Universidade de Aveiro
2023

João Miguel
Silva Lopes

Registo inteligente e remoto de dados, com processamento para equipamentos de termotecnologia da Bosch

Smart remote data logging and data processing for Bosch thermotechnology equipment, in Bosch

Universidade de Aveiro
2023

João Miguel
Silva Lopes

Registo inteligente e remoto de dados, com processamento para equipamentos de termotecnologia da Bosch

Smart remote data logging and data processing for Bosch thermotechnology equipment, in Bosch

Relatório de Estágio apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação científica de José Paulo Oliveira Santos, Professor Auxiliar, do Departamento de Engenharia Mecânica da Universidade de Aveiro.

O presente trabalho foi realizado ao abrigo do Projeto "Agenda ILLIANCE" [C644919832-00000035 | Projeto n.º46], financiado pelo PRR - Programa de Recuperação e Resiliência, no âmbito do Next Generation EU da União Europeia, e contou com apoio laboratorial do Centro de Tecnologia Mecânica e Automação (TEMA), projetos UIDB/00481/2020 e UIDP/00481/2020.

O júri / The jury

Presidente / President

Prof. Doutor Vítor Manuel Ferreira dos Santos
Professor Associado C/ Agregação da Universidade de Aveiro

Vogais / Committee

Prof. Doutor Pedro Alexandre de Sousa Gonçalves
Professor Adjunto da Universidade de Aveiro

Prof. Doutor José Paulo Oliveira Santos
Professor Auxiliar da Universidade de Aveiro (orientador)

Agradecimentos / Acknowledgements

Agradeço ao meu orientador, Professor Doutor José Santos, por me ter motivado a começar este projeto e por todo o apoio prestado durante a sua execução.
Pretendo também agradeçer a todos os profissionais da Bosch Home Comfort de Aveiro e em especial ao Engenheiro Daniel Mota por todo o apoio e pela confiança que depositaram em mim.
Quero ainda agradecer à minha família e à minha namorada pelo apoio incondicional ao longo deste trabalho e do meu percurso académico.
Agradeço também ao Departamento de Engenharia Mecânica e à Universidade de Aveiro pelo percurso que me proporcionaram.
Por último, quero agradecer a todas as pessoas que não foram mencionadas, mas que contribuíram para a realização deste trabalho.

Keywords Remote data logging, internet of things, machine-to-machine communications, cloud computing, data aggregation, outlier detection

Abstract In order to develop new water heaters, Bosch Thermotechnology S.A. does lifetime tests to the equipments. These tests require data related to the load profiles of these appliances while in the field. Limitations in the current solution used by the company to gather this data have been complicating the process of the development of the required test plans. Some of these limitations are: data is registered only in a SD card and there is no automatic data processing.

In this project, it was developed a data logger that allows to acquire information from water heaters during field tests and sending that data to a cloud server, using wireless communications. The system also includes the data processing, in order to calculate the necessary metrics to define the load/operation profiles of these equipments, including the number of starts, the consumed energy and the temperature, flow rate and power distributions.

It was built an hardware to acquire data from external sensors or directly from the electronic control units of the appliances, using a micro controller (ESP32). This hardware includes a local interface that allows an installer to set up and receive feedback from the device.

A cloud instance was used to integrate all the tools necessary to have a backend where data is ingested into a data base (InfluxDB) and aggregate the information to calculate the relevant metrics, namely power, flow and temperature distributions, number of starts and consumed energy.

It was also studied the use case of pre-processing of time series algorithms, namely Isolation Forest for outlier detection and KNNImputer for missing data imputation.

It was also built a frontend with dynamic dashboards, that allow a user to visualize data, from time series to load profiles of a specified equipment or field test.

The solution had positive results on multiple levels: the time series were coherent between each other and with the reality, the metrics represented the time series with precision and the algorithm detected outliers effectively. The data logger turned out to be quite robust, however there is some future work that can be done, to improve the solution.

Palavras-chave Registo remoto de dados, internet das coisas, comunicações entre máquinas, computação em cloud, agregação de dados, deteção de outliers

Resumo

Para o desenvolvimento de novos esquentadores, a Bosch Termotecnologia S.A. faz testes de durabilidade aos equipamentos. Para a realização destes testes necessita de dados com informação acerca dos perfis de utilização dos aparelhos quando estão no terreno. Limitações nas formas atualmente utilizadas pela empresa para recolher dados durante os ensaios de campo têm dificultado o processo de desenvolvimento dos planos de teste necessários. Algumas dessas principais limitações são: registo de dados apenas em cartão de memória e inexistência de processamento automático de dados.

Neste projeto foi desenvolvido um data logger que permite a aquisição de informação dos esquentadores durante os ensaios de campo e envio desses dados para um servidor alojado na cloud, através de uma ligação à internet. O sistema inclui também o processamento dos dados recolhidos, de forma a calcular as métricas necessárias para traçar os perfis de carga/utilização dos aparelhos, incluindo o número de arranques, a energia consumida e as distribuições de temperatura, caudal e potência.

Foi construído um hardware de aquisição de dados que pode recolher informação tanto de um conjunto de sensores externos como diretamente das unidades de controlo eletrónico dos equipamentos, utilizando um microcontrolador (ESP32). Este hardware inclui uma interface local que permite a um instalador tanto configurar um dispositivo como receber feedback do mesmo.

Foi utilizada uma instância na cloud, onde foi implementado um backend responsável por inserir os dados recolhidos numa base de dados (InfluxDB). Além disso, permite a agregação da informação de forma a traçar algumas métricas relevantes, nomeadamente distribuições de potência, caudal e temperaturas de entrada e saída da água, número de arranques e consumo de energia.

Foram estudados alguns algoritmos de pré-processamento de séries temporais, nomeadamente o algoritmo Isolation Forest para deteção de outliers e o algoritmo KNNImputer para preencher dados em falta.

Foi ainda construído um frontend com um conjunto de dashboards dinâmicas e interativas onde um utilizador poderá visualizar os dados, desde as séries temporais até aos perfis de carga de um determinado equipamento.

A solução implementada teve resultados positivos em vários níveis: as séries temporais foram coerentes entre si e com a realidade, as métricas calculadas traduziram as séries temporais com precisão e o algoritmo implementado detetou outliers com eficácia.

O data logger revelou ser bastante robusto, contudo ainda há pontos que podem ser melhorados em trabalho futuro.

Índice

Lista de Tabelas

Intentionally blank page.

Lista de Figuras

Intentionally blank page.

Capítulo 1

Introdução

Este trabalho foi desenvolvido em estágio na empresa Bosch Termotecnologia S.A. Neste capítulo será apresentado o enquadramento do trabalho, em particular as necessidades e o problema da empresa, o ponto de partida dos trabalhos, a motivação e o objetivo do mesmo.

1.1 Enquadramento

Nas últimas décadas, o avanço tecnológico tem permitido o desenvolvimento de sistemas de monitorização de equipamentos cada vez mais sofisticados. Atualmente, conceitos como Internet das Coisas (IoT) e Computação em Cloud abrem portas para um novo paradigma na aquisição remota de dados. Com estas tecnologias é possível dar vida aos equipamentos, tornando automáticas e remotas tarefas que antes tinham de ser realizadas localmente. Desta forma consegue-se simplificar processos, reduzir despesas com deslocações, aceder a dados em qualquer lugar do planeta e interligar equipamentos distribuídos remotamente. Os dois conceitos anteriores são a base tecnológica que possibilita a realização deste trabalho.

A Bosch Termotecnologia, S.A. é uma empresa da região de Aveiro focada no desenvolvimento e produção de equipamentos de termotecnologia, como esquentadores, termoacumuladores, caldeiras e bombas de calor. No centro de engenharia da empresa há a necessidade de realizar testes de durabilidade aos equipamentos nas mesmas condições a que eles são sujeitos no campo, em casa dos clientes. As condições de teste são definidas em planos que se baseiam nos perfis de carga e têm como objetivo a melhoria contínua dos equipamentos já desenvolvidos pela empresa e suportar o desenvolvimento de novos. Para a realização destes testes, a Bosch precisa de informação relativa aos perfis de carga/operação dos equipamentos quando estão a operar no terreno. Os perfis de carga dos esquentadores traduzem-se por: quantas vezes por dia/ano são ligados, quanto tempo estão em operação, o rácio de tempo com uma determinada gama de temperatura, caudal e potência, o número de arranques, a energia consumida, etc. Para obter estes perfis, a Bosch recorre a ensaios de campo, através dos quais são instalados esquentadores em casas de clientes e dos quais são recolhidos dados. Posteriormente, esses dados são utilizados para calcular métricas que, em conjunto com as métricas dos outros ensaios, permitem traçar os perfis de carga dos equipamentos.

Posto isto, o problema presente é: como recolher, processar e visualizar os dados de

operação dos esquentadores que são instalados em casas de clientes. Algumas dificuldades associadas a este problema são:

- Existência de vários modelos diferentes de esquentadores, o que dificulta a uniformização de uma solução que possa ser utilizada em todos eles.
- Necessidade de utilização de soluções remotas, visto que os esquentadores são distribuídos remotamente por muitos países diferentes.
- Necessidade de tornar a solução fácil de instalar, sem haver necessidade de conhecimentos ou equipamentos específicos ou mesmo reprogramação da solução.
- Necessidade de incluir mecanismos para gerir modos de falha, minimizando os impactos de eventuais problemas.

Os esquentadores são equipamentos de aquecimento de água instantânea. No caso dos esquentadores a gás há uma entrada de gás natural para uma câmara de combustão onde ocorre a combustão desse gás, sendo esse o elemento de aquecimento. No caso dos esquentadores elétricos o elemento de aquecimento é um conjunto de resistências elétricas. Em ambos os casos, o caudal de água entra no esquentador e passa pela câmara/resistência onde aquece. Assim, tem-se um caudal de entrada de água fria e uma saída de água quente. No caso dos esquentadores a gás, os gases de combustão são depois ventilados para o exterior através da parte superior do esquentador. O equipamento tem uma Unidade de Controlo Eletrónico, ECU, que permite, por exemplo, o controlo do arranque da combustão, do ventilador e das válvulas e ainda a monitorização de vários parâmetros de operação do equipamento. Das gamas de esquentadores produzidos pela empresa destacam-se três grandes grupos:

- Equipamentos totalmente elétricos - equipamentos que apenas usam corrente elétrica para toda a operação.
- Equipamentos com aquecimento a gás e alimentação por corrente elétrica - equipamentos que utilizam gás para aquecimento mas têm uma ECU alimentada pela corrente elétrica, através de uma tomada.
- Equipamentos com aquecimento a gás e alimentação a pilhas - equipamentos que utilizam gás para aquecer a água, no entanto a ECU é alimentada através de duas pilhas de 1.5V.

Dentro de cada um destes grupos há vários modelos diferentes. Uma questão relevante para este trabalho diz respeito às características da ECU de cada um dos equipamentos. No caso dos aparelhos desenvolvidos pela Bosch é possível aproveitar as potencialidades das ECUs dos aparelhos para adquirir muitos dos dados pretendidos. Nestes casos, a ECU tem uma interface e um protocolo de comunicação específico que permite que a solução que se pretende desenvolver adquira os dados diretamente da ECU. No entanto, existe um outro grupo de equipamentos que não foram desenvolvidos pela Bosch, mas que a empresa também produz e pretende colocar nos ensaios de campo. Nesses aparelhos terão sempre de ser utilizados sensores externos para adquirir os dados. Na figura 1.1 estão representadas duas ECUs de dois esquentadores da Bosch.

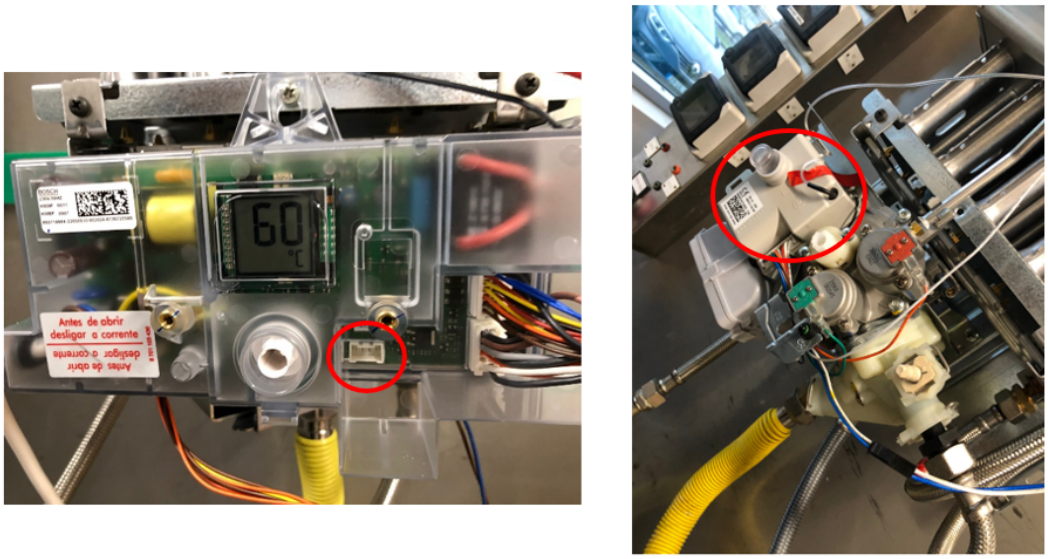

Figura 1.1: Fotografias das ECUs de um esquentador a gás com alimentação elétrica (à esquerda) e de um esquentador a gás, mas com alimentação da ECU a pilhas (à direita)

Conforme apresentado na figura 1.1, uma delas permite a comunicação através de uma porta de comunicação (assinalada na imagem da esquerda) e a outra não tem sequer a porta de comunicação (ver imagem à direita). No caso das ECUs desenvolvidas pela Bosch, o protocolo de comunicação utilizado está implementado na interface *Universal Asynchronous Receiver / Transmitter* (UART). As ECUs dos equipamentos enviam automaticamente mensagens com os dados de operação para quaisquer periféricos que se liguem à interface UART. Desse modo, com a documentação necessária, é possível ter um microcontrolador a adquirir dados das ECUs. Por motivos de confidencialidade, alguns detalhes do protocolo de comunicação utilizado pelas ECUs não serão abordados.

Dada a importância da distinção entre os vários equipamentos apresenta-se na Tabela 1.1 um resumo das características mais relevantes de cada um dos modelos que serão alvo deste trabalho.

Com base na tabela 1.1, é possível verificar a quantidade de modelos diferentes dos quais se pretendem registar dados. Em cerca de metade dos modelos é possível comunicar com a ECU, o que permite reduzir o número de sensores externos necessários. No entanto, em muitos modelos não é possível comunicar com a ECU e, por isso, o sistema de registo de dados tem de estar preparado para utilizar apenas sensores externos.

Atualmente a empresa tem utilizado uma solução que permite cumprir com algumas necessidades e possibilitado o registo de dados em vários ensaios de campo. Esta solução, construída na empresa (ver figura 1.2), é capaz de registar alguns dos dados pretendidos. Ainda assim, apresenta várias limitações e problemas.

Conforme mostra a figura 1.2, a solução atual utiliza uma placa PCB bastante complexa, que corresponde à ECU de um dos esquentadores da Bosch. Esta opção apresenta algumas desvantagens, nomeadamente limitações ao nível dos sensores que podem ser utilizados, dificuldades na configuração, dificuldades na produção e o espaço ocupado. Além disto, o registo de dados, na maior parte dos casos, pode apenas ser feito para um cartão de memória, dificultando o processo de recolha dos dados. Nesses casos, é necessário um

Tabela 1.1: Catálogo dos equipamentos com as características mais relevantes para este trabalho

Modelo	Aquecimento	Alimentação	Comunicação c/ECU
EWI DNA	Resistência elétrica	Trifásica e 32/40 A	Sim
EWI de baixa potência	Resistência elétrica	Monofásica e 16 A	Não
EWS (cilindros)	Resistência elétrica	Monofásica e 16 A	Não
KME	Gás	Corrente	Sim
FP2	Gás	Corrente	Sim
FPLowNox	Gás	Corrente	Sim
Compact 4	Gás	Pilhas	Não
Aparelhos OF a baterias	Gás	Pilhas	Não

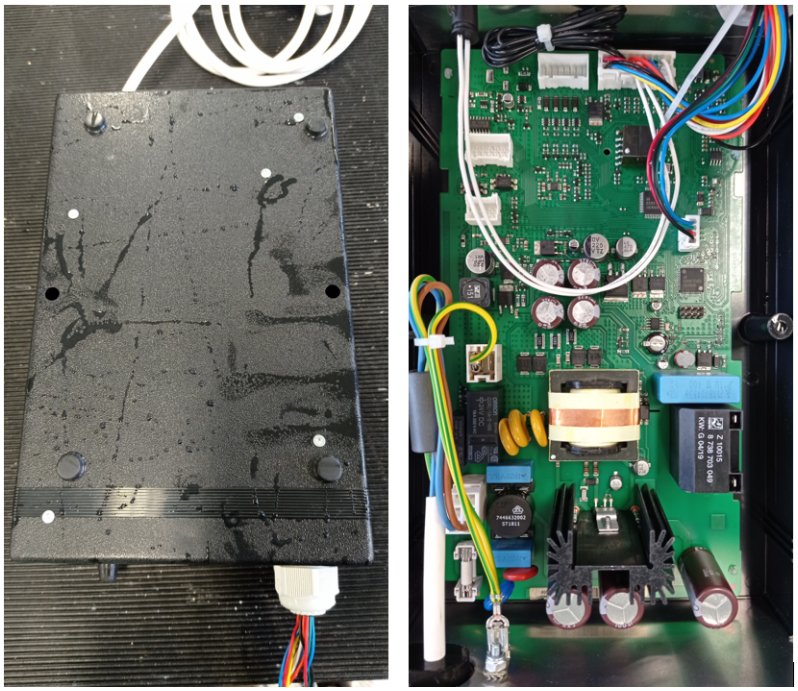

Figura 1.2: Fotografias da solução atual utilizada pela Bosch

colaborador da Bosch deslocar-se a casa do cliente e recolher os dados armazenados no cartão de memória. Isto tem várias desvantagens, pois limita as possibilidades onde estes equipamentos podem ser instalados, visto que têm de estar a distâncias relativamente curtas e acarretam os custos financeiros, temporais e materiais dessas deslocações. Esta solução também permite que o cliente envie os dados armazenados no cartão de memória por email para a Bosch. Contudo, essa via está sujeita à vontade e capacidade do cliente para o fazer. Em alguns casos, a versão atual também permite que sejam enviados al-

guns dados para um servidor da empresa. Porém, essa abordagem não foi implementada de forma escalável e flexível, visto que foi utilizado um servidor que foi construído para outras aplicações.

Para um dos equipamentos foi desenvolvida uma solução que permite extrair alguns dados da ECU e enviá-los para um servidor por WiFi, contudo esta solução foi desenvolvida apenas para esse equipamento e não é adaptável a outros.

Atualmente, os dados recolhidos pelos data loggers são utilizados para calcular um conjunto de métricas, nomeadamente distribuições intervaladas de potência, caudal e temperatura, número de arranques, energia consumida, entre outras. Estas métricas são calculadas pelos colaboradores com os dados que recolhem diretamente dos cartões de memória, tornando a solução dependente da intervenção manual. O objetivo do cálculo destas métricas é definir os perfis de carga dos equipamentos e assim auxiliar no desenvolvimento de planos de teste à durabilidade dos aparelhos. Atualmente, estes perfis de carga são atualizados à medida que os colaboradores recolhem os dados dos cartões de memória, agregam esses dados num conjunto de métricas e fundem-nas aos dados já existentes. Sendo grande parte deste processo realizado manualmente pelos colaboradores, torna-se naturalmente pouco eficiente e pouco escalável.

