as comunicações entre os nós de monitorização, a *Gateway* e *Cloud*. Como sabíamos o intervalo de tempo que cada mote iria comunicar, foi-nos possível saber quantas comunicações iria efetuar.

Foram pleneados e realizados testes para analisar o comportamento do sistema ao nível do consumo energético. Estes testes foram efetuados em duas condições distintas: i) para simular uma monitorização mais intensa, adaptada a um nível elevado de chuvas; ii) para simular uma monitorização mais suave, ajustada a um baixo nível de precipitação. Estes testes não só permitiram testar os nós de monitorização, como também todo o processo de comunicação para a *Gateway*, e o consequente envio para a *Cloud*.

Na primeira fase de testes, os resultados não foram muito animadores, uma vez que as baterias só tiveram autonomia para 24 dias. Estes resultados pareciam também incongruentes uma vez que os nós de monitorização estavam sujeios a um número de leituras e comunicações diferente, contudo ambos esgotaram as suas baterias no mesmo período de tempo. Procurou-se perceber a razão para isto acontecer e chegou-se à conclusão que os sensores estavam ligados directamente às baterias e consumiam constantemente grande parte da energia. A resolução deste problema passou pela ligação dos sensores às portas do ESP32. Também uma melhoria em código foi feita para ligar e desligar os pins. Desta forma poderíamos ligar os sensores apenas quando estes eram necessários, e desligá-los quando o sistema se encontrava em *deepsleep*. Relativamente à taxa de sucesso das comunicações, os resultados foram muito animadores. Esta taxa foi de 100% em ambos os nós, o que provaram ser excelentes resultados. Tanto a tecnologia ESP-NOW, como o MQTT provaram ser excelentes tecnologias a ser utilizadas neste teste.

Na segunda fase de testes, já com a alimentação dos sensores via ESP32, tivemos resultados muito mais animadores. Ambos os nós de monitorização conseguiram efetuar comunicações durante os 30 dias de testes e acabaram com 71% e 88%, respectivamente o mote 1 e mote 2. Estes resultados já faziam mais sentido, porque sabíamos que a gestão da energia dos sensores através do ESP32 estava a funcionar. O mote 1 de monitorização, sujeito a um maior número de leituras e comunicações do que o mote 2, acabou com um menor nível de bateria (cerca de 17% menos) do que o mote 2. Mais uma vez a taxa de sucesso de comunicações não desiludiu e manteve-se nos 100%. Isto permitiu-nos saber

que os resultados das comunicações foram constantes.

Com todos os testes concluídos procurou-se fazer uma previsão da autonomia dos nós de monitorização. Esta previsão baseou-se no consumo dos nós 1 e 2, durante a fase de trinta dias de testes. Utilizando a ferramente de previsão do *Excel* conseguimos calcular a previsão de autonomia de ambos os nós de monitorização. Com base nesta ferramenta, verificamos que a autonomia poderia variar entre os 101 e os 201 dias, dependendo do nível de leituras e comunicações efectuados. Salienta-se no entanto que devido a problemas de stock de pilhas, só nos foi possível efetuar testes com duas pilhas em cada mote de monitorização, existindo espaço para mais pilhas. Portanto, com a adição de mais duas ou até quatro pilhas em cada mote, poderíamos extender a autonomia dos nós para o dobro ou triplo. Desta forma conseguia-se tornar um sistema independentes de manutenções por períodos consideravelmente maiores.

Chapter 6

Conclusão

O trabalho proposto nesta dissertação foi concluído na sua totalidade, cumprindo-se todos os objetivos propostos. Temos consciência que existe, contudo, sempre possibilidade para novos desenvolvimentos e melhorias à solução proposta.

O trabalho iniciou-se com uma pesquisa sobre o problema da drenagem de águas pluviais nas vias públicas. Este estudo incidiu sobretudo em artigos científicos e documentos técnicos sobre este problema. Deste trabalho resultou a motivação para descobrir uma possível solução que permitisse resolver ou mitigar os problemas associados às cheias, que danificam a via pública e acarretam despesas para as autarquias e população em geral.