Visto que a Bosch tem realizado grandes investimentos no desenvolvimento de novos produtos, a necessidade de realização de testes de durabilidade também aumentou. Para a realização desses testes, a empresa precisa de realizar ensaios de campo mas, face às necessidade atuais, a solução que a Bosch atualmente utiliza para fazer o registo de dados não permite cumprir com eficiência as necessidades. Posto isto pretende-se desenvolver uma solução mais eficiente, adaptável, autónoma e escalável.

1.2 Motivação

Ao desenvolver novos produtos, a Bosch necessita de realizar testes de durabilidade aos equipamentos. Estes testes deverão traduzir a realidade da operação dos aparelhos no terreno. Para isso são realizados ensaios de campo, que permitem recolher dados e assim gerar os perfis de carga dos equipamentos, que depois são utilizados para gerar os planos de teste de durabilidade. A principal motivação deste trabalho é a necessidade de realizar mais ensaios de campo e com mais facilidade, poupando recursos humanos, financeiros e de tempo. As características da solução atual dificultam a realização de ensaios de campo e por isso a Bosch necessita de uma solução mais versátil, flexível e que não necessite de tanta invertenção manual.

1.3 Objetivo

O objetivo deste trabalho é recolher de forma remota e inteligente dados relativos à operação de esquentadores no terreno e agregar essa informação num conjunto de métricas. Para isso, pretende-se construir um sistema de data logging, desde a recolha de dados até à visualização de métricas. Para concretizar este objetivo foi traçado um conjunto de metas:

- Seleção e construção de um hardware de aquisição de dados.

- Envio de dados em backend[1] para a cloud.
- Processamento das séries temporais.
- Construção de um frontend[2] para apresentação dos dados em dashboards.

Foi também definido um conjunto de requisitos da solução a implementar:

1. Solução flexível e adaptável a todos os diferentes esquentadores.
2. Sempre que possível deve ser utilizada a ECU para aquisição de dados.
3. Recolha de temperaturas de entrada e saída da água, caudal e pressão hidráulica.
4. Recolha de dados relativos às condições atmosféricas.
5. Registo de dados com a respetiva data e hora absoluta.
6. Taxa de aquisição de 1 s.
7. Armazenamento de dados offline.
8. Permitir colocar/retirar sensores externos.
9. Ser escalável e permitir ligar até dez sensores de temperatura, para poder ser aplicável noutras necessidades que surjam.
10. Medição da corrente consumida nos equipamentos totalmente elétricos.
11. Rearme automático em caso de falha de energia.
12. Capacidade de reestabelecer a conectividade ao servidor.
13. Capacidade de recuperar dados que por algum motivo não sejam enviados para o servidor.
14. Permitir configurações locais, nomeadamente credenciais para a rede wifi, sem a necessidade de reprogramar a solução.
15. Possuir uma interface local que forneça informação do estado de funcionamento do data logger.
16. Permitir definir configurações do sistema de tratamento dos dados remotamente.
17. O processamento deve permitir calcular automaticamente um conjunto de métricas, como distribuições intervaladas, máximos e mínimos, número de ciclos e consumo energético.
18. Uma dashboard para as séries temporais e outra para dados agregados.
19. As dashboards devem permitir fazer o download dos dados.
20. O frontend deve permitir a um utilizador visualizar os dados de forma dinâmica e interativa.

Apesar de não ser um requisito para este trabalho pretende-se também estudar o uso de algoritmos para identificação de outliers nas séries temporais.

[1] Conjunto de operações que correm em segundo plano e não interagem com um utilizador
[2] Serviços que interagem diretamente com o utilizador

1.4 Organização do documento

Este documento está organizado da seguinte forma:

- **Capítulo 1:** introdução do trabalho, com o enquadramento do problema, ponto de partida dos trabalhos, motivação, objetivo e requisitos.
- **Capítulo 2:** apresentação do estado da arte no âmbito do tema proposto, incluindo soluções implementadas por outros autores com objetivo semelhante.
- **Capítulo 3:**: apresentação da solução proposta, com as devidas justificações das decisões tomadas.
- **Capítulo 4:** apresentação da implementação da solução proposta, incluindo as metodologias e abordagens.
- **Capítulo 5:**: análise dos resultados obtidos com os testes realizados à solução final.
- **Capítulo 6:** conclusões retiradas do desenvolvimento dos trabalhos e proposta de trabalhos futuros.

Intentionally blank page.

Capítulo 2

Estado da Arte

Neste capítulo será feita uma revisão do estado do conhecimento e tecnológico atual relativo às temáticas de recolha, processamento e visualização de dados. Visto que este trabalho tem por base a internet das coisas e a computação em cloud, primeiramente serão abordados estes dois conceitos. Também será feita uma revisão de arquiteteturas com objetivos semelhantes e ainda uma análise de algoritmos de deteção de outliers para o tratamento de dados. No final será realizada uma análise critíca de sistemas de *data logging*[1] apresentados por outros autores.

A internet das coisas tem vindo a tornar-se um tema popular para descrever cenários onde a conectividade à internet é também incorporada por uma grande variedade de objetos, dispositivos e sensores [1]. Estando ligados à internet estes dispositivos ficam também conectados entre si, permitindo, por exemplo, uma comunicação remota entre os equipamentos e um utilizador.

O desenvolvimento tecnológico tem possibilitado a conexão de cada vez mais e mais pequenos dispositivos de uma forma barata e fácil [1]. Isto abre portas para um novo mundo de potencialidades, como por exemplo, receber dados de equipamentos instalados no terreno, através de uma aplicação no computador ou smartphone e de uma forma fácil e confortável. Assim, a internet das coisas tem como objetivo facilitar e simplificar processos de trocas de informação entre dispositivos e monitorização e controlo de equipamentos. Aliado a este conceito, a computação em cloud é relevante para complementar as potencialidades da internet das coisas.

A internet das coisas tem vindo a aumentar drasticamente a quantidade de dados gerados que, por sua vez, coloca grande pressão na infraestrutura da internet. A computação em cloud permitiu criar as condições para o armazenamento, análise e tratamento desta grande quantidade de dados. Este conceito aplica-se em muitas situações reais, por exemplo, para uma empresa que recolha grandes quantidades de dados de sensores, os servidores cloud permitem que esses dados sejam enviados e depois processados na cloud, reduzindo assim as necessidades de processamento e energia [2]. A computação em cloud atua como um complemento à internet das coisas. Enquanto a internet das coisas gera grandes quantidades de dados, a computação em cloud permite a navegação, processamento e análise desses dados de uma forma rápida e eficiente. Assim, as duas principais vantagens da utilização de servidores cloud com a internet das coisas são o aumento da performance, visto que grandes quantidades de dados necessitam de mais capacidade de processamento e o aumento da escalabilidade, uma vez que há necessidade

[1]Sistemas que permitem registar dados de outros equipamentos, com o objetivo de posterior análise

de armazenar uma quantidade de dados muito elevada [3].

As tecnologias anteriores têm permitido gerar grandes quantidades de dados, que por sua vez, tem impulsionado os avanços nas temáticas da inteligência artificial. O machine learning é uma forma de inteligência artificial onde, através do desenvolvimento de modelos e algoritmos, os equipamentos têm a capacidade de analisar dados com o objetivo de aprenderem algo que podem utilizar no futuro para melhorar o desempenho de uma determinada tarefa. Isto permite aos equipamentos automaticamente executarem tarefas que não foram diretamente programadas por um utilizador. A capacidade destes sistemas é obtida a partir de modelos de análise que geram previsões, respostas, recomendações e regras [4]. Estas técnicas permitem o desenvolvimento de sistemas inteligentes através da aplicação de algoritmos que podem, por exemplo, ser usados para detetar outliers nos dados recolhidos em sistemas IoT.

2.1 Sistemas de recolha, processamento e visualização de dados

As arquiteturas de soluções de recolha, processamento e visualização de dados incluem várias ferramentas, cada uma com um papel a desempenhar no fluxo de informação desde a criação até à visualização pelo utilizador. Existem algumas abordagens diferentes para caracterizar este tipo de arquiteturas. No artigo "SCoTv2: Large scale data acquisition, processing, and visualization platform" [5], Santiago et al. fazem a divisão deste tipo de arquiteturas num conjunto de blocos, distinguidos pelo papel que desempenham. Os blocos são:

- Criação de dados – bloco responsável por gerar os dados, quer seja através de sensores ou pelo próprio equipamento em estudo.
- Recolha de dados – bloco responsável por fazer a conexão entre os sensores e as camadas superiores da arquitetura. Neste bloco há a recolha dos dados e seguida do envio para um servidor.
- Plataforma IoT – bloco responsável por interligar todos os outros blocos, atuando como framework[2] onde os restantes blocos são implementados.
- Armazenamento de dados – bloco responsável pelo armazenamento dos dados gerados e enviados para o servidor.
- Processamento de dados – bloco responsável pelo controlo do fluxo de dados, que pode ser usado de diferentes formas consoante o objetivo.
- Data Mining – bloco que pode ser visto como uma forma de processamento de dados, com o objetivo de encontrar padrões e aspetos importantes nos conjuntos de dados e retirar informação relevante.
- Visualização de dados – bloco que interage com o utilizador, apresentado-lhe os dados em dashboard e emitindo alertas e avisos.

[2]Espaço de trabalho que inclui um conjunto de funcionalidades que auxiliam no desenvolvimento de aplicações

Esta abordagem permite categorizar a arquitetura da solução a implementar de uma forma intuitiva. Cada bloco tem uma função e aparece numa posição fixa no fluxo de dados.

Uma abordagem diferente é apresentada por Camarneiro [6]. Esta abordagem divide a arquitetura em três níveis distintos:

- Nível físico – neste nível estão presentes os dispositivos de criação de dados, como sensores e equipamentos, e uma gateway, responsável pela recolha dos dados e comunicação com o nível digital.
- Nível digital – neste nível está presente um broker onde são publicadas e subscritas mensagens permitindo a comunicação entre o nível digital e o físico. Além disso, está ainda presente uma base de dados, onde são armazenados os dados enviados pela gateway.
- Nível de serviços – neste nível estão presentes os serviços prestados ao utilizador, nomeadamente as dashboards de visualização de dados e outras funções pertinentes para cada aplicação, sendo que neste caso foi implementada uma REST API, para permitir ao utilizador enviar comandos para o nível digital e depois para o físico.

Esta abordagem permite dividir a arquitetura da solução a implementar em três principais níveis bastante intuitivos, sendo que dentro de cada nível há depois um conjunto de ferramentas com papeis a desempenhar.

Além das anteriores, Coelho [7] propõe ainda uma abordagem ligeiramente diferente. Nesta solução a arquitetura é dividida em:

- Equipamentos – local onde ocorre a criação dos dados a partir de sensores.
- Edge devices – dispositivos responsáveis por recolher os dados dos sensores e enviar para o servidor.
- Servidor – plataforma onde são integrados um conjunto de ferramentas, nomeadamente armazenamento, processamento e visualização de dados.

Esta abordagem permite agrupar as ferramentas da arquitetura consoante a sua função e posição no fluxo de dados. Tem-se um grupo onde é feita a criação dos dados nos equipamentos, um grupo responsável por recolher esses dados e enviar para o servidor, e um grupo (o servidor) onde são integradas várias ferramentas.

2.1.1 Criação de dados

A base da arquitetura a implementar é a criação dos dados, pois para que as restantes camadas possam funcionar, necessitam dos dados gerados neste bloco. Para criar dados podem ser usados sensores externos, aplicados nos aparelhos, ou o sistema eletrónico do próprio equipamento. Neste trabalho, para a comunicação com os equipamentos será obrigatoriamente utilizado um protocolo próprio da Bosch e por isso essa temática não será abordada neste capítulo.

Para fazer a aquisição de alguns dados serão utilizados sensores, os quais podem ser divididos em dois grupos, os digitais e os analógicos. Os sensores analógicos tem um output que corresponde a um sinal de tensão que varia consoante grandeza que se pretende

medir. Normalmente necessitam de curvas de calibração e conversores analógico-digitais. Os digitais possuem uma interface de comunicação SPI ou I2C através da qual se faz a leitura da grandeza que se pretende medir. Estes sensores ficam sujeitos às vantagens e limitações destes protocolos de comunicação.

Um dos principais tipos de sensores relevantes para este trabalho é o sensor de temperatura. Existem essencialmente dois tipos, termopares e termístores. Nos termopares, o parâmetro que depende da temperatura é a tensão induzida nos seus terminais, já nos termístores a temperatura tem influência na resistência elétrica. Estes últimos podem ser NTC (resistência diminui com o aumento de temperatura) ou PTC (resistência aumenta com o aumento de temperatura). Para fazer a leitura da temperatura será necessário implementar um circuito que permita passar de um output fornecido pelo sensor para uma grandeza que possa ser interpretada por um microcontrolador (MCU). No caso dos termopares, o output fornecido pelo sensor em mV, deve ser amplificado através de um amplificador. Assim, haverá uma tensão analógica que pode ser lida por um MCU. Para que isto ocorra, é necessário que o MCU realize a conversão A/D desse sinal, e posteriormente efetue cálculos para chegar à temperatura. No caso dos termístores pode ser utilizado um divisor resistivo onde o MCU faz a conversão A/D da tensão aos terminais do sensor, cujo valor irá variar à medida que a variação de temperatura altera a resistência elétrica do sensor. Através da curva de calibração do sensor é possível passar o valor digital para a temperatura.

Existe também a possibilidade de utilizar circuitos integrados digitais em placas PCB (placas de circuito impresso) que façam a recolha e tratamento dos sinais. Existem, tanto para termístores como para termopares, placas que comunicam com o MCU através de protocolos de comunicação I2C ou SPI e que já fazem por si a calibração. Nestes casos, o MCU pode solicitar ao sensor a temperatura. Um exemplo deste tipo de sensor é o MAX6675, que possui uma placa com uma interface SPI através da qual é possível recolher a temperatura do respetivo termopar.

A escolha do sensor de temperatura depende dos requisitos da aplicação, nomeadamente grau de precisão, tempo de resposta, temperaturas máximas e mínimas e disponibilidade de recursos. Para aplicações com temperaturas entre -40 e 125^{o}C os sensores NTC são uma boa opção, pelo seu baixo custo, flexibilidade, facilidade de integração em sistemas e baixo consumo de energia [8]. Para aplicações mais exigentes será necessário utilizar termopares. Algo importante a considerar é a qualidade da conversão analógica dos MCUs. Morais apresenta um conjunto de testes realizados com a ADC (conversão analógica digital) do ESP32 e chegou à conclusão de que a ADC do ESP32 tem alguns problemas de estabilidade e exatidão [9]. Se for necessária uma conversão A/D com mais qualidade, existem placas ADC externas que, de acordo com o mesmo autor, fornecem uma conversão A/D muito melhor que os sistemas embutidos nos MCUs. Estas placas fazem a conversão A/D e depois através de protocolos de comunicação I2C ou SPI é possível "pedir" os valores digitais, com um grau de precisão e estabilidade muito maior. Um exemplo deste tipo de dispositivo é a placa Adafruit ADS1115[3], que possui uma interface I2C. Cada uma destas placas permite implementar quatro sensores analógicos. Como a comunicação é feita por I2C, cada placa terá de ter um endereço I2C diferente, caso sejam ligadas no mesmo barramento. Para resolver esta questão, esta placa pode ser ligada de quatro formas distintas, sendo que cada uma possuirá um endereço diferente.

[3]https://www.best-microcontroller-projects.com/ads1115.html

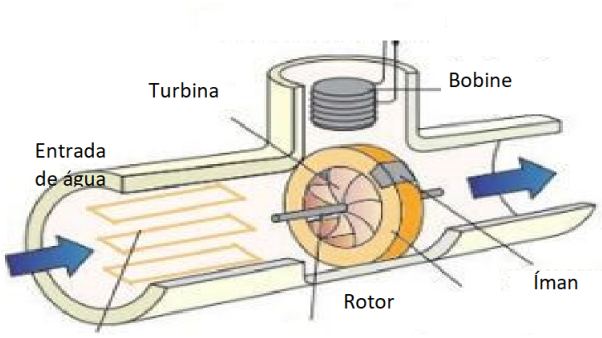

Figura 2.1: Esquema de funcionamento de um caudalímetro de efeito de Hall, adaptado do artigo "Turbine Flow Meter: Technology and Working Principle" do fabricante *Eltra trade*

Deste modo podem ser ligadas quatro placas, sendo que no total serão dezasseis sensores analógicos por barramento I2C.

Outro sensor importante é o caudalímetro, sendo que uma possibilidade é o caudalímetro de efeito de Hall. Este dispositivo incorpora um rotor que gira à medida que a água passa pela turbina. O segundo elemento é o sensor de efeito hall, que gera tensão a partir da deteção de um campo magnético. Como o rotor tem um íman, à medida que gira permite ao sensor de efeito Hall gerar uma onda quadrada de frequência proporcional ao caudal de água que passa pelo sensor. Com um MCU é possível contabilizar os pulsos gerados pelo sensor de modo a saber o caudal de água que passa no equipamento. Na figura 2.1 [10] é possível visualizar o princípio de funcionamento de um caudalímetro de efeito de Hall.

Para detetar ciclos on/off em equipamentos elétricos pode ser utilizado um sensor de corrente. Este sensor permite detetar a passagem de corrente na resistência do equipamento e assim saber quando é ligado. Para este tipo de sensor, o príncipio de funcionamento também se baseia no efeito de Hall. Contudo, difere na forma como é gerado o campo magnético que irá induzir tensão na bobine do sensor. Tal como está representado na figura 2.2 [11] à medida que a corrente passa por um condutor, forma-se um campo magnético num núcleo magnético. Através do sensor de Hall será induzida uma tensão proporcional à intensidade do campo magnético, que por sua vez também é proporcional à corrente que passa pelo condutor. Este sinal de saída é depois amplificado para poder ser lido por um MCU.

Para a caracterização completa dos perfis de carga dos esquentadores é importante ainda monitorizar as condições atmosféricas do ambiente em que estes estão instalados. Para concretizar este objetivo o sensor BME280, desenvolvido pela Bosch, tem várias vantagens. É um sensor muito compacto, com baixo consumo energético e que permite medir todas as condições pretendidas, nomeadamente a temperatura do ar, a pressão atmosférica e a humidade relativa. Este sensor possui uma interface SPI e I2C, pelo que também é bastante versátil ao nível da comunicação com o MCU.

2.1.2 Aquisição de dados

A aquisição de dados é o processo de recolha das medições realizadas pelos sensores, através de controladores. Os controladores devem ser programados para ler os sinais pro-

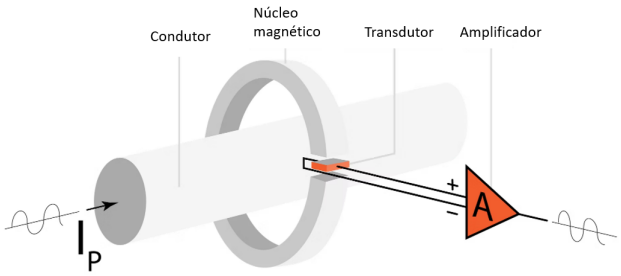

Figura 2.2: Princípio de efeito de Hall nos sensores de corrente, adaptado do artigo "How To Measure Current Using Current Transducer" de *Grant Maloy Smith*

venientes dos sensores, quer sejam analógicos, fazendo uma conversão analógica digital, ou digitais, comunicando por I2C ou SPI. Este componente deve ser programado para enviar os dados recolhidos para o servidor, através do protocolo de comunicação definido para esse efeito. A pesquisa bibliográfica revelou que existem soluções implementadas por outros autores com vários controladores diferentes, desde microcontroladores Arduino ou ESP a microcomputadores Raspberry Pi. Foi realizado um estudo comparativo entre os microcontroladores Arduino UNO, ESP32 e o microcomputador Raspberry Pi 4. Um dos critérios mais diferenciadores entre estes é o processador, sendo que o Arduino Uno tem um processador single-core com 16 MHz, enquanto que o ESP32 tem um processador dual-core com 160 MHz, e o Raspberry Pi 4 tem um processador quad-core com 1500 MHz. Deste modo, as três opções anteriores estão em níveis completamente diferentes. Outra questão bastante relevante é o número de pinos GPIO e de entradas analógicas. Por exemplo, apesar de o Raspberry Pi 4 ter muitos pinos GPIO, não tem canais de conversão analógica. Por outro lado, o ESP32 tem 18 canais analógicas, mas se estiver ligado a uma rede WiFi só pode usar 6 deles. As interfaces de comunicação não são um critério diferenciador, visto que todos podem comunicar por I2C, SPI e UART. O Arduino Uno apresenta uma desvantagem, pois necessita de um módulo WiFi externo. Das três opções, aquela que tem custo mais baixo e dimensões mais reduzidas será o ESP32. O número de núcleos do processador do dispositivo selecionado tem uma importância bastante significativa, pois permite uma gestão eficiente das tarefas que devem ser executadas.