De seguida procuraram-se diversos artigos não só relacionados diretamente com o tema da drenagem de águas pluviais, mas também com outros projetos que incidissem em áreas mais gerais sobre *smart cities* e IoT. O estudo foi orientado em três subtemas: tecnologias de comunicação, tecnologias de coordenação e tecnologias de sensorização. Estes três pontos foram cruciais para o desenvolvimento desta dissertação. Foi importante tomar conhecimento sobre vários trabalhos desenvolvidos nessas áreas. No que toca aos sensores, foi importante tentar perceber as formas de sensorização existentes e adaptadas concretamente aos cenários de utilização pretendidos. Não encontrámos nenhuma solução para os sistemas de drenagem em Portugal, mas foram identificados diversos projetos internacionais com soluções sensorizadas para o problema proposto. Para cada sistema analisamos as respetivas vantagens e desvantagens das soluções propostas.

Na estruturação e desenvolvimento da solução para o *Smart-city Drainage System* procurou-se aplicar uma arquitetura escalável recorrendo a ferramentas e tecnologias adequadas. Efetuaram-se testes a vários tipos de sensores para decidir quais fariam mais sentido utilizar, e analisaram-se vários componentes de hardware para escolher aqueles que melhor se adequavam ao protótipo pretendido. Esta metodologia foi fundamental para planear e efetuar a montagem e implementação do protótipo.

Para o desenvolvimento do protótipo foi decidido construir dois nós de monitorização e uma *Gateway*, os primeiros baseados em microcontroladores ESP32 e a segunda baseada numa Raspberry Pi 4. Foi ainda desenvolvido todo o software necessário para

estes três componentes. Foi também desenvolvida a infraestrutura de serviços para enviar e receber da *cloud*, uma aplicação *web* para apresentar os dados e um *backend* para interagir com o *website* e a base de dados. Um dos grandes desafios colocados ao desenvolvimento deste sistema estava relacionado com a autonomia dos nós. Utilizou-se um sistema de previsão meteorológica que permitiu gerir de forma mais adequada os tempos de ativação dos nós, permitindo dessa forma aumentar significativamente a sua autonomia sem comprometer as funcionalidades desejadas.

A avaliação do protótipo procurou analisar os vários aspetos funcionais e não funcionais do sistema. Foram criados os requisitos para efetuar os testes no protótipo, ou seja, em que condições iriam ser realizados. Este capítulo foi importante, pois foi nele que o sistema foi todo montado e desenvolvido. Esta montagem permitiu-nos saber se o sistema era viável para ser colocado nas caixas de drenagem e coletoras, com toda a montagem de hardware que foi feita. O desenvolvimento a nível de software também foi importante para todo o fluxo de dados do sistema. Foram realizados vários testes que permitiram comparar resultados e identificar melhorias ao sistema. Em particular os resultados dos testes à autonomia dos nós foram muito encorajadores, uma vez que com apenas duas pilhas se conseguiram períodos de funcionamento autónomo acima dos 100 dias. Só conseguimos efetuar os testes com duas pilhas devido a uma crise na encomenda de baterias que estamos a passar neste momento, mas mesmo assim foi-nos possível efetuar os testes e comprovar o conceito, onde demonstraram serem bastante animadores. A partir destes testes também nos foi possível estudar a taxa de sucesso das comunicações. Os resultados foram ótimos, uma vez que em ambos os nós a taxa de sucesso foi de 100%. Com estes resultados comprovou-se que as tecnologias de comunicação ESP-NOW e MQTT foram boas escolhas para o sistema.

Adicionalmente, os testes com os sensores selecionados deram boas indicações sobre a possibilidade de deteção de entupimentos nas caixas de recolha. Relativamente às comunicações, testaram-se as ligações dos nós à *gateway* em situações de alcance semelhantes às reais, também com resultados que garantem cobertura nas diversas situações em que os nós possam ser instalados dentro das caixas coletoras (abaixo do solo) na via pública.