2.1.3 Plataforma IoT

As plataformas IoT são ferramentas na cloud que ofereçem um conjunto de serviços como o armazenamento e processamento de dados, a monitorização remota e o suporte no processo de tomada de decisão. Nestas plataformas há vários dispositivos conectados pela internet que permitem a troca de informação sem interferência humana [12]. Algumas das principais são: Smart Cloud of Things (SCoT), Amazon Web Services, Microsoft Azure, IBM Cloud e Google Cloud Platform. Como no contexto deste trabalho, a empresa pretende que seja utilizada a plataforma Microsoft Azure, será realizada uma revisão mais aprofundada desta plataforma. A Azure é uma plataforma que fornece um conjunto de serviços de computação, análise de dados, armazenamento e networking permitindo a integração e desenvolvimento de aplicações. Um dos principais usos desta plataforma é a utilização de *Virtual Machines* que correm na cloud. Através da instalação de uma

Virtual Machine na Azure é possível ter na cloud os blocos de armazenamento, processamento e visualização de dados tal como seria possível num computador. Assim será possível passar uma arquitetura com as ferramentas desejadas para a cloud, escalando a solução para um nível remoto e distribuído por todo o planeta. A Azure também oferece um conjunto de serviços de processamento e visualização de dados. No entanto, por serem dispendiosos, esses serviços não serão utilizados neste trabalho.

2.1.4 Envio de dados para o servidor

O envio de dados para o servidor pode ser realizado através de vários protocolos de comunicação. No artigo "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP", Naik faz a revisão dos quatro protocolos mais conhecidos e utilizados: MQTT, CoAP, AMQP e HTTP [13]. Pela compatibilidade com os requisitos deste trabalho foram avaliados os protocolos MQTT e HTTP, amplamente utilizados [6, 7, 14–17]. Com base nas referências mencionadas, fez-se um resumo das principais características de ambos os protocolos.

- Protocolo MQTT - Este protocolo baseia-se numa comunicação do tipo publicar/subscrever. Os clientes publicam e subscrevem tópicos num broker. Neste protocolo, o transporte de dados é feito através do protocolo TCP/IP e a segurança é feita por TLS/SSL. Este protocolo é adequado para redes de muitos e pequenos equipamentos que têm de ser monitorizados através de um servidor em backend. É um protocolo com poucas opções de controlo, não havendo possibilidade de comunicação device-device nem mensagens de multicast[4].

- Protocolo HTTP - Apresenta uma comunicação do tipo pedido/resposta com recurso a URL. Usa o protocolo TCP/IP para controlo do fluxo de dados e TLS/SSL para segurança. É um protocolo standard bem estebelecido, permite conexões persistentes e a segurança é eficaz. Para a comunicação entre muitos equipamentos de baixo processamento, este protocolo pode provocar muito tráfego na rede, pois os equipamentos ficam ocupados durante as trocas de mensagens. Não é um protocolo desenvolvido para comunicações IoT, pelo que a forma de comunicação não é a mais eficiente.

Com base na análise de vários trabalhos realizados por outros autores, concluiu-se que o protocolo mais utilizado para permitir a vários clientes enviar dados para um mesmo servidor é o MQTT [6, 14–16]. Múltiplos estudos demonstraram a superioridade deste protocolo relativamente aos restantes, tanto ao nível da velocidade como peso no sistema [18, 19]. No entanto, para o desenvolvimento de uma interface local de baixo tráfego de dados onde um instalador pretende configurar um data logger, o protocolo HTTP é mais adequado. Este protocolo consiste numa comunicação direta entre dois agentes, com pedido e resposta, o que torna a configuração do data logger mais eficaz e o desenvolvimento desta interface mais simples (visto não ser necessário um terceiro agente intermediário).

[4]Um dispositivo envia uma mensagem para vários dispositivos em simultâneo

2.1.5 Bases de dados

Na arquitetura deste tipo de aplicações deverá haver uma base de dados (BD) para permitir guardar grandes quantidades de informação, obter históricos de funcionamento e processar grandes quantidades de dados. Existem, no entanto, três grandes grupos de BDs [6]:

- Relacionais - Os dados são organizados na forma de tabelas, linhas e colunas e os dados no interior de cada linha estão relacionados. Os comandos para manipular a base de dados baseiam-se na linguagem SQL (Structured Query Language) e o principal exemplo é a BD MySQL. A limitação deste tipo de BD está na inflexibilidade da estrutura. As tabelas têm de ser planeadas e as colunas têm de ser preenchidas. Para situações em que há grande variação nos dados este tipo de BD é difícil de implementar. Qualquer modificação no sistema obriga à interferência complexa na BD. Deste modo, este tipo de BD é pouco flexível e pouco escalável, sendo apenas adequadas para modelos de dados pré-definidos e que variam pouco.

- Bases de dados não relacionais - Nestas BD os dados são guardados de uma forma não relacional, utilizando estruturas como documentos. Cada documento pode ter diferentes formatos, quantidades e tipos de dados. Os dados em cada documento não estão relacionados. Esta solução é mais adequada para aplicações cujos conjuntos de dados variam bastante e não estão relacionados, pois a base de dados irá armazenar esses documentos de forma independente. Este tipo de BD é flexível, dinâmica e escalável permitindo o armazenamento data sets que variam frequentemente e que não estão relacionados. Uma opção deste tipo seria a MongoDB, que é uma base de dados não relacional que armazena os dados na forma de documentos JSON.

- Bases de dados de séries temporais - Estas bases de dados são otimizadas para a gestão de séries temporais. Trabalham essencialmente com dados que têm uma componente relacionada com o tempo, permitindo funcionalidades de agregação de dados com espaçamentos temporais significativos com muita eficiência. Este tipo de BD permite também definir períodos de retenção em que consoante o tipo e a importância dos dados pode-se definir que ao fim de um determinado tempo são automaticamente apagados. A principal base de dados deste tipo é a InfluxDB. Esta opção apresenta grande flexbilidade de armazenamento de dados e permite uma gestão muito eficiente dos dados ao nível temporal.

As séries temporais serão uma grande parte dos dados que serão utilizados neste trabalho. Será importante realizar inúmeras querys à BD em que o tempo será uma variável central. O InfluxDB proporciona uma linguagem própria para efetuar queries designada Flux. Esta linguagem permite uma vasta gama de funções que permitem gerir a BD de qualquer forma pretendida.

2.1.6 Visualização de dados

Neste trabalho pretende-se essencialmente visualização de dados através da WEB, sendo que o tipo de dados irá incluir séries temporais e métricas. Para este efeito, a principal ferramenta que pode ser utilizada é o Grafana. Esta plataforma open-source permite

aceder a uma vasta gama de fontes de dados para os apresentar na forma de dashboards. Esta ferramenta permite ainda efetuar o processamento dos dados entre a BD e a dashboard, nomeadamente cálculo de médias, máximos, mínimos, agregação, entre outras. O Grafana pode ser instalado num computador local ou correr na cloud. Outras opções incluem o Kibana, que é bastante semelhante ao Grafana, ao Power BI, que é um serviço prestado pela Microsoft Azure mas que acarreta grandes custos e ao Node-Red dashboard, que é a ferramenta de visualização de dados própria da plataforma Node-Red.

2.1.7 Processamento de dados

O processamento de dados é o bloco responsável pela filtragem, preenchimento de dados em falta, agregação, análise, deteção de anomalias, machine learning e fusão de dados. No artigo "An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques" [20], Krishnamurthi et al. propõem uma divisão das funcionalidades de tratamento de dados IoT em três níveis (ver figura 2.3). Os três níveis representados na figura 2.3 e

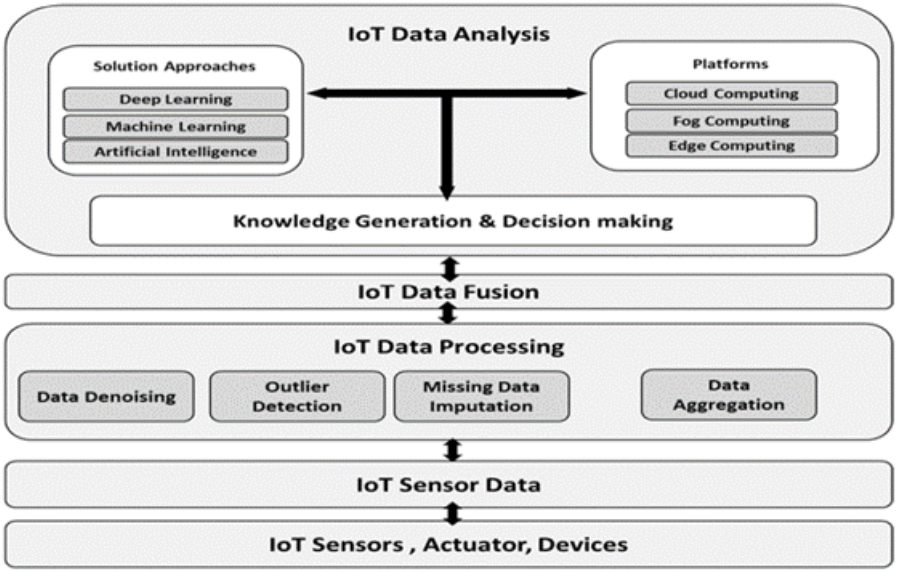

Figura 2.3: Arquitetura de processamento de dados em aplicações IoT, retirado do artigo "An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques", de Krishnamurthi et al.

propostos pelos autores são:

- O primeiro nível consiste no pré-processamento de dados e pode ser utilizado para filtragem, para remover a componente de ruído nos dados, inserir dados em falta, porque em aplicações IoT é bastante comum haver dados incompletos, deteção de outliers, porque os sensores são muito suscetíveis a fatores externos e podem produzir dados anormais [21] e agregação, com o objetivo de extrair a informação relevante a partir de uma grande quantidade de daods.

- O segundo nível é a fusão de dados, que consiste na integração de dados provenientes de várias fontes para fornecer maior confiança e fiabilidade nos dados.

- O último nível consiste na análise dos dados resultantes dos níveis anteriores, através de modelos de machine learning e deep learning para gerar conhecimento relevante e suportar tomadas de decisão.

A agregação de dados é uma forma de processamento cujo princípio é conseguir extrair informação útil e relevante, a partir de uma grande quantidade de dados, através de um conjunto de cálculos matemáticos [20, 22]. Visto que um dos pontos mais importantes deste trabalho é o cálculo de métricas estatísticas a partir dos dados recolhidos, conclui-se que será obrigatória a implementação de um processamento que consista na agregação de dados. Para que o cálculo das métricas, através da agregação das séries temporais seja o mais precisa possível, é necessário aplicar algumas das técnicas apresentadas na figura 2.3, nomeadamente o preenchimento dos dados em falta e a deteção de outliers.

2.1.8 Algoritmos de deteção de outliers

O desenvolvimento tecnológico tem vindo a aumentar a quantidade de dispositivos conectados nos sistemas IoT, aumentando também a quantidade de dados que são depois analisados. No passado, era relativamente fácil um utilizador visualizar um conjunto de dados e identificar falhas e padrões. Contudo após a mudança de paradigma, esta tarefa tornou-se demasiado complexa, criando a necessidade de utilização dos algoritmos de deteção de anomalias, para auxiliar o utilizador [23]. As anomalias podem ser vistas como observações que se desviam significativamente das restantes, gerando a suspeita de que foram obtidas por um mecanismo diferente [24]. No contexto IoT, as anomalias são consequências de uma mudança indesejada no estado de um sistema para fora da normalidade [23]. Estas anomalias podem afetar análises futuras efetuadas nos dados recolhidos pelo que a deteção das mesmas é importante.

Nas aplicações IoT existem dois contextos possíveis de deteção de anomalias. Podem ser analisados dados que dizem respeito apenas a uma só variável, também designados por *Univariate time series Data*, ou dados que representem um conjunto de variáveis, também designados por *Multivariate Time-Series Data* [23].

No contexto deste trabalho, existirão vários sensores a monitorizar os esquentadores. Assim, analisando os dados através de uma abordagem *Multivariate Time-Series Data* dever-se-á obter um modelo mais preciso e fiável, permitindo detetar anomalias que não seriam encontradas analisando os dados individualmente. Por vezes é importante realizar um tratamento dos dados antes de passarem pelos algoritmo de deteção de outliers. Uma abordagem referida por Cook et al., para este tipo de situações é "Dimensionality Reduction" [23]. Este conceito deriva do facto de que muitas vezes, quando se tem vários sensores a monitorizar um equipamento, ser possível definir uma relação entre as variáveis associadas aos sensores. Desse modo, é possível reduzir a quantidade de dados a serem analisados ao reduzir um conjunto de variáveis dependentes a uma única variável que englobe todas as outras. No contexto deste trabalho, este princípio pode ser utilizado para reduzir as variáveis: caudal, temperatura de entrada e temperatura de saída da água à potência do equipamento, através da equação 2.1.

$$Q = mC_p\Delta T \tag{2.1}$$

Realizado o tratamento dos dados, pode avançar-se para a utilização de algoritmos para a deteção de outliers. Estes algoritmos podem ser classificados como supervisionados

e não supervisionados. No caso dos supervisionados é necessária intervenção humana, visto que os dados são classificados, por exemplo como "OK" ou "NOT OK". Neste caso, o algoritmo é treinado usando estas classificações e irá depois permitir indicar se um determinado ponto é um outlier ou não com base nesse treino. A vantagem deste tipo de algoritmo é ser mais preciso, no entanto necessita que os dados sejam previamente classificados. Por outro lado, os algoritmos não supervisionados apenas precisam que lhes seja fornecido o conjunto de dados, sem qualquer classificação. Neste caso, o algoritmo irá procurar por padrões escondidos no conjunto de dados, sem necessidade de intervenção humana. Este tipo de algoritmo tem a vantagem de não necessitar de tratamento de dados prévio, no entanto é importante a validação dos outputs fornecidos pelo mesmo. A seleção do algoritmo mais indicado para uma determinada aplicação depende muito do tipo de dados que se pretende analisar e dos objetivos da aplicação. Por exemplo, para aplicações em que se pretendam analisar grandes volumes de dados praticamente em tempo real é muito mais eficiente utilizar algoritmos não supervisionados. Por outro lado, se se pretender analisar dados com requisitos de alta fiabilidade e com a possibilidade de classificação prévia dos dados então um algoritmo supervisionado será uma opção melhor. Alguns dos algoritmos de deteção de outliers mais utilizados são:

- Angle Based Outlier Detection [25]: Este método consite em avaliar os vetores distâncias entre pontos, não só ao nível da distância mas também das direções dos vetores. Para os outliers será de esperar que a variação entre ângulos formados pelos vetores distâncias seja pequena visto que os restantes pontos se encontram agrupados em clusters. Pelo mesmo motivo para os pontos que não são outliers haverá uma grande variação dos ângulos. Este princípio está representado na figura 2.4.

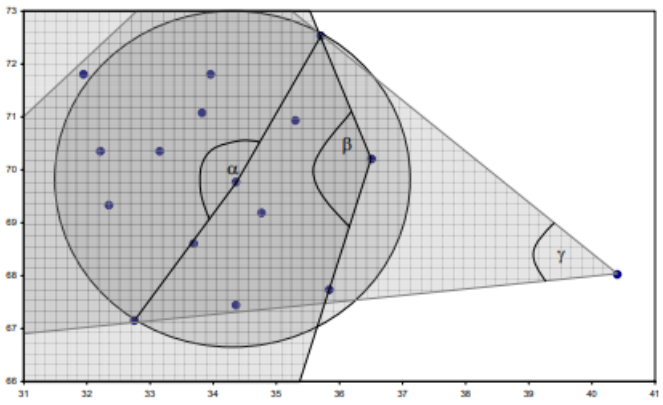

Figura 2.4: Princípio do Angle Based Outlier Detection, retirado do "Angle-based outlier detection in high-dimensional data" artigo de Kriegel et al.

- Stochastic Outlier Selection [26]: Este método consiste em estabelecer a afinidade entre pontos de um determinado conjunto de dados. A partir dessa afinidade é definida para cada ponto a probabilidade de esse ponto ser um outlier. A afinidade entre pontos é proporcional ao grau de semelhança entre eles. No mesmo artigo o autor apresenta resultados obtidos com a utilização deste algoritmo que revelam que

pode ter uma performance superior a muitos outros algoritmos bem estabelecidos em muitos cenários diferentes.

- Copula-Based Outlier Detection [27]: Este algoritmo permite fornecer como input um conjunto de dados multidimensional e como output um vetor que fornece a probabilidade relativa de cada ponto ser outlier. O output deverá ser intepretado de forma relativa e não absoluta. O princípio utilizado neste método é baseado nas funções Cópulas, que permitem codificar a dependência entre as variáveis. Depois de definir as funções cópulas é calculada, nessas funções, a probabilidade de se observar um ponto tão extremo como cada um dos pontos do conjunto de dados. Se um determinado ponto for outlier, a probabilidade de ser observado na cópula será menor, pelo que assim se determina a probabilidade relativa de cada ponto ser outlier.

- Isolation Forest [28]: Este método baseia-se num conjunto de elementos que são as "Isolation Trees", responsáveis por isolar os pontos de um determinado conjunto de dados. De forma aleatória, os pontos vão sendo progressivamente isolados e, assumindo que as anomalias são pontos que aparecem apenas ocasionalmente e estão afastados dos restantes, é de esperar que sejam isolados mais rapidamente. Desta forma o algoritmo consegue detetar os pontos mais isolados e que mais provavelmente são anomalias. No final do processo ao conjunto de "isolation trees" dispostas aleatoriamente dá-se o nome de "isolation forest".

Coelho et al [7], utiliza uma abordagem baseada na redução dimensional, para detetar anomalias no funcionamento de equipamentos industriais. Realizaram um estudo comparativo entre três dos algoritmos anteriores, cujos resultados revelaram que o COPOD obteve melhor desempenho que o ABOD e o SOS, com base em critérios de precisão. Mendes et al [29] realizaram um estudo comparativo entre o Isolation Forest e outros algoritmos de Deep Learning e chegaram à conclusão de que o Isolation Forest será uma solução mais robusta e facilmente adaptável a vários tipos de cenários. Os resultados obtidos pelos autores revelaram que para diferentes cenários o Isolation Forest fornecia resultados geralmente mais estáveis.

2.1.9 Preenchimento de dados em falta

Para o cálculo de métricas, um conjunto de dados incompleto pode-se traduzir em resultados pouco precisos e que levarão a tomadas de decisão erradas. Por esse motivo o preenchimento de dados em falta é uma tarefa de pré-processamento tão importante. Para isto podem ser utilizadas várias técnicas, desde simples cálculos de médias e atribuição de valores seguintes ou anteriores, até interpolações ou mesmo algoritmos de machine learning. No artigo "A Comprehensive Survey on Imputation of Missing Data in Internet of Things" [30], Deepak et al. fazem uma revisão das várias técnicas para lidar com conjuntos de dados incompletos. Algumas das mais relevantes são:

- **Atribuição baseada na média:** esta técnica é bastante simples e utiliza a média entre alguns dados existentes para atribuir valores aos dados em falta. Esta técnica, embora simples de implementar, pode quebrar a tedência dos dados e criar anomalias no conjunto de dados.