Em resumo, o sistema proposto oferece uma solução viável às autarquias para recolherem informação sobre o estado do sistema de drenagem de águas pluviais. O sistema permite, por exemplo, consultar o nível de bateria dos nós de monitorização bem como os alertas gerados para cada caixa de drenagem, de modo a fazer um planeamento das ações de manutenção destas caixas. Ou seja, com a informação recolhida pela autarquia será possível desencadear a limpeza das caixas e assim prevenir que ocorram cheias. Será por isso uma mais valia pois, como se tratam de sistemas de baixo custo e facilmente escaláveis, irá permitir à autarquia poupar nos custos de manutenção e em potenciais prejuízos resultantes de cheias na via pública.

Estamos conscientes que o sistema tem margem para novas funcionalidades e melhoramentos. Por isso, o trabalho futuro poderá contemplar a utilização de algoritmos de inteligência artificial que melhorem a previsão da ocorrência de cheias, recorrendo não só à informação recolhida, mas também à integração de outros serviços de *cloud* (e.g. previsão atmosférica, gestão de alertas, interação e *feedback* dos transeuntes, etc.). Outro aspeto fundamental está relacionado com a utilização de energia produzida localmente para melhorar a autonomia dos nós. Por exemplo, os nós poderiam ser alimentados por fontes de energia acionados pela passagem de carros nas vias públicas. Também poderia ser explorada a utilização de energia solar ou do vento, por exemplo. Por fim, poderíamos considerar também a alimentação das *gateways* com baterias, para situações em que não pudessem ser instaladas nas luminárias. Poderíamos assim alargar a sua instalação a locais onde o sistema de iluminação pública não estivesse disponível, explorando também formas de produção energética alternativas.

Referências

Aazam, M., Khan, I., Alsaffar, A. A. and Huh, E.-N. (2014). Cloud of things: Integrating internet of things and cloud computing and the issues involved, *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th - 18th January, 2014*, pp. 414–419. 8, 9

Al-Haidari, F. and Alqahtani, E. (2020). Securing communication between fog computing and iot using constrained application protocol (coap): A survey, *Journal of Communications* **15**: 14–30. 18

Ananth, S., Sathya, P. and Madhan Mohan, P. (2019). Smart health monitoring system through iot, *2019 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0968–0970. 8

Atlam, H., Alassafi, M., Alenezi, A., Walters, R. and Wills, G. (2018). Xacml for building access control policies in internet of things, *XACML for Building Access Control Policies in Internet of Things*. 13

Atlam, H., Walters, R. and Wills, G. (2018). Fog computing and the internet of things: A review, *Big Data and Cognitive Computing* **2**. 13

Bagherzadeh, L., Shahinzadeh, H., Shayeghi, H., Dejamkhooy, A., Bayindir, R. and Iranpour, M. (2020). Integration of cloud computing and iot (cloudiot) in smart grids: Benefits, challenges, and solutions, *2020 International Conference on Computational Intelligence for Smart Power System and Sustainable Energy (CISPSSE)*, pp. 1–8. 9

Bandyopadhyay, S. and Bhattacharyya, A. (2013). Lightweight internet protocols for web enablement of sensors using constrained gateway devices, *2013 International Conference on Computing, Networking and Communications (ICNC)*, pp. 334–340. 18, 19

Bauer, J. and Aschenbruck, N. (2017). Measuring and adapting mqtt in cellular networks for collaborative smart farming, *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pp. 294–302. 18

Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F. and Parashar, M. (2017). Mobility-aware application scheduling in fog computing, *IEEE Cloud Computing* **4**(2): 26–35. 13

Bonomi, F. and Milito, R. (2012). Fog computing and its role in the internet of things, *Proceedings of the MCC workshop on Mobile Cloud Computing* . 11

Bonomi, F., Milito, R., Natarajan, P. and Zhu, J. (2014). *"Fog Computing: A Platform for Internet of Things and Analytics" Flavio Bonomi, Rodolfo Milito, Preethi Natarajan and Jiang Zhu, N. Bessis and C. Dobre (eds.), Big Data and Internet of Things: 169 A Roadmap for Smart Environments, Studies in Computational Intelligence 546, DOI: 10.1007/978-3-319-05029-4_7, © Springer International Publishing Switzerland 2014*, Springer International Publishing Switzerland, chapter Fog Computing: A Platform for Internet of Things and Analytics, pp. 169–186. 12

Cama-Pinto, A., Piñeres-Espitia, G., Zamora-Musa, R., Acosta-Coll, M., Caicedo-Ortiz, J. and Sepulveda Ojeda, J. (2016). Design of a wireless sensor network for monitoring of flash floods in the city of barranquilla, colombia, **24**: 581–599. 29

Cao, K., Liu, Y., Meng, G. and Sun, Q. (2020). An overview on edge computing research, *IEEE Access* **8**: 85714–85728. 10

Carvalho, G., Cabral, B., Pereira, V. and Bernardino, J. (2021). Edge computing: current trends, research challenges and future directions, *Computing* **103**. 11

Chang, N.-B. and Guo, D.-H. (2006). Urban flash flood monitoring, mapping and forecasting via a tailored sensor network system, *Urban Flash Flood Monitoring, Mapping and Forecasting via a Tailored Sensor Network System*, pp. 757 – 761. 29

Chiang, M. and Zhang, T. (2016). Fog and iot: An overview of research opportunities, *IEEE Internet of Things Journal* **3**(6): 854–864. 13

Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J. R., Mellouli, S., Nahon, K., Pardo, T. A. and Scholl, H. J. (2012). Understanding smart cities: An integrative framework, *2012 45th Hawaii International Conference on System Sciences*, pp. 2289–2297. 6

Cisco et al. (2015). Fog computing and the internet of things: Extend the cloud to where the things are. 12

Dinculeană, D. and Cheng, X. (2019). Vulnerabilities and limitations of mqtt protocol used between iot devices, *Applied Sciences* **9**: 848. 17

Garcia, F., Retamar, A. and Javier, J. (2015). A real time urban flood monitoring system for metro manila, *A real time urban flood monitoring system for metro Manila*, pp. 1–5. 29

Hassan, N., Gillani, S., Ahmed, E., Yaqoob, I. and Imran, M. (2018). The role of edge computing in internet of things, *IEEE Communications Magazine* **56**(11): 110–115. 11

Hassan, Z., Ali, H. and Badawy, M. (2015). Internet of things (iot): Definitions, challenges, and recent research directions, *International Journal of Computer Applications* **128**: 975–8887. 7

Hoang, T., Van, S.-T. and Nguyen, B. (2019). Esp-now based decentralized low cost voice communication systems for buildings, *ESP-NOW Based Decentralized Low Cost Voice Communication Systems For Buildings*, pp. 108–112. 15, 16

Hori, M., Kawashima, E. and Yamazaki, T. (2010). Application of cloud computing to agriculture and prospects in other fields, *Fujitsu scientific & technical journal* **46**: 446–454. 7

Kanade, P., Alva, P., Prasad, J. P. and Kanade, S. (2021). Smart garbage monitoring system using internet of things(iot), *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 330–335. 7, 24, 51

Ketel, M. (2017). Fog-cloud services for iot, *Fog-Cloud Services for IoT*, pp. 262–264. 13

Lampkin, V. et al. (2012). *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*, RedBooks, United States. 17

Li, Y., Cheng, X., Cao, Y., Wang, D. and Yang, L. (2018). Smart choice for the smart grid: Narrowband internet of things (nb-iot), *IEEE Internet of Things Journal* **5**(3): 1505–1515. 14, 15

Liu, Y., Fieldsend, J. and Min, G. (2017). A framework of fog computing: Architecture, challenges and optimization, *IEEE Access* **PP**: 1–1. 12

Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R. H., Morrow, M. J. and Polakos, P. A. (2018). A comprehensive survey on fog computing: State-of-the-art and research challenges, *IEEE Communications Surveys & Tutorials* **20**(1): 416–464. 13

Muteba, F., Djouani, K. and Olwal, T. (2019). A comparative survey study on lpwa iot technologies: Design, considerations, challenges and solutions, *Procedia Computer Science* **155**: 636–641. The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2019),The 14th International Conference on Future Networks and Communications (FNC-2019),The 9th International Conference on Sustainable Energy Information Technology.
URL: *https://www.sciencedirect.com/science/article/pii/S1877050919310063* 16