- **Atribuição baseada em interpolação:** esta técnica considera os pontos imediatamente antes e depois dos dados em falta, atribuindo o valor em falta através da média ponderada entre esses valores. Esta técnica permite obter resultados razoáveis quando o "furo" de dados é curto. À medida que o número de dados consecutivos for maior, a precisão será menor.

- **K-Nearest Neighbor:** é um dos algoritmos de machine learning mais usados para preenchimento de dados em falta. Esta técnica consiste em procurar pelos K vizinhos mais próximos que tenham dados completos, com base em critérios de distância. Depois é calculada a média entre esses vizinhos para que esses valores sejam atribuidos ao valor em falta. Um estudo realizado com o objetivo de comparar o desempenho deste algoritmo com outras estratégias para imputação de valores em séries temporais revelou que este algoritmo é bastante superior [31].

Para definir a estratégia para preencher dados em falta é importante saber qual o tipo de dados a repôr. As duas principais possibilidades são: inexistência de uma linha completa de dados, caso o conjunto de dados tenha várias variáveis e inexistência de apenas um ou mais valores dentro de uma linha de dados. As melhores estratégias para lidar com cada situação podem ser diferentes. Por exemplo, no caso do algoritmo KNN, caso haja uma linha de dados completa em falta, provavelmente será mais difícil determinar os vizinhos mais próximos com precisão e nesse caso a utilização de uma interpolação poderá ser interessante. Caso apenas falte um elemento no interior de uma linha, o algoritmo KNN será muito mais eficaz.

2.2 Trabalhos desenvolvidos por outros autores

Com vista a perceber de que forma a solução a desenvolver neste trabalho permitirá acrescentar algo ao estado de conhecimento foi importante conhecer os sistemas de data logging desenvolvidos por outros autores com objetivos semelhantes.

2.2.1 Remote datalogging of solar UV irradiation using open-source ESP32 platform and MQTT protocol

Neste trabalho [14] é implementado um sistema de data logging que utiliza as potencialidades da internet das coisas para criar uma solução remota. Os autores utilizaram um microcontrolador ESP32 para permitir enviar dados de sensores de luz UV para um servidor pelo protocolo MQTT, via WiFi. Do lado do servidor, os autores utilizaram uma máquina virtual Linux instalada num computador local onde foi instalado o broker Mosquitto, a base de dados InfluxDB e o Grafana.

Os resultados relevaram que o envio de dados para o servidor foi realizado com boa performance utilizando o protocolo MQTT. A solução desenvolvida permite a visualização dos dados no Grafana em tempo real sem haver perdas de dados.

Apesar das vantagens referidas, esta solução apresenta algumas limitações:

- Servidor instalado num computador local, o que limita a escalabilidade da solução para distribuição de vários data logger remotamente.

- Não possui um sistema de armazenamento offline além da memória do microcontrolador, o que limita a capacidade da solução de gerir falhas de rede WiFi.

- O frontend desenvolvido não está adaptado para integrar um sistema de vários data loggers de forma dinâmica.

- A solução não possui uma forma de interagir com o microcontrolador sem ser reprogramando o ESP32.

2.2.2 Low-Cost ESP32, Raspberry Pi, Node-Red, and MQTT Protocol Based SCADA System

Este artigo [15] apresenta uma solução de registo de dados utilizando comunicação wireless. O sistema utiliza um conjunto de sensores de luz e temperatura ambiente em conjunto com um ESP32. O microcontrolador recolhe os dados dos sensores e envia por WiFi para um servidor local instalado num Raspberry Pi. Para a comunicação entre o ESP32 e o servidor utiliza o protocolo MQTT e a plataforma para a gestão dos dados no servidor é o NodeRed. Para permitir apresentar os dados em dashboards é utilizado também o NodeRed.

Esta solução apresenta algumas vantagens, nomeadamente utilizar uma arquitetura bastante eficiente, com um esp32 e Raspberry Pi. A forma de implementação permite a instalação de vários data loggers a enviarem dados para o mesmo servidor instalado num só Raspberry Pi. Porém, esta solução tem algumas limitações ao nível da escalabilidade. Como o servidor é instalado localmente, todos os dispositivos têm de estar ligados no mesmo router para poder comunicar. Além disto em caso de falhas de connectividade não há qualquer recolha de dados, nem forma de recuperação de dados que por algum motivo não sejam enviados, tornando solução muito suscetível a falhas.

2.2.3 IoT and Cloud Based Remote Monitoring of Wind Turbine

Este trabalho [16] apresenta uma solução de registo remoto de dados que utiliza uma arquitetura baseada na internet das coisas e na computação na cloud para monitorizar remotamente turbinas eólicas. Esta solução inclui um conjunto de sensores e um Raspberry Pi que recolhe os dados dos sensores e por WiFi envia os dados para a plataforma Azure. Na plataforma é utilizado um serviço, Power Bi, para apresentar os dados na forma de dashboards. Esta solução tem várias vantagens, sendo que a principal é utilizar a plataforma Azure de modo a permitir instalar múltiplos data loggers remotamente, sendo que todos eles podem enviar os dados para o mesmo servidor. Além disso, o facto de utilizar um Raspberry Pi fornece alta performance na recolha de dados e envio para o servidor. Este dispositivo também permite que haja armazenamento de dados offline, o que torna a solução mais robusta em caso de falhas de rede WiFi. A principal limitação desta solução está no alto custo. Não só o Raspberry Pi é uma opção dispendiosa, relativamente a outras possibilidades, como os vários serviços utilizadas na Azure também trazem grandes custos. Outra limitação desta solução está na interação local com um instalador. Se houver necessidade de configurar credenciais WiFi, por exemplo, será necessário reprogramar o Raspberry Pi. Esta solução também não prevê formas de processamento que incluam cálculos estatísticos ou deteção de outliers.

2.2.4 Low-Cost Data Acquisition System for Solar Thermal Collectors

Este trabalho [17] utiliza uma arquitetura de baixo custo com recurso a "Docker containers" para fazer o registo de dados de coletores solares. Esta arquitetura inclui um microcontrolador ESP32 que recolhe dados de vários sensores. Em backend, o ESP32 envia os dados por MQTT para um broker Mosquitto e através de um ambiente desenvolvido em NodeRed os dados são depois armazenados numa base de dados MySQL. Em frontend, os dados são apresentados ao utilizador através do grafana. Para implementar as camadas a montante do ESP32, o autor utilizou Docker Containers. Esta solução apresenta várias vantagens, nomeadamente utilizar Docker Containers que permitem fornecer alta performance à solução e também permitir escalar a solução para recolher dados de múltiplos data loggers distribuídos remotamente. Algumas limitações desta solução são:

- Não inclui processamento dos dados recolhidos, quer seja com fins estatísticos, deteção de outliers ou auxílio na tomada de decisão.
- A base de dados MySQL não está otimizada para gestão de séries temporais, nem fornece boa flexibilidade.
- Não inclui mecanismos de gestão de falhas WiFi para prevenir perdas de dados.

2.2.5 Low-cost data acquisition systems for photovoltaic system monitoring and usage statistics

Este trabalho [32] apresenta uma solução de baixo custo para a recolha de dados de paineis fotovoltaicos em locais remotos. A solução inclui um data logger composto por um microcontrolador, uma placa de comunicação com um cartão SD, uma interface composta por Leds e botões e um conjunto de sensores. Esta solução tem a vantagem de poder ser instalada em qualquer lugar sem haver necessidade de redes WiFi ou servidores. Todavia, esta vantagem também é uma limitação, visto que é impossível saber o estado de funcionamento do data logger ou mesmo ter acesso aos dados sem uma deslocação ao local. Os dados recolhidos são registados num cartão de memória e depois podem ser utilizados para fins estatísticos, no entanto é necessário que um utilizador faça a recolha desses dados localmente e terá de fazer manualmente os cálculos estatísticos. Esta é uma solução desenvolvida para situações muito particulares e que não usufrui de muitas das potencialidades da tecnologia atual.

Com base na análise das soluções de data logging implementadas por outros autores conclui-se que as abordagens adotadas diferem bastante. Algumas soluções utilizam servidores instalados localmente em computadores, outras utilizam uma cloud, enquanto outras fazem apenas o registo de dados em cartão de memória. A utilização da cloud é a opção mais versátil e escalável caso haja acesso a redes wifi nos locais de registo de dados. Ao nível da forma como os dados recolhidos são processados, raramente é implementada uma forma de processamento autónoma dos dados por parte do sistema. Caso haja necessidade de realizar cálculos estatísticos, normalmente são realizados manualmente, o que também é uma limitação das soluções atuais. Ao nível da gestão das tarefas executadas pelos microcontroladores, nenhuma solução analisada prevê mecanismos de gestão de falhas, para prevenir perdas de dados por má conectividade ou mesmo períodos sem ligação à internet. Esta é uma questão muito importante para tornar a solução mais robusta e fiável. Por último, uma limitação das soluções atuais está na forma como lhes

são fornecidos parâmetros de configuração, como por exemplo credenciais wifi. Nas soluções analisadas essa questão não é prevista e isso obriga a que haja uma reprogramação do microcontrolador, fornecendo as credenciais através do código. Também seria uma mais valia para um sistema de data logging se pudesse ser trocado de lugar e facilmente houvesse forma de configurar a rede wifi sem necessidade de reprogramar o microcontrolador. Estas várias limitações encontradas nas soluções atuais serão um foco importante da solução que será proposta neste trabalho, sendo que o objetivo será colmatar todas estas falhas.

Capítulo 3

Solução proposta

Tendo por base os requisitos, objetivo e a revisão do estado da arte definiu-se uma solução conceptual a propor para este trabalho. Com o conceito proposto pretende-se cumprir com todos os requisitos e colmatar as limitações identificadas nos data loggers analisados nos capítulos anteriores.

3.1 Estrutura da solução

O conceito proposto divide-se em cinco grupos:

1. **Equipamento:** engloba todos os componentes que são aplicados nos equipamentos com o objetivo de criar dados, nomeadamente sensores externos e barramento de comunicação com as ECUs;

2. **Data Logger:** grupo responsável por fazer a conexão entre o equipamento e o servidor, cujas principais tarefas serão recolher os dados e enviá-los para o servidor;

3. **Servidor:** grupo onde são integrados um conjunto de funcionalidades de gestão da solução, como a base de dados e o processamento;

4. **Interface utilizador:** neste grupo são incluidos os serviços prestados ao utilizador final, nomeadamente apresentação dos dados em dashboards e possibilidade de envio de configurações para o servidor.

5. **Interface local:** neste grupo inclui-se a interface entre o instalador local e o data logger no terreno, nomeadamente configurações da rede WiFi e verificação do correto funcionamento do data logger.

O fluxo de dados ao longo desta estrutura é bidirecional. Por um lado tem-se um fluxo que começa no equipamento e termina no utilizador. Por outro lado, há também um fluxo de dados no sentido contrário para que o instalador e o utilizador final possam submeter configurações ao sistema.

3.2 Arquitetura conceptual

A arquitetura conceptual proposta está representada na figura 3.1. Em cada um dos

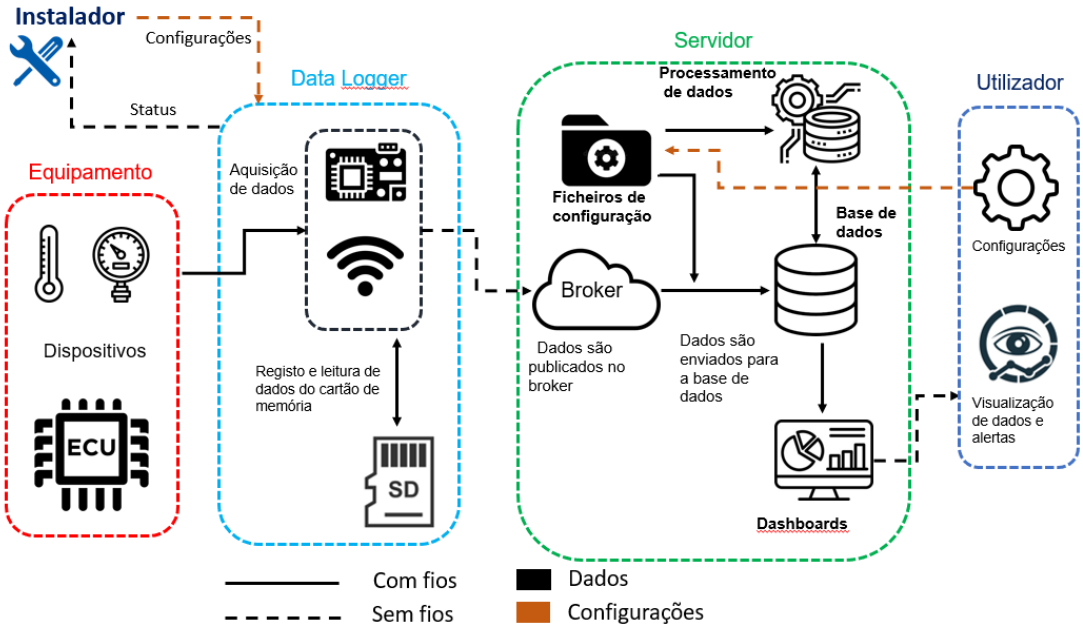

Figura 3.1: Arquitetura conceptual proposta

grupos acima referidos há um conjunto de importantes componentes que desempenham uma determinada função na arquitetura. A definição desta solução teve por base os requisitos definidos para este trabalho e as conclusões retiradas da análise do estado da arte.

Para este trabalho foi definido que deveria ser possível aproveitar, sempre que possível, as potencialidades das ECUs para fazer a recolha de dados. Além disso, também deveria haver um conjunto de sensores externos para recolher dados de temperatura, pressão, corrente e caudal. Desse modo, definiu-se que no primeiro grupo da figura 3.1, que diz respeito ao equipamento, haverá dois blocos: um que diz respeito ao conjunto de sensores externos e outro às ECUs dos equipamentos. Estes componentes serão responsáveis por criar os dados que serão posteriormente tratados como as séries temporais. Para este trabalho será selecionado e construído um hardware que permita a interação entre as fontes de dados e um microcontrolador.

No segundo grupo da figura 3.1 (o data logger), o principal componente será o controlador que terá como função fazer a aquisição dos dados do equipamento e comunicar com as camadas superiores. Definiu-se que deveria incluir uma forma de armazenamento de dados local, pelo que haverá também um cartão de memória. Este data logger deverá ainda permitir uma taxa de aquisição de dados de 1 s e recolher dados de pelo menos dez sensores de temperatura, sensor de pressão hidráulica, caudal, corrente, condições atmosféricas e diretamente da comunicação com a ECU. O data logger deverá manter uma conexão WiFi ao servidor e enviar os dados recolhidos na forma de buffers, através de uma tarefa independente da tarefa de recolha de dados. Para este trabalho é obrigatório que haja o envio de dados para o servidor remotamente, pelo que deverá ser selecionado um protocolo de comunicação adequado para esse efeito. Com base na análise do estado de arte, a seleção será feita entre os protocolos HTTP e MQTT. Neste trabalho,

pretende-se construir um protótipo de um data logger, incluindo o software que permita ao microcontrolador recolher e enviar os dados ao servidor.

O terceiro grupo da figura 3.1 é o servidor alojado na cloud, para o qual foram definidos um conjunto de requisitos, nomeadamente armazenamento e processamento de dados, envio de dados de vários dispositivos para o mesmo servidor, apresentação dos resultados em dashboards e configuração dos diferentes data loggers com vários parâmetros que depois serão utilizados pelo sistema para gerir o fluxo de dados. Nesse sentido, o servidor terá vários componentes:

- **Broker:** será agente intermediário onde são publicados os dados, atuando como a primeira camada do servidor;
- **Base de dados:** terá como objetivo armazenar os dados recolhidos, permitindo gerar históricos que, por sua vez, possibilitam o processamento de dados;
- **Processamento de dados:** terá a função de calcular um conjunto de métricas relevantes e aplicar algoritmos de deteção de anomalias. Deverá aceder aos dados armazenados na base de dados;
- **Dashboards:** terá o objetivo de apresentar os dados relevante de uma forma gráfica e interativa com o utilizador.
- **Ficheiros de configuração:** será um conjunto de ficheiros, (um para cada data logger) que armazenará um conjunto de parâmetros importantes, como por exemplo os sensores que estão em funcionamento, o equipamento do qual está a registar dados, a identidade do ensaio de campo e características do modelo de esquentador em questão.

Neste trabalho pretende-se utilizar um conjunto de ferramentas e desenvolver o software que permita a interação entre elas.

No grupo do utilizador na figura 3.1, haverá duas componentes. A primeira diz respeito à visualização dos dados em dashboards e neste caso pretende-se que haja a possibilidade de visualizar as séries temporais e as métricas calculadas no processamento de forma separada. O frontend deverá permitir que o utilizador interaga com as dashboards permitindo-lhe através de alguns inputs selecionar aquilo que deseja visualizar, nomeadamente escolher parâmetros, data loggers, equipamentos, ensaios de campo e intervalos de tempo. Também deverá permitir visualizar resultados de vários data loggers ao mesmo tempo para que se possam realizar comparações. Um requisito importante que o frontend deve cumprir é a possibilidade de o utilizador fazer o download em formato csv das séries temporais ou métricas que pretenda. Neste trabalho pretende-se utilizar uma ferramenta de visualização para construir o frontend.

Também se pretende que sejam apresentados os outliers identificados pelos algoritmos de deteção de outliers e a emissão de alertas em caso de situações anormais. A segunda componente diz respeito ao envio de ficheiros de configuração para o servidor, os quais serão depois chamados pelos scripts python para efetuar as devidas formatações, em função daquilo que o utilizador pretender.

Por último, o instalador (ver figura 3.1) será o responsável por interagir com o data logger no terreno. A interface com o instalador deverá permitir-lhe acompanhar o estado de funcionamento do data logger, como por exemplo verificar se está a recolher dados

corretamente, se tem ligação à internet, se consegue enviar dados ao servidor, etc. Além disto também deverá permitir que o instalador envie parâmetros de configuração ao data logger, nomeadamente as credenciais da rede WiFi da casa em que for instalado, uma identificação (ID) ao data logger e o equipamento a que ele está conectado. Neste trabalho pretende-se desenvolver o software e hardware necessário para que um instalador possa interagir com o data logger.

Com base na análise do estado da arte concluiu-se que as soluções desenvolvidas por outros autores têm duas limitações que prejudicam o desempenho dos data loggers. Uma dessas limitações é a falta de mecanismos que tornem o data logger à prova de falhas, nomeadamente recuperação de dados que por algum motivo não sejam enviados para o servidor, armazenamento de dados durante períodos de operação em que haja falhas de conexão à internet, rearme automático em caso de falha de energia e recuperação dos dados armazenados antes da falha, auto-restart em caso de longos períodos sem conexão ao servidor e usar leds para apresentar status de operação que permitam identificar anomalias no data logger. Estes mecanismos são fulcrais para melhorar o desempenho do data logger e por isso serão implementados nestes trabalho. A outra limitação diz respeito à utilização de microcontroladores com processadores de dois núcleos. Este tipo de processador, se for bem utilizado, permite tornar tarefas diferentes independentes de forma a que o mau funcionamento de uma não prejudica o funcionamento da outra. Um exemplo disto pode ser ter um núcleo a recolher dados dos sensores e outro a enviá-los ao servidor. Se por algum motivo o envio de dados ao servidor falhar, o outro núcleo poderá continuar a recolher os dados sem problemas, e poderá até armazenar esses dados, para que quando o segundo núcleo recupere a ligação ao servidor, este envie os dados anteriores. Embora não tenha sido referido nas soluções analisadas, o processamento dual-core pode melhorar significativamente o desempenho do data logger. Por esse motivo, é algo que será implementado neste trabalho.

Capítulo 4

Implementação da solução proposta

De forma a cumprir os requisitos para este projeto, a solução proposta no capítulo anterior foi implementada consoante a arquitetura apresentada na figura 4.1. Conforme

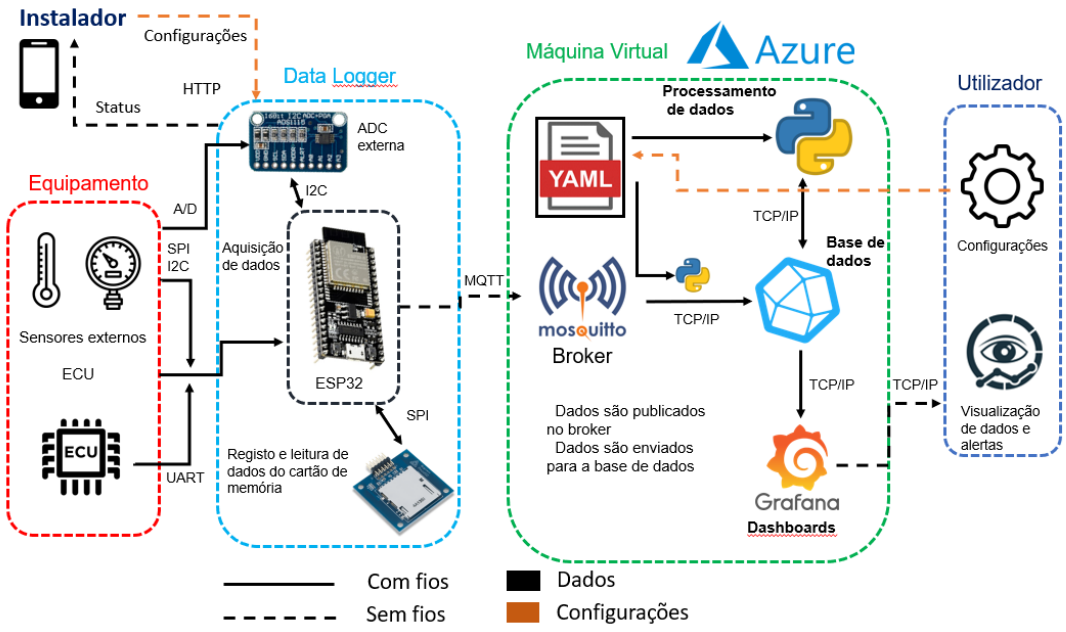

Figura 4.1: Esquema da solução implementada

esquematizado na figura 4.1, a solução implementada inclui a recolha de dados a partir de vários sensores externos e da ECU do equipamento. O hardware selecionado será apresentado na secção 4.1.1.

Além disso foi construído um protótipo do data logger, que será apresentado na secção 4.1.1. Tal como apresentado na figura 4.1, este protótipo inclui um ESP32, placas de conversão analógica externas e uma placa de leitura e escrita num cartão de memória. Foram ainda implementadas comunicações através de protocolos de comunicação c/fios I2C, SPI e UART.

Foi desenvolvido um software (apresentado na secção 4.1.2) para que, em backend, o ESP32 publique os dados num broker Mosquitto, via comunicação sem fios MQTT.

Conforme apresentado na figura 4.1, um script python subscreve e envia os dados para a base de dados InfluxDB, através de comunicação HTTP. Na secção 4.2.4 será apresentado o software desenvolvido para receber os dados dos data loggers e inseri-los na base de dados.

Conforme apresentado na figura 4.1, um bloco desenvolvido em Python interage com a base de dados e com os ficheiros de configuração permitindo o processamento das séries temporais (secção 4.2.5).

Para permitir a visualização de dados na WEB foi utilizado o Grafana, que apresentará ao utilizador um frontend dinâmico e interativo. As dashboards construídas serão apresentadas na secção 4.3.

Estas funcionalidades foram todas instaladas numa máquina virtual Azure para dar a possibilidade de colocar múltiplos data loggers distribuídos remotamente. Além disso, tem como objetivo permitir que um utilizador possa visualizar os dados de qualquer data logger em qualquer lugar com ligação à internet.

Para interagir com os data loggers foi implementada uma interface local (descrita na secção 4.1.2), que permite ao instalador enviar configurações para o ESP32 a partir de páginas WEB no telemóvel. Também a partir do browser poderá acompanhar o estado de operação do data logger para verificar o funcionamento correto. Para esta interface foi construído um hardware constituído por LEDs e um software para que o microcontrolador ESP32 atuasse como servidor WEB.

Do lado do servidor, o utilizador pode enviar ficheiros de configuração para a máquina virtual, que depois são utilizados pelo script de inserção de dados na BD e pelos scripts de processamento para filtrar/processar os dados da forma que o utilizador desejar. Foram desenvolvidos ficheiros em formato YAML, que serão apresentados na secção 4.2.2.

A implementação desta arquitetura foi dividida em três secções fundamentais, as quais são:

- Construção e programação de um protótipo para registo remoto de dados, descrita na secção 4.1;
- Gestão e tratamento dos dados em backend, descrito na secção 4.2;
- Desenvolvimento de um frontend para interagir com um utilizador, apresentado na secção 4.3.

A primeira fase da implementação foi divida em duas partes distintas, mas que trabalham em conjunto. A primeira diz respeito ao hardware, nomeadamente os vários componentes selecionados e o esquema elétrico. A segunda parte diz respeito ao software, em particular o programa fornecido ao ESP para cumprir com os requisitos. No final desta primeira fase havia um protótipo capaz de registar dados de um conjunto de sensores e da ECU do equipamento, manter ligações WiFi, interagir com um cartão de memória e interagir com um instalador local.

A segunda fase da implementação diz respeito ao conjunto de serviços executados em segundo plano, o backend. Neste caso foi implementada uma plataforma para: integrar os vários agentes intervenientes, estabelecer a inserção de dados para a base de dados, a agregação das séries temporais, a deteção de outliers e a interação com ficheiros de configuração dos data loggers.

Por último, a terceira fase permite a um utilizador visualizar os dados desejados através de um frontend dinâmico e interativo, onde será possível, através de um conjunto

de inputs, selecionar variáveis que irão depois ser utilizadas pelo frontend para adaptar os dados apresentados.

4.1 Construção e programação do protótipo de registo remoto de dados

Tal como foi referido, a primeira fase de implementação diz respeito à construção de um hardware de aquisição de dados e de um código para permitir ao ESP32 executar as tarefas desejadas.

4.1.1 Construção do hardware

Nesta secção será apresentado o hardware construído para fazer a aquisição de dados. Primeiro serão apresentados os componentes selecionados, nomeadamente sensores e outros componentes úteis. De seguida será apresentado o desenvolvimento do esquema elétrico que integrará todos os componentes numa só solução e por último o protótipo construído.

Hardware selecionado e forma de ligação

A seleção do hardware depende muito da escolha do microcontrolador, visto que a escolha dos restantes componentes será condicionada por este, tanto ao nível da comunicação e como da alimentação. Os sensores e outros dispositivos foram selecionados com base em vários critérios, nomeadamente:

- Tensão de alimentação e consumo de corrente - este parâmetro é muito importante para garantir que há forma de alimentar o componente a partir do microcontrolador sem ser necessário implementar hardware mais complexo;
- Interface de comunicação - este parâmetro é importante para estabelecer o esquema elétrico e garantir que o microcontrolador consegue interagir com todos os componentes;
- Acessibilidade de aquisição - visto que este trabalho foi desenvolvido para uma empresa, um dos critérios considerados foi a facilidade com que a empresa pode adquirir os componentes;
- Gamas de medição - foi importante garantir que os componentes selecionados poderiam ser utilizados nas condições de temperatura e presssão desta aplicação.
- Calibração - optar por sensores que já incluam a calibração simplifica significativamente a implementação.

Com base nestes critérios foram selecionados os vários componentes, que serão apresentados de seguida.

ESP32

O principal componente desta solução é o microcontrolador ESP32. Este apresenta as características indicadas para este trabalho, nomeadamente: processador dual-core com alta capacidade de processamento, número elevado de pinos GPIO, várias interfaces de comunicação (UART, SPI, I2C) e módulo WiFi. O processamento dual-core é uma característica bastante relevante, pois permite ao ESP32 definir um núcleo para desempenhar um determinado tipo de tarefas como, por exemplo, recolha de dados, e outro núcleo para desempenhar as tarefas relativas ao envio de dados para o servidor. Desta forma, é possível tornar as duas tarefas independentes e se ocorrer, por exemplo, uma falha de WiFi, a recolha de dados não será afetada. Este componente também permite ligar uma quantidade elevada de periféricos, visto que tem muitos pinos GPIO e várias interfaces de comunicação. A placa utilizada para o desenvolvimento deste trabalho foi a ESP32 Dev KitC V4. Esta placa pode ser alimentada por uma fonte de 5V e depois poderá alimentar outros componentes com 3.3V. É de salientar que, dos vários GPIO disponíveis na placa, nem todos podem ser utilizados. Os pinos 6-11 não poderão ser utilizados para qualquer função, pois estão conectados à memória flash. Os pinos 34-36 e 39 só poderão ser utilizados como entradas e poderá haver falhas no arranque do microcontrolador caso os pinos 3 e 12 sejam utilizados como entradas. Outro aspeto a considerar é que, caso o ESP32 esteja ligado a uma rede WiFi, não poderá utilizar todo o canal ADC2 (10 pinos) para realizar conversões analógicas digitais.

Sensores NTC

Para recolher dados de temperatura foram utilizados termistores NTC. Estes componentes têm uma resistência elétrica que diminui com o aumento de temperatura. Assim, através de um divisor resistivo e de uma conversão A/D é possível obter a resistência do sensor. Os sensores utilizados têm um intervalo de medição entre -40 e 125 ºC, no entanto para esta aplicação foi apenas considerado o intervalo 5-80 ºC. Com a datasheet do sensor foi traçada a curva de calibração para converter a resistência elétrica em temperatura. Para fazer a leitura da temperatura foi utilizado um divisor resistivo alimentado a 3.3V que incluiu uma resistência de 10 $k\Omega$ e o termístor NTC. Em função da temperatura, a resistência do sensor será diferente e por isso a tensão aos terminais do sensor também será diferente. Através da conversão A/D da tensão e de um conjunto de cálculos foi possível obter a temperatura. Os termístores NTC utilizados têm uma resistência de cerca de 10 $k\Omega$ à temperatura de 25ºC. Por esse motivo, as resistências elétricas utilizadas também foram de 10 $k\Omega$. Deste modo, não haverá uma discrepância signifcativa entre a diferença de potencial no sensor e a diferença de potencial na resistência que possa comprometer a qualidade das medições.

Placas ADS1115

Estas placas[1] permitem realizar a conversão analógica digital externamente e depois enviar os valores digitais por I2C para o microcontrolador. Assim, estas placas são responsáveis por obter os valores digitais que depois o ESP32 solicita. Visto que a conversão analógica do ESP32 tem algumas limitações (tal como se concluiu na análise do estado da arte) e estas placas permitem realizar conversões A/D com um output de 16 bits (implica uma resolução de 0.1875 mV), a utilização das mesmas permite obter

[1] https://www.best-microcontroller-projects.com/ads1115.html

melhores resultados. Como cada uma destas placas permite ligar quatro sensores NTC, foram utilizadas três placas, possibilitanto assim ligar doze NTCs. Visto que comunicam com o ESP32 por I2C foi necessário ter endereços diferentes para cada uma. De acordo com a datasheet do componente, conectando o pino ADDR a um dos seguintes pinos: VCC, GND, SCL e SDA é possível obter quatro endereços I2C diferentes. Assim, a montagem elétrica consistiu em ligar três destas placas ao ESP32, através dos condutores SCL e SDA e variar o endereço através do pino ADDR. Estas placas não só permitem obter conversões A/D mais estáveis e precisas como também permitem ligar mais sensores NTC, pois o ESP32 em modo WiFi só pode utilizar seis entradas analógicas. A montagem elétrica para recolha de dados de temperatura utilizandos os termístores NTC, as placas ADS1115 e o ESP32 está representada na figura 4.2. Conforme representado na figura

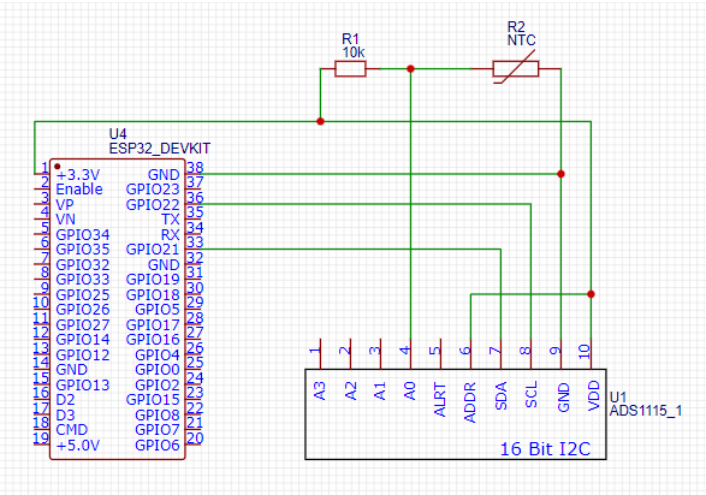

Figura 4.2: Esquema de ligações para aquisição de temperatura de um sensor NTC utilizando uma placa de conversão A/D externa

4.2, o ESP32 comunica através dos pinos 22 (SCL) e 21 (SDA) com a placa ADS1115. Esta é também alimentada a 3.3V pelo ESP32, tal como o divisor resistivo onde foi colocado o sensor NTC. Uma das entradas analógicas da placa é utilizada para medir a tensão nos terminais do sensor. Para configurar o endereço I2C da placa, o pino ADDR pode ser ligado ao VCC, SDA ou SCL, ficando com endereços 0x49, 0x4A, or 0x4B, respetivamente. Caso o pino ADDR não seja ligado a nenhum destes o endereço é 0x48, por defeito. Na figura anterior foi representado apenas o esquema para ligação de um sensor, mas no esquema elétrico final serão apresentados todos.

Caudalímetro

Para medir o caudal de água percorrido no esquentador foi selecionado um caudalímetro ELK[2] tipicamente usado pela empresa. O esquema elétrico associado a este sensor inclui um VCC, um GND e um output, que corresponde a uma onda quadrada com frequência proporcional ao caudal. A ligação entre o ESP32 e este sensor consistiu em ligar o VCC aos 3.3V, os GND e a saída do sensor V_{out}, ao GPIO 4 do ESP32. Foi selecionado este pino pois pode ser utilizado como input sem quaisquer conflitos.

[2]https://www.saginomiya.co.jp/en/auto/searchresult11.php?FilterID1=3&FilterID2=2

Sensor de pressão hidráulica

O sensor de pressão hidráulica selecionado foi um sensor da série NPI-19[3] desenvolvido pelo fabricante Nova Sensor. Um dos requisitos para este trabalho é a existência de um sensor que possa medir a pressão de circulação do circuito hidráulico e que esteja preparado para medir pressões até 16 bar, garantindo que o fenómeno de Golpe de Aríete[4] não danifique o sensor. O sensor selecionado possui vários modelos com diferentes gamas de pressão máxima, sendo que o modelo selecionado foi o NPI-19J-300A2 que pode medir pressões até 300 psi, ou seja 20.7 bar. Este sensor possui uma interface I2C, que pode ser utilizada para adquirir a pressão hidráulica e a temperatura ambiente. O esquema elétrico de montagem deste sensor está representado na figura 4.3. Neste caso foi utilizado

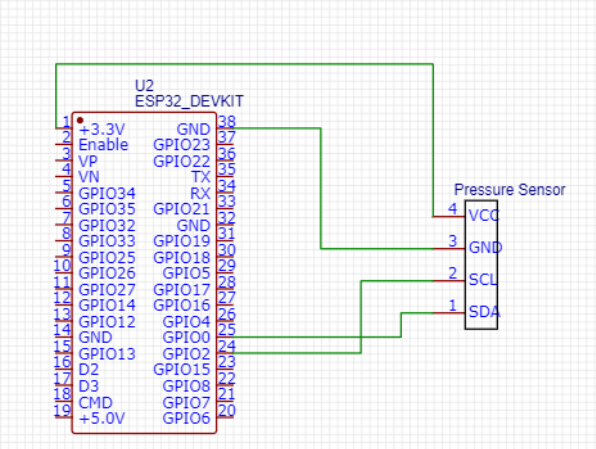

Figura 4.3: Esquema de ligação entre o ESP32 e o sensor de pressão hidráulica

um novo barramento I2C, ao invés do mesmo barramento das placas ADS1115. Foram realizados testes com as placas e o sensor no mesmo barramento, e concluiu-se que a ligação do pino ADDR das placas aos pinos SDA ou SCL iria causar conflitos com outros componentes ligados no mesmo barramento, ou seja para usar mais do que duas placas ADS1115 foi necessário implementar um novo barramento para este sensor. Este sensor pode ser alimentado a 3.3V.

Sensor de corrente

O sensor de corrente selecionado foi o sensor ACS712, desenvolvido pela Allegro microsystems, neste caso uma placa de circuito[5] que inclui o sensor. Este sensor necessita de ser alimentado a 5V, pelo que foi necessário utilizar a fonte de 5V (que alimentará o ESP32) para alimentar também este componente. O output deste sensor é um sinal analógico proporcional à corrente que passa pelo sensor. Se a corrente for nula, a tensão de saída é 2.5V, se a corrente for positiva, a tensão varia entre 2.5 e 5V e se a corrente for negativa, a tensão varia entre 2.5V e 0. Posto isto, como no ESP32 só podem ser utilizados como input sinais de tensão até 3.3V, foi necessário usar um divisor resistivo para diminuir o nível de tensão. Para passar de 5V para 3.3V foi necessário implementar

[3]https://www.amphenol-sensors.com/en/novasensor/pressure-sensors/3160-npi-19-low

[4]Variações rápidas de pressão causadas por perturbações na circulação de líquidos

[5]https://www.seeedstudio.com/blog/2020/02/15/acs712-current-sensor-features-how-it-works-arduino-guide/

um divisor resistivo com uma razão de cerca de 2/3, utilizando a equação 4.1.

$$V_{saida} = \frac{R_2}{R_1 + R_2} V_{entrada} \tag{4.1}$$

Foram selecionadas duas resistências, uma de 2.2 $k\Omega$ (R_2) e outra de 1.2 $k\Omega$ (R_1). O esquema elétrico implementado está representado na figura 4.4. Desta forma consegue-

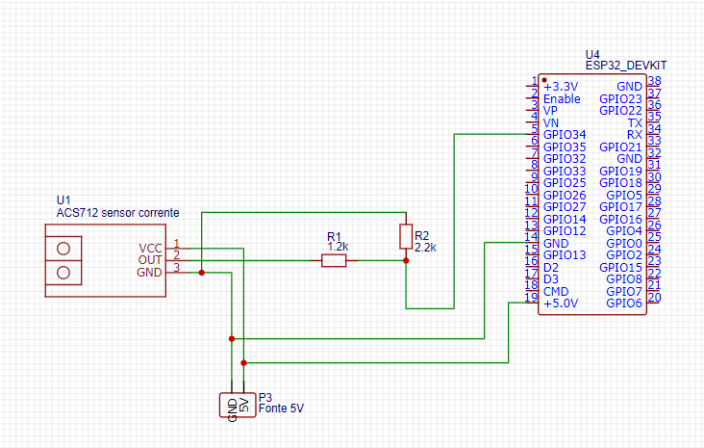

Figura 4.4: Esquema elétrico de montagem do sensor de corrente

se reduzir a tensão, de modo que se a tensão de saída do sensor for de 5V a tensão do sinal analógico lido pelo ESP32 será de 3.33V. O modelo deste sensor selecionado foi o ACS712ELCTR-30A-T, que pode medir corrente até 30A e tem uma sensibilidade de 66 mV/A. Um dos requisitos para o trabalho é que o sensor de corrente consiga medir corrente na gama 12-16 A. Assim, seria mais adequada a utilização da versão ACS712ELCTR-20A-T, que poderia medir até 20 A com maior precisão. Todavia, o modelo a que se teve acesso foi o de 30A.

Sensor BME280

Um dos requisitos para este trabalho foi incluir um sensor para permitir monitorizar as condições atmosféricas, nomeadamente a temperatura, pressão e humidade do ar ambiente. Nesse sentido selecionou-se o sensor BME280, neste caso uma placa de desenvolvimento com esse sensor[6], que permite medir todos esses parâmetros, simplificando assim o hardware. Este sensor possui interfaces de comunicação I2C e SPI e pode ser alimentado pelos 3.3V do ESP32. No caso da humidade, tem um erro de medição inferior a 3%, e no caso da pressão atmosférica inferior a 0.25 %. Como já tinha sido implementado um novo barramento I2C para o sensor de pressão hidráulica, neste sensor também foi utilizada a interface I2C, cujo esquema de montagem está representado na figura 4.5.

Sensor MAX6675

Apesar de não ser um requisito para este trabalho decidiu-se também utilizar um sensor de temperatura MAX6675 [7], com o objetivo de atuar como referência de calibração

[6]https://wiki.seeedstudio.com/Grove-Barometer_Sensor-BME280/

[7]https://www.botnroll.com/pt/temperatura/3883-m-dulo-sensor-temperatura-0-800-tipo-k-max6675.html

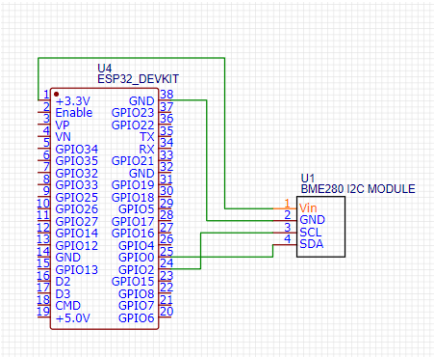

Figura 4.5: Esquema de ligação entre o sensor BME280 e o ESP32

para os sensores NTC. Este sensor consiste num termopar ligado a uma placa de transdução, com a qual se pode comunicar por uma interface SPI. Desta forma é possível enviar pedidos para recolher a temperatura medida pelo sensor. Para implementar este componente foi necessário criar um barramento SPI, cujo esquema elétrico está representado na figura 4.6.

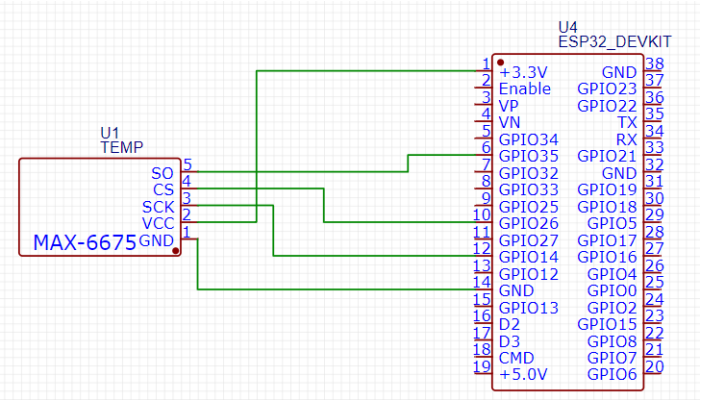

Figura 4.6: Esquema elétrico de ligação entre o sensor MAX6675 e o ESP32

Placa de leitura e escrita num cartão de memória

Para permitir realizar o armazenamento de dados localmente foi utilizada uma placa Diligent Pmod SD[8], que permite inserir um cartão de memória e interagir pelo protocolo de comunicação SPI. A montagem elétrica consistiu em estabelecer um barramento SPI, no qual se ligaram os pinos SCK, MISO, MOSI e CS da placa aos GPIO 18, 19, 23 e 5 do ESP32. Esta placa pode ser alimentada a 3.3V. Foi utilizado um barramento SPI diferente daquele utilizado para o sensor MAX6675, pois os testes realizados revelaram que a interação com esta placa e o sensor num mesmo barramento SPI iria provocar conflitos que impediriam o funcionamento desejado. Quando o ESP32 tentasse ler o cartão SD para enviar os dados ao servidor, como necessitaria de vários segundos, o barramento ficaria ocupado e não poderia ser utilizado para recolher dados do sensor.

[8]https://digilent.com/reference/pmod/pmodsd/start

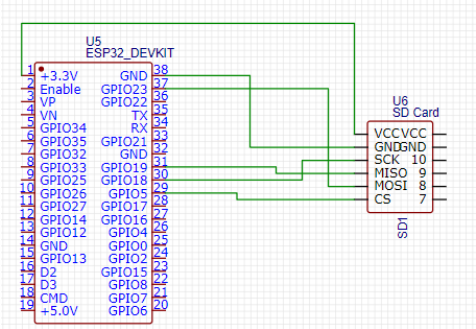

Figura 4.7: Esquema de ligação entre o ESP32 e a placa PCB para leitura e escrita num cartão SD

Por esse motivo, foi necessário que esta placa ficasse num barramento sem nenhum outro dispositivo.

Interface UART

Para permitir a comunicação com as unidades de controlo eletrónicos dos equipamentos foi também implementada uma interface UART. Isto foi implementado através da ligação entre pinos TX, RX e GND conforme o esquema da figura 4.8. Para isso foram

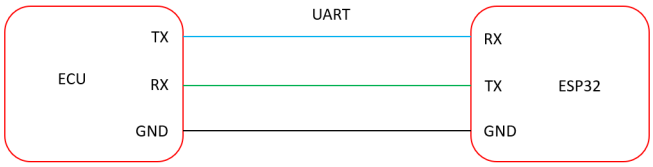

Figura 4.8: Esquema de ligações UART entre a ECU e o ESP32

utilizados os pinos 16 e 17 do ESP32 para atuar como RX e TX, respetivamente.

LEDs

Para permitir verificar o estado de funcionamento do ESP32 localmente sem ser necessário utilizar um computador foram também utilizados LEDs ligados a saídas digitais do microcontrolador. A forma de ligação destes LEDs está representada na figura 4.9. As características dos LEDs selecionados são 2V e 20 mA. Posto isto, utilizou-se a equação 4.2 para calcular a resistência do elemento resistivo a utilizar.

$$R = \frac{V_{out} - V_{led}}{I_{led}} \tag{4.2}$$

Pela equação 4.2 resistência selecionada deveria ter cerca de 65 Ω.

Esquema elétrico

A integração de todos os componentes anteriores e respetivos esquemas de montagem numa solução culminou no esquema elétrico final representado na figura 4.10. Este cir-

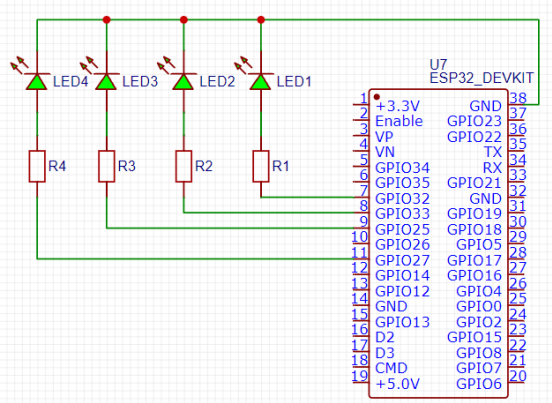

Figura 4.9: Esquema de montagem dos LEDs

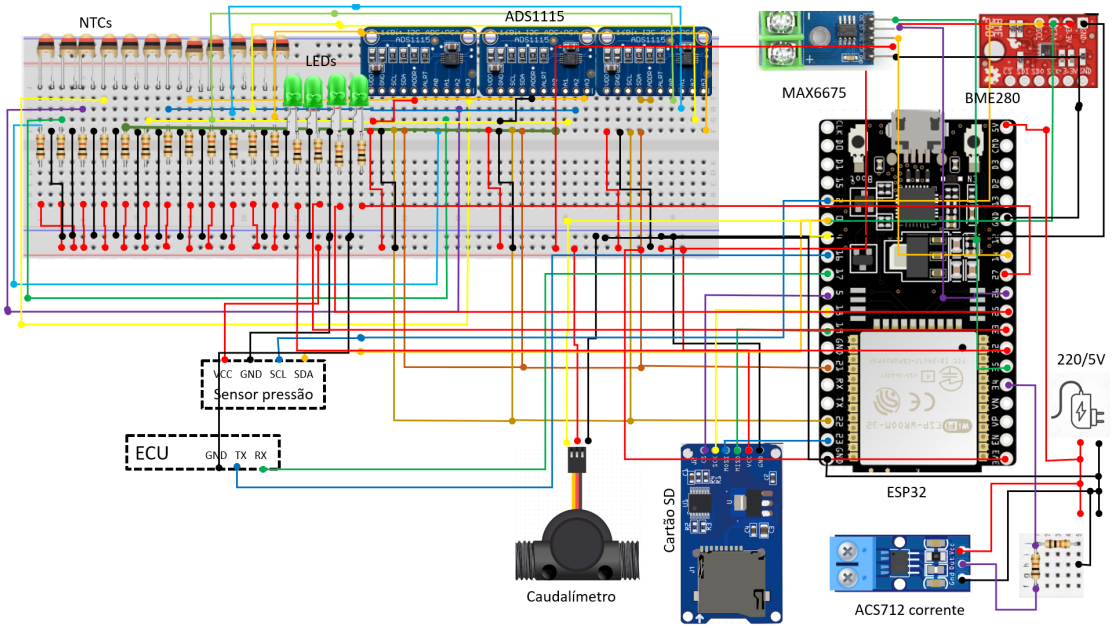

Figura 4.10: Esquema elétrico da solução implementada

cuito é alimentado por uma fonte de 5V, que será responsável por fornecer energia diretamente ao ESP32 e ao sensor de corrente. Como o ESP32 tem incorporado um regulador de tensão de 3.3V, os restantes perfiféricos serão alimentados diretamente pelo microcontrolador. Este esquema elétrico resulta de um processo iterativo, onde em cada iteração foram adicionados periféricos e testado o seu funcionamento. Ao longo do processo foram encontrados alguns problemas que levaram a alterações no esquema de montagem, nomeadamente a impossibilidade de utilizar as placas ADS1115 no mesmo barramento I2C de outros periféricos, se pelo menos uma das placas tivesse o pino ADDR ligado ao Serial Clock ou ao Serial Data. Outro problema encontrado foi que se a placa do cartão de memória fosse ligada por SPI no mesmo barramento do sensor MAX6675 não haveria um bom funcionamento de ambos. Inicialmente, uma das razões pelas quais foi escolhido o ESP32 era o número de pinos de conversão A/D, o que daria a possibilidade de ligar

vários sensores de temperatura NTC. No entanto, conforme será apresentado no capítulo 5, a conversão analógica do ESP32 revelou ter alguns problemas, nomeadamente falta de precisão e exatidão. Por esse motivo, para atingir maior consistência e exatidão nos dados recolhidos decidiu-se utilizar apenas as placas ADS1115 para fazer a aquisição de temperatura a partir dos sensores NTC. Para simplificar a leitura dos sensores NTC, apenas se teve em conta um único valor de resistência elétrica para as resistências dos divisores resistivos. Contudo, é importante ter em conta que as resistências comuns têm uma incerteza de 5%.

Condicionamento de sinal

Depois de definido o esquema elétrico para integrar todos os componentes foram também implementados condensadores eletrolíticos entre o VCC e o GND de cada componente, para condicionar os sinais de alimentação. Colocados desta forma, os condensadores atuam como um filtro passa-baixo, forçando os sinais de ruído eletromagnético AC, que têm altas frequências, a ser encaminhados para o GND, enquanto que o sinal DC será encaminhado para o componente desejado. Desta forma, retira-se o ruído dos sinais DC, tornando-os mais regulares. Estes condensadores têm ainda uma segunda vantagem, pois atuam como fontes de alimentação DC quando os componentes exigem picos de corrente. Para este propósito foram selecionados condensadores de 10 μF. Na figura 4.11 está representado o esquema elétrico para o sensor de pressão com um condensador entre o VCC e o GND a condicionar o sinal DC de alimentação.

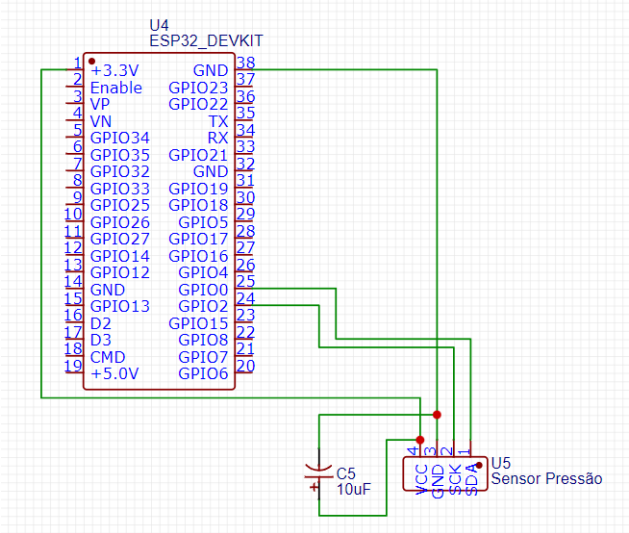

Figura 4.11: Esquema de ligação elétrica entre o sensor de pressão e o ESP32 com o condicionamento do sinal

Para os restantes componentes foi implementado o mesmo condicionamento que o que está representado na figura anterior.

Protótipo

Para se obter uma solução mais robusta e versátil foi desenhada e encomendada uma placa PCB para integrar os vários componentes eletrónicos de forma ergonómica. Para esse

efeito recorreu-se ao software EasyEDA. Esta placa inclui espaço para acoplar todos os sensores, fichas de comunicação UART, a placa de comunicação com cartão SD, divisores resistivos, condensadores, o ESP32 e as placas ADC. O resultado obtido, com dimensões de cerca de 150x140 mm encontra-se representada na figura 4.12.

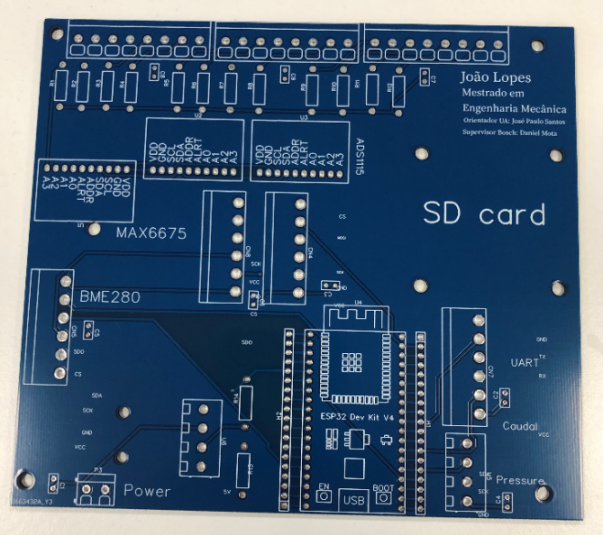

Figura 4.12: Placa PCB construída para o data logger

Conforme é possível verificar na figura anterior deixaram-se os furos necessários para soldar os vários componentes. Na PCB desenhada não foi desde logo prevista a implementação dos LEDs, pelo que essa componente foi construída à parte. Os vários sensores foram ligados a conectores de parafuso e foram utilizadas PCB sockets para colocar o ESP e as placas ADS1115. Foram ainda utilizadas resistências elétricas e condensadores THT (through hole technology). O resultado final desta implementação encontra-se representado na figura 4.13. Conforme pode ser visto na figura 4.13 o conceito implementado permite ligar até doze sensores de temperatura NTC, um caudalímetro, um sensor de pressão hidráulica, um sensor de corrente, um sensor max6675, um sensor bme280 e um cartão de memória. Possui também uma ficha para ligar à ECU de um esquentador e um conjunto de LEDs para fornecer informação acerca do estado de funcionamento do data logger.

4.1.2 Software implementado

Nesta secção será apresentado o software implementado no ESP32 para permitir ao data logger cumprir os requisitos deste trabalho.

Descrição geral

Para programar o microcontrolador ESP32 foi utilizado o Arduino IDE, que é uma ferramenta desenvolvida para microcontroladores Arduino, mas que com as configurações necessárias também pode ser utilizada para programar ESPs.

O software implementado no ESP32 divide-se em duas fases principais. Primeiramente, uma fase que é executada no arranque do ESP e que se traduz pelo modo de

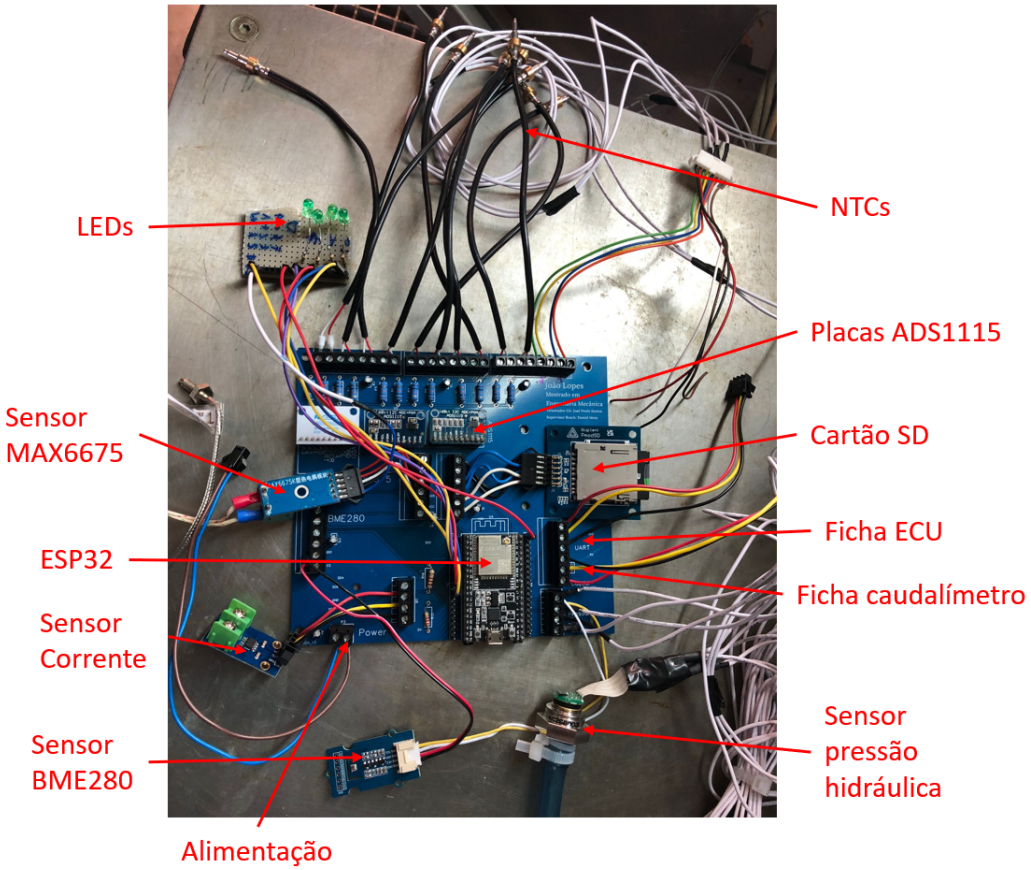

Figura 4.13: Protótipo final depois de instalados os vários componentes

configuração. A fase seguinte, executada caso a configuração tenha sucesso, será responsável por recolher dados e enviá-los para o servidor. Esta segunda fase, vista como o funcionamento normal, utiliza a programação multi-core do microcontrolador para alcançar uma performance à prova de falhas. Neste caso, a tarefa de recolha de dados é independente da tarefa de envio ao servidor. Assim, as duas tarefas correm paralelamente sem interferência e o cartão de memória atua como agente intermediário entre ambas. Estas tarefas funcionam de forma independente, mas cada uma delas atua de forma síncrona. Além destas, durante o modo de funcionamento normal foi implementada uma terceira tarefa, em modo assíncrono, que permite efetuar novas configurações sem ser necessário ativar o modo de configuração. Apresentada a visão geral do programa implementado, segue-se a descrição mais detalhada da sequência de tarefas executadas, começando pelo modo de configuração:

1. Ativa as interfaces de comunicação, UART, I2C e SPI com os vários componentes.

2. Define um conjunto de parâmetros, como credentiais de conexão ao broker, limite de tamanho de mensagem MQTT e parâmetros de configuração da obtenção de hora absoluta a partir de um servidor NTP (network time protocol).

3. Lê os parâmetros de configuração armazenados na memória flash[9].
4. Tenta estabelecer uma ligação WiFi com o router. Caso a tentativa não tenha sucesso, ativa o modo Ponto de Acesso WiFi. Caso tenha sucesso, salta para a etapa 8.
5. Em modo de Ponto de Acesso aguarda que algum dispositivo tente estabelecer uma ligação WiFi. Caso não aconteça o ESP reinicia (volta à etapa 1).
6. Se algum dispositivo se conseguir conectar, aguarda por um pedido HTTP do tipo GET (método utilizado para pedir dados a um servidor). Quando esse pedido chegar, responde com uma página WEB.
7. Aguarda por um pedido HTTP do tipo POST (método utilizado para trocar dados com um servidor) com um conjunto de parâmetros de configuração. Quando recebe esse pedido, armazena os novos parâmetros na memória flash e reinicia (volta à etapa 1).
8. Utiliza o protocolo Multicast DNS (mDNS) para permitir que seja atribuido um hostname ao endereço IP do ESP.
9. Ativa uma tarefa em modo assíncrono para atuar como servidor e estar à escuta de pedidos HTTP.
10. Cria uma tarefa que será executada pelo núcleo 0.
11. Ativa o modo de funcionamento normal com duas tarefas paralelas a serem executadas por núcleos diferentes.

Ativado o modo de funcionamento normal haverá duas tarefas a serem executadas paralelamente. Segue-se a sequência de tarefas que serão executadas por cada uma delas:

- Núcleo 0 (recolha de dados):
    1. Ativa uma rotina de interrupção para contar os pulsos do caudalímetro.
    2. Ativa um compasso de espera até ter decorrido 1 s desde a última recolha de dados.
    3. Verifica a ligação WiFi ao router. Se não estiver conectado salta para a etapa 5.
    4. Estabelece uma ligação com um servidor NTP e atualiza a data e hora atual. Salta para a etapa 6.
    5. Desliga o LED que sinaliza o estado da ligação WiFi. Atualiza manualmente a data e hora absoluta, incrementando o tempo que passou desde a recolha de dados anterior.
    6. Calcula o caudal a partir do número de pulsos contabilizados desde a última medição. Faz reset aos pulsos.
    7. Verifica a ligação ao sensor max6675. Se estiver "OK", lê a temperatura do sensor max6675, se não estiver "OK", envia uma mensagem de erro e atualiza o valor para um código de erro.

[9]Memória não volátil que em caso de falha de energia as informações não são perdidas

8. Tenta estabelecer uma comunicação com o sensor BME280. Se conseguir faz a leitura da temperatura ambiente, pressão atmosférica e humidade relativa. Se não conseguir envia uma mensagem de erro e atualiza os outputs para códigos de erro.
9. Faz a leitura do sensor de corrente. Caso detete que o sensor não está bem ligado, envia mensagem de erro e atualiza a variável "corrente" para um código de erro.
10. Estabelece uma interação com o sensor de pressão hidráulica e faz a leitura da pressão hidráulica. Caso não receba nenhuma resposta envia uma mensagem de erro e atualiza a variável pressão hidráulica para um código de erro.
11. Verifica a ligação com as várias placas ADS1115. Para cada uma delas, se conseguir estabelecer a interação pede os valores das conversões analógicas digitais em cada uma das entradas da placa e calcula a respetiva temperatura do sensor NTC. Caso não consiga comunicar com a placa, envia uma mensagem de erro e atribuí um código de erro às variáveis que dizem respeito a essa placa.
12. Inicia o processamento das mensagens recebidas por UART das ECUs. Caso haja bytes guardados no buffer procederá à sua leitura. Se não houver salta para a etapa 18.
13. Lê um byte armazenado no buffer. Caso encontre os bytes de início de mensagem começa a construir a mensagem a partir dos bytes que lê. Caso contrário ignora e volta para a etapa 12.
14. Continua a ler bytes armazenados no buffer e a guardá-los num array. Se encontrar os bytes de final de mensagem avança para a etapa seguinte, caso contrário continua nesta etapa até atingir o limite máximo definido. Se atingir o limite máximo definido salta para a etapa 18 e envia mensagem de erro.
15. Quando encontra o final da mensagem, inicia o processamento da mensagem que terminou de ler. Verifica qual o equipamento ao qual está ligado para determinar como deve processar a mensagem e verifica o byte que indica o tipo de mensagem. Se esse byte indicar que se trata da mensagem esperada avança para a etapa seguinte, caso contrário envia mensagem a indicar que não é uma mensagem conhecida, ignora e salta para a etapa 18.
16. Em função do equipamento ao qual está ligado, retira um conjunto de parâmetros e guarda-os num ficheiro de texto.
17. Guarda toda a mensagem num ficheiro de texto.
18. Guarda as variáveis com as leituras dos sensores num ficheiro de texto criado para os sensores.
19. Envia mensagens de status por porta série e através dos LEDs.
20. Volta à etapa 2.

- Núcleo 1:

1. Tenta estabelecer uma ligação WiFi com o router até conseguir.
2. Tenta estabelecer uma ligação com um servidor NTP. Se não conseguir envia uma mensagem de erro. Se conseguir pede a data e hora absoluta.

3. Tenta estabelecer uma ligação com o broker até conseguir.
4. Verifica quanto tempo passou desde a última vez que enviou dados ao broker. Se passaram menos de 10 s, volta à etapa 1 do núcleo 1, caso contrário inicia o envio de dados ao servidor.
5. Verifica qual foi o último ficheiro de texto com dados dos sensores externos que leu. Se leu o "sensores0.txt" abre o "sensores1.txt", se leu o "sensores1.txt" abre o "sensores0.txt". Se não conseguir encontrar o ficheiro, envia uma mensagem de erro e salta para a etapa 11.
6. Lê os dados guardados no ficheiro até encontrar uma mudança de linha.
7. Separa os dados por vírgulas e cria um objeto JSON.
8. Publica o JSON no broker. Se falhar o envio, pisca os LEDs para sinalizar um erro de envio e restabelecer a ligação WiFi ao router e a ligação MQTT ao broker. Depois volta a publicar o JSON.
9. Verifica se ainda há linhas de dados para ler. Se houver volta à etapa 6 .
10. Sinaliza através dos LEDs que terminou de ler um ficheiro e apaga-o.
11. Ativa um tempo de espera. Se tiver lido "mensagens UART" salta para a etapa 13, se tiver lido "fp2", "ewi" ou "kme", salta para a etapa 14.
12. Volta à etapa 5, mas substitui "sensores" por "mensagens UART".
13. Verifica a variável "equipamento" na memória flash. Se for "fp2", "ewi" ou "kme", volta à etapa 5 mas substitui sensores por "fp2", "ewi" ou "kme", respetivamente.
14. Atualiza a variável que armazena a hora do último envio e volta à etapa 1.

Além do modo de funcionamento normal também é criada uma tarefa que corre em modo assíncrono. As funções desempenhadas por esta tarefa são as seguintes:

1. Fica à escuta de pedidos HTTP. Quando recebe um pedido passa à etapa seguinte.
2. Envia uma resposta HTTP com uma página WEB e fica em compasso de espera. Quando recebe um novo pedido avança.
3. Verifica que tipo de pedido recebeu. Se for um pedido "GET", a solicitar a página WebSerial avança para a etapa seguinte. Se for "POST" salta para a etapa 5.
4. Responde com uma página WEB com mensagens de debug e volta à etapa inicial desta tarefa.
5. Armazena os novos parâmetros de configuração na memória flash e reinicia o ESP.

Nas figuras que se seguem estão representados os diagramas UML das sequências de funções das tarefas referidas. Devido à grande quantidade de funções os diagramas foram separados de forma a facilitar a interpreteção. O primeiro diagrama (ver figura 4.14) diz respeito ao modo de configuração que depois iniciar as restantes tarefas. O segundo diagrama (ver figura 4.15) diz respeito à tarefa desempenha pelo núcleo 0 e o último diagrama (ver figura 4.16) diz respeito à tarefa desempenhada pelo núcleo 1.

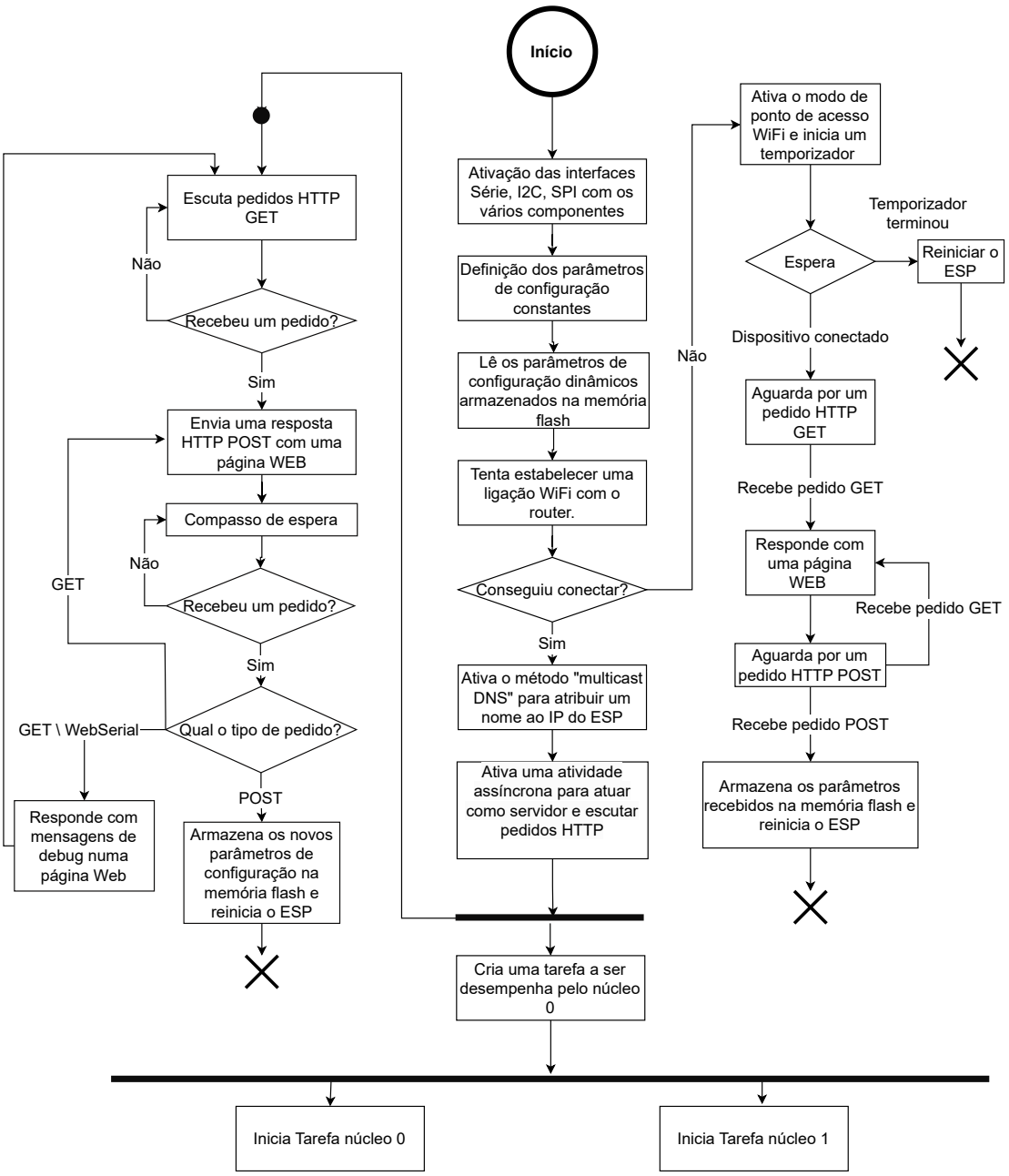

Figura 4.14: Fluxograma da sequência de funções desempenhadas pelo ESP32 durante o modo de configuração e arranque

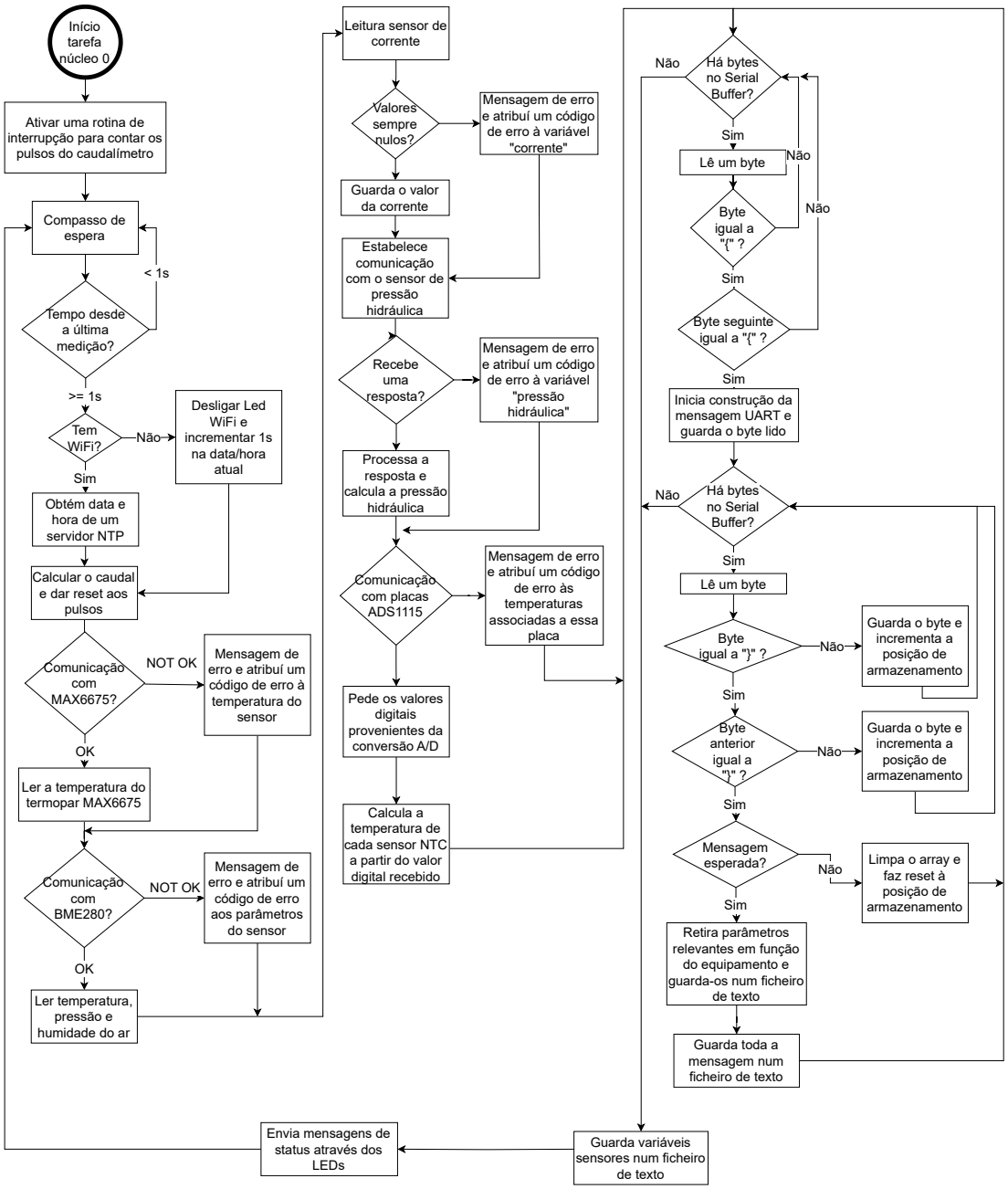

Figura 4.15: Fluxograma da sequência de funções desempenhadas pelo ESP32 durante a execução da tarefa do núcleo 0

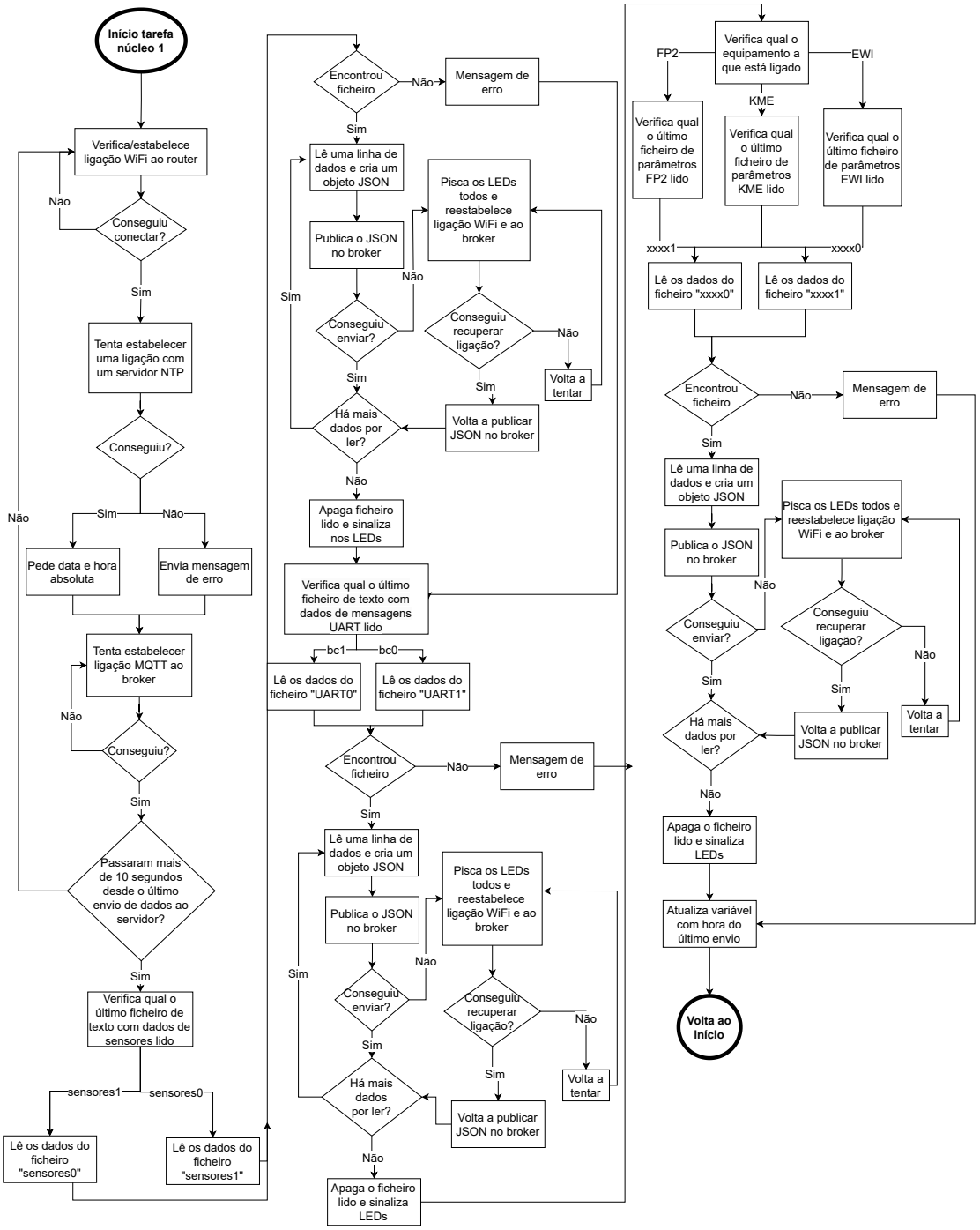

Figura 4.16: Fluxograma da sequência de funções desempenhadas pelo ESP32 durante a execução da tarefa do núcleo 1

Implementação do modo de configuração

Para que o data logger funcione da forma desejada é necessário fornecer-lhe alguns parâmetros de configuração, nomeadamente as credenciais para ligação à rede WiFi, a identidade do data logger e o equipamento em que ele foi instalado. No caso das configurações da rede WiFi, estas terão de ser feitas localmente no momento da instalação, pelo que se excluíram quaisquer formas de envio remoto. De forma local existem algumas opções, entre elas:

- Configurações hard coded: esta opção consiste em definir os parâmetros de configuração diretamente no código do ESP. Esta opção foi excluída pois tornaria a solução pouco adaptável, obrigando a habilidades de programação e equipamento específico no momento da instalação.

- Registo dos parâmetros em cartão de memória: esta opção coloca os dados de configuração num ficheiro ".txt" no cartão SD e quando o ESP arranca lê estas configurações. Esta opção obriga a aceder ao cartão de memória dentro da caixa do data logger e à sua configuração através de um computador. Além disso, por questões de segurança de dados, colocar passwords de redes WiFi em cartões de memória é uma opção pouco segura.

- Implementação de uma HMI no data logger: esta opção permite interagir com o ESP fornecendo localmente as credenciais através de uma interface que pode incluir um ecrã touch, ou um conjunto de botões. Esta opção é viável do ponto de vista técnico, porém é bastante dispendioso aplicar uma interface como esta em todos os data loggers.

- Configurações Wireless através de páginas WEB: esta opção passa por ligar o ESP em modo de Ponto de Acesso e de seguida o instalador pode conectar o telemóvel à rede do ESP. Depois disso, o instalador pode aceder ao browser, colocar o IP do ESP e surge um formulário com um conjunto de campos para preencher. Esta opção foi a selecionada, pois trará apenas a necessidade de o instalador utilizar o próprio telemóvel, não haveria problemas de segurança de dados e facilmente se instala o data logger em qualquer lugar.

Conforme explicado anteriormente, no momento que o ESP32 é ligado, passa por um modo de configuração em que mesmo que seja a primeira vez que o data logger é ligado num novo local será possível configurá-lo sem necessidade de reprogramar o ESP. A rotina de configuração começa por ler os parâmetros guardados na memória flash do microcontrolador. Naturalmente, na primeira vez que o data logger for ligado, não terá nenhum parametro guardado. Posteriormente, quando já estiver em funcionamento e for trocado de equipamento ou de lugar, já existirão parâmetros armazenados na memória flash. A utilização da memória flash permite ao ESP guardar parâmetros de configuração mesmo que seja desligado. Desta forma, caso haja uma falha de energia ou seja desligado da alimentação, quando voltar a ligar não é necessário fornecer os parâmetros de configuração novamente. Depois de ler os parâmetros armazenados na memória flash, que são as credenciais para ligação à rede WiFi, a ID do data logger e o equipamento em que foi aplicado, tenta ligar-se à rede WiFi utilizando essas credenciais. Se o datalogger não tiver sido anteriormente ligado à rede WiFi em que está colocado, não conseguirá

conectar-se pois as credenciais estarão erradas. Nesse caso, se ao fim de 30 segundos o microcontrolador não tiver sucesso na ligação ao router, então passará para modo de Ponto de Acesso WiFi. Isto vai permitir ao instalador conectar-se à rede do ESP32 e depois através do browser enviar um pedido HTTP do tipo GET ao ESP32. O microcontrolador, por sua vez, responde com uma página WEB com um conjunto de quatro campos de entrada: "ssid", que diz respeito ao nome da rede WiFi, "password" da rede WiFi, "id" que será a identificação do datalogger e "app" que será o equipamento em que foi aplicado. Quando o instalador submeter o pedido HTTP POST com os parâmetros de configuração, o ESP32 guarda as novas credenciais na memória flash e reinicia. Ao reiniciar volta a tentar conectar-se ao router, desta vez com as credenciais corretas. Quando tiver sucesso na ligação à rede WiFi ativa o modo de funcionamento normal, tal como foi apresentado no diagrama da figura 4.14. Durante este modo, o ESP vai ainda funcionar como um servidor WEB. Assim, permite ao instalador (connectado ao mesmo router) comunicar por HTTP com o ESP. Isto permite enviar novos parâmetros de configuração, como alterar a identidade do data logger, sem alterar as credenciais WiFi. Neste formulário há ainda a hipótese de o instalador abrir uma página "WEB Serial", que permite em tempo real obter informação acerca do estado de funcionamento do data logger, visto que o ESP mostra nesta página web mensagens de debug.

Para implementar a configurabilidade foi necessário incluir algumas bibliotecas e desenvolver código para que estas funções fossem executadas. Uma das principais bibliotecas utilizadas foi a biblioteca "WiFi.h"[10] que permite executar inúmeras funções relacionadas com redes WiFi. Para que o ESP se ligue em modo de Ponto de Acesso WiFi, foi utilizada a função "WiFi.softAP()". Esta função tem dois parâmetros de entrada: o nome que se quer dar à rede do ESP e a password. Depois de o instalador se conectar à rede do ESP ainda precisa de conhecer o IP que deve colocar no browser. Por defeito, todos os ESP32 utilizam o mesmo endereço IP, "192.168.4.1", quando se ligam em modo AP. Desse modo, qualquer que seja o data logger, o IP que o instalador deverá colocar no browser será o mesmo. Em modo AP, caso nenhum dispositivo se tente conectar à rede do ESP, ao fim de 30 segundos, o ESP reinicia. Deste modo, garante-se que o ESP não ficará infinitamente em modo AP, caso haja alguma falha.

Para permitir ao ESP atuar como servidor e processar pedidos HTTP enviados a partir de um browser foi necessário utilizar as bibliotecas "ESPAsyncWebServer"[11] e "AsyncTCP" [12]. Através destas bibliotecas, o ESP fica à escuta de pedidos HTTP, GET e POST. Se receber um pedido do tipo GET responde com uma página HTML que consite num formulário. Quando o instalador submeter os parâmetros de configuração, através do formulário, o ESP recebe um pedido POST. Ao receber este pedido, processa a mensagem, com o objetivo de extrair os parâmetros enviados na mensagem HTTP. Através de funções da biblioteca "SPIFFS.h" guarda esses parâmetros na memória flash e envia uma resposta HTTP a confirmar a receção do pedido. Estas funções são executadas em modo assíncrono. Isto signifca que o ESP estará sempre à escuta de pedidos HTTP, independentemente da tarefa que estiver a desempenhar, e chama as funções referidas como "callback" aos pedidos HTTP.

Conforme foi referido anteriormente, mesmo quando o data logger já estiver em modo de funcionamento normal é possível comunicar com ele pelo browser, através de um

[10]https://www.arduino.cc/reference/en/libraries/wifi/
[11]https://github.com/me-no-dev/ESPAsyncWebServer
[12]https://github.com/me-no-dev/AsyncTCP

dispositivo connectado à mesma rede WiFi. Porém, para isso seria necessário conhecer o IP com que o ESP se ligou ao router da casa do cliente. Como não foi incluído nenhum display, nem será considerada uma ligação a um "serial monitor", foi necessário arranjar forma de enviar pedidos HTTP ao ESP sem saber o IP com que este se ligou ao router. Para isso, utilizou-se a biblioteca "ESPmDNS.h"[13], que permite definir um "hostname" para o ESP, que fica associado ao seu endereço IP. Deste modo, inserindo no browser "http://hostname.local" consegue-se enviar pedidos HTTP ao ESP. Desta forma, será possível comunicar com o ESP, sem a necessidade de displays, "serial monitors" ou definição de um IP estático para o ESP. Esta última foi excluída, porque não seria versátil e adaptável a qualquer rede WiFi e poderia criar conflitos com outros equipamentos ligados ao router na casa do cliente.

Para auxiliar a instalação de um dispositivo destes foi também implementada uma forma de "debug" que não necessita de comunicação pela porta série de um computador. A solução passou por utilizar uma biblioteca designada "WebSerial.h"[14]. Esta biblioteca permite enviar mensagens de debug para um ecrã de monitorização que pode ser aberto na web. Desta forma, será possível continuar a receber mensagens de debug do data logger para comprovar que este está a funcionar corretamente.

Para que o ESP possa processar os pedidos HTTP realizados pelo instalador foi utilizada a função "server.on("/", HTTP_GET, (AsyncWebServerRequest *request)" da biblioteca assíncrona "ESPAsyncWebServer.h". Deste modo, quando o instalador escrever no browser "http://esp32.local/", esta função será chamada de imediato. Esta função envia uma resposta que consite num formulário desenvolvido através das linguagens HTML e CSS, o qual está apresentado na figura 4.17.

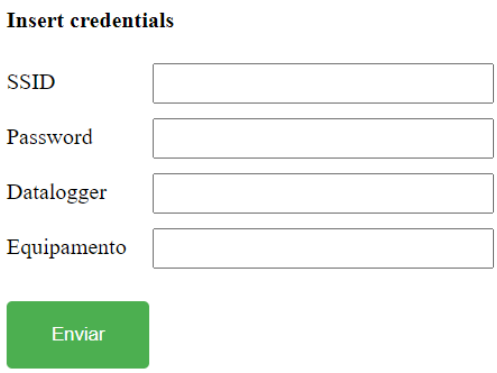

Figura 4.17: Página Web enviada pelo ESP para permitir a configuração do data logger

Conforme apresentado na figura 4.17, o formulário inclui quatro campos de entrada e um botão de "submit", que permite enviar um pedido POST ao ESP. Para processar esse pedido há outra função de "callback". Como nem sempre o instalador pretenderá alterar todos os parâmetros, aqueles que forem deixados em branco serão reconhecidos pelo ESP como inalterados. Os formulários desenvolvidos para o modo AP e para o modo de funcionamento normal são idênticos. No entanto, no segundo foi implementado um botão que se for pressionado, pede ao ESP a página Web Serial. Ao longo do código do microcon-

[13]https://github.com/espressif/arduino-esp32/blob/master/libraries/ESPmDNS/src/ESPmDNS.h
[14]https://github.com/ayushsharma82/WebSerial

trolador foram incluídas funções "WebSerial.print()" com mensagens de debug que irão ser apresentadas no browser. As mensagens mais relevantes dizem respeito a erros de comunicação com sensores, a alguns dados reocolhidos, a mensagens de confirmação de envio de dados ao servidor, etc. A página Web apresentada para este efeito faz parte da biblioteca "WebSerial.h" utilizada e não foi desenvolvida durante a implementação deste trabalho. Por esse motivo inclui também um campo de entrada que não será utilizado.

Implementação da recolha e registo de dados no cartão SD

Nesta secção será apresentada a forma como foi implementada a recolha de dados de todos os sensores e das ECUs e sucessivo registo no cartão SD. Esta parte do código do ESP32 foi implementada na função "void Task1()" do código Arduino e é desempenhada pelo núcleo 0 do ESP. Dentro desta função são chamadas todas as funções necessárias para recolher os dados dos vários sensores, gestão de tempo, processamento das mensagens recebidas a partir da ECU e registo de todos os dados recolhidos no cartão de memória nos ficheiros corretos.

Conforme apresentado no diagrama da figura 4.15 a rotina de recolha de dados opera em ciclos de 1 s. Ou seja, o ESP32 contabiliza o tempo que demora a realizar um ciclo, e no final da leitura espera o tempo que falta para os 1000 ms. Quando atinge os 1000 ms inicia uma nova medição. O ciclo de recolha de dados começa pela verificação do estado da ligação WiFi ao router. Caso a ligação esteja "ON", chama uma função "printTimeStamp()", que permite estabelecer uma ligação a um servidor NTP e atualizar a data e hora atual. Para armazenar este parâmetro foi utilizada a estrutura "struct tm", através da biblioteca "time.h". Esta estrutura inclui várias variáveis que guardam por exemplo o ano, mês, dia, hora, minutos e segundos. Caso a ligação WiFi ao router esteja "OFF", o ESP incrementa 1 s na estrutura do "timestamp". Isto permite que mesmo que não haja ligação WiFi, o ESP consiga manter o timestamp da recolha de dados sincronizado com a hora real.

De seguida, o ESP inicia a recolha de dados dos sensores, começando pelo cálculo do caudal medido pelo caudalímetro. Para este efeito, foi utilizada uma rotina de interrupção que contabiliza o número de pulsos do sensor. Para executar os cálculos, o ESP32 considera o número de pulsos contabilizados desde a última medição e o tempo que passou desde a última medição. Para chegar ao valor do caudal, o ESP tem de fazer dois cálculos. O primeiro é efetuado através da equação 4.3 e permite chegar à frequência do sinal de saída do sensor.

$$f = \frac{n_{\text{pulsos}}}{\Delta t} \quad [\text{pps}] \tag{4.3}$$

A partir da frequência, recorreu-se à reta de calibração dada na datasheet do sensor, que permitiu chegar à equação 4.4 para calcular o caudal.

$$caudal = \frac{f + 2.76}{6.65} \quad [\text{l/min}] \tag{4.4}$$

Depois de feita a leitura do caudal de água, o ESP faz a recolha da temperatura medida pelo sensor termopar MAX6675. Para este sensor foi utilizada a biblioteca "max6675.h"[15], que permite criar um objeto max6675 e utilizar a função "readCelsius()" para ler a temperatura em ºC. Este sensor, tal como referido, comunica com o ESP32 pelo

[15]https://github.com/adafruit/MAX6675-library

protocolo de comunicação SPI. Inicialmente, o pino MISO do sensor tinha sido ligado ao GPIO 12 do ESP. No entanto, os testes realizados revelaram que esta ligação despoletaria problemas sempre que se ligasse o ESP. Aquilo que acontecia era que utilizando o GPIO 12, como este pino estava a ser alterado durante o "boot" do ESP, este não conseguia arrancar. Para resolver o problema decidiu-se trocar este pino para o GPIO 35. Tal como foi apresentado no diagrama 4.15, o ESP verifica o estado da comunicação com este sensor. Como a biblioteca não inclui uma função para verificar a comunicação com o sensor, a verificação do "status" da ligação ao sensor passou por analisar o resultado da função "readCelsius". Caso esta função retornasse o valor "0.00" então conclui-se que a probabilidade de a comunicação com o sensor estar a falhar é muito alta e por isso substitui-se esse valor por um código de erro, que neste caso será "666.66". Este código permite, da perspetiva da visualização dos dados, perceber de forma remota que algo está errado com o sensor.

Depois do MAX6675, o ESP faz a leitura do sensor BME280, sendo que neste caso são lidos três parâmetros, a temperatura ambiente, a pressão atmosférica e a humidade do ar. Para isso foram usadas as bibliotecas "Adafruit_BME280.h"[16], "Adafruit_Sensor.h" [17] e "Wire.h"[18]. Através destas bibliotecas foi possível criar um barramento I2C para a comunicação com o sensor, verificar o estado da comunicação e fazer as leituras dos parâmetros desejados.

De seguida é feita a medição da pressão hidráulica a partir do sensor de pressão. Este sensor possui uma interface I2C, de modo que é possível utilizar a biblioteca "Wire.h" para fazer a aquisição da pressão hidráulica. O código utilizado para a leitura deste sensor foi adaptado a partir de um código fornecido pelo fabricante. O output é uma grandeza digital com base decimal que permite calcular a pressão medida pelo sensor em termos percentuais, visto que são grandezas diretamente proporcionais. De seguida, a partir da pressão percentual pode ser calculada a pressão hidráulica, considerando a gama de medição do sensor. Para isso foi utilizada a equação 4.5, fornecida pelo fabricante.

$$P_{bar} = \frac{P_{\%} \times 100 - 1000}{26.67 \times 6.895 \times 100} \quad \text{[bar]} \tag{4.5}$$

De seguida, o ESP faz a leitura da corrente medida pelo sensor ACS712. Ao contrário dos anteriores, este sensor foi alvo de um conjunto de testes e de um processo iterativo para chegar a resultados aceitáveis. Primeiramente, com este sensor pretende-se medir corrente AC, o que dificulta o processo de medição da corrente. Por esta razão, o ESP tem de fazer a leitura do sensor de forma a conseguir obter os valores máximo e mínimo da curva, para depois calcular a corrente eficaz. Tal como foi explicado anteriormente, o output deste sensor é um sinal de tensão analógico proporcional à corrente aplicada, variando entre 2.5 e 5 V para corrente positiva e 0 e 2.5V para correntes negativas. Posto isto, depois de passar pelo divisor resistivo implementado, o ESP tem de fazer a conversão analógica digital deste sinal, calcular a tensão correspondente e depois a corrente, a partir do parâmetro de sensibilidade do sensor. A dificuldade surgiu na necessidade de obter os picos da curva da corrente. Fazendo apenas uma medição seria impossível garantir que se estaria a medir o pico da onda. Assim, a estratégia passou por considerar um intervalo de tempo igual a um período completo da onda, ou seja 20 ms, visto que a frequência da

[16]https://github.com/adafruit/Adafruit$_{B}ME280_{L}ibrary$
[17]https://github.com/adafruit/Adafruit$_{S}ensor$
[18]https://www.arduino.cc/reference/en/language/functions/communication/wire/

rede é de 50 Hz e durante esse intervalo de tempo fazer o máximo de medições possíveis. Dessas medições seriam guardados os valores de pico. Os picos de corrente iriam provocar os picos máximo e mínimo de valores digitais no ESP. A partir da diferença entre ambos pode ser calculada a tensão de amplitude entre os picos, utilizando a equação 4.6.

$$V = (max - min) \times \frac{3.3}{4096} \tag{4.6}$$

Nesta equação o termo "3.3" corresponde à tensão máxima de leitura do ESP e o "4096" ao valor digital correspondente, visto que o ESP faz uma conversão digital de 12 bits e $2^{12} = 4096$. Com este resultado pode ser calculada a tensão eficaz, pela equação 4.7.

$$V_{ef} = \frac{V}{2} \times \frac{\sqrt{2}}{2} \tag{4.7}$$

A partir deste valor de tensão eficaz calcula-se a corrente eficaz tendo em conta a sensibilidade do sensor ACS712 utilizado que é de 66 mV/A. Para isso utiliza-se a equação 4.8.

$$I_{ef} = \frac{V_{ef} \times 1000}{Sensibilidade} \tag{4.8}$$

Deste modo, chega-se finalmente ao valor da corrente eficaz. Mesmo quando a corrente a passar no sensor for nula, a tensão de saída não será exatamente metade do valor máximo medido pelo ESP. Por esse motivo, foi necessário determinar o valor medido enquanto a corrente a passar no sensor fosse nula e por sua vez aplicar esse valor como offset.

Para fazer a leitura da tensão analógica à saída do sensor de corrente será utilizado o ESP em vez das placas ADS1115. Esta troca deve-se à necessidade de realizar várias conversões A/D sucessivas num intervalo de tempo muito curto. Utilizando as placas ADS1115 não é possível realizar mais do que 2/3 medições por período da onda, enquanto que utilizando a conversão A/D do ESP é possível realizar mais de 300 medições. Para que se consiga obter os valores de pico da onda é necessário realizar um número de medições muito significativo, por isso foi utilizada a conversão A/D do microcontrolador. Como a frequência da onda AC é de 50 Hz, o período da onda é de 20 ms