Nagaraja, G. S., Soppimath, A. B., Soumya, T. and Abhinith, A. (2019). Iot based smart agriculture management system, *2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, pp. 1–5. 7, 27

Naik, N. (2017). Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http, *2017 IEEE International Systems Engineering Symposium (ISSE)*, pp. 1–7. 18, 20

Nastase, L. (2017). Security in the internet of things: A survey on application layer protocols, *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, pp. 659–666. 16

Patel, K., Patel, S., Scholar, P. and Salazar, C. (2016). Internet of things-iot: Definition, characteristics, architecture, enabling technologies, application & future challenges. 7

Pedroso, J. et al. (2013). *Hidrologia urbana – Sistemas de drenagem de águas pluviais urbanas*, entidade reguladora dos serviços de Águas e resíduos - universidade de coimbra edn, Universidade de Coimbra, Lisboa. 28

Peralta, G., Iglesias Urkia, M., Barceló, M., Cid Fuentes, R., Moran, A. and Bilbao, J. (2017). Fog computing based efficient iot scheme for the industry 4.0, *Fog computing based efficient IoT scheme for the Industry 4.0*. 12, 13

Peter, N. (2015). Fog computing and its real time applications, *International Journal of Emerging Technology and Advanced Engineering* . 11, 12

Ratasuk, R., Vejlgaard, B., Mangalvedhe, N. and Ghosh, A. (2016). Nb-iot system for m2m communication, *2016 IEEE Wireless Communications and Networking Conference*, pp. 1–5. 14, 15

Satyanarayanan, M. (2017). The emergence of edge computing, *Computer* **50**(1): 30–39. 10

Shi, W. and Zhang, X. (2019). Edge computing: State-of-the-art and future directions, *Journal of Computer Research and Development* **56**: 69–89. 10

Singh, S. and Singh, N. (2015). Internet of things (iot): Security challenges, business opportunities & reference architecture for e-commerce, *Internet of Things (IoT): Security challenges, business opportunities & reference architecture for E-commerce*. 7

Sinha, R. S., Wei, Y. and Hwang, S.-H. (2017). A survey on lpwa technology: Lora and nb-iot, *ICT Express* **3**(1): 14–21.
URL: *https://www.sciencedirect.com/science/article/pii/S2405959517300061* 16

Soni, D. and Makwana, A. (2017). A survey on mqtt: A protocol of internet of things(iot), *A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)*. 17

Soomro, T. and Wahba, H. (2010). Perspectives of cloud computing: An overview, *Perspectives of Cloud Computing: An Overview*. 8

Sunkpho, J. and Oottamakorn, C. (2011). Real-time flood monitoring and warning system, *Songklanakarin Journal of Science and Technology* **33**: 227–235. 29

Systems, E. (2021). Esp-now: Espressif's wireless communication protocol, `https://www.espressif.com/en/news/ESP-NOW`. Accessed: 2022-03-26. 15

Systems, E. (2022). Esp-now, `https://www.espressif.com/en/products/software/esp-now/overview`. Accessed: 2022-03-27. 15

Tanim, M. M. Z. (2015). Study on provision of low-cost machine-type communications (mtc) user equipments (ues) based on lte. 14

Vaquero, L. and Rodero-Merino, L. (2014). Finding your way in the fog: Towards a comprehensive definition of fog computing, *HP Laboratories Technical Report* **44**. 12

Verma, M., Bhardwaj, N. and Yadav, A. (2016). Real time efficient scheduling algorithm for load balancing in fog computing environment, *International Journal of Information Technology and Computer Science* **8**: 1–10. 12

Wang, Y.-P. E., Lin, X., Adhikary, A., Grovlen, A., Sui, Y., Blankenship, Y., Bergman, J. and Razaghi, H. S. (2017). A primer on 3gpp narrowband internet of things, *IEEE Communications Magazine* **55**(3): 117–123. 14

Wukkadada, B., Wankhede, K., Nambiar, R. and Nair, A. (2018). Comparison with http and mqtt in internet of things (iot), *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 249–253. 18

Yarlagadda, R. T. (2018). Internet of things & artificial intelligence in modern society, *SSRN Electronic Journal* **6**: 374. 6, 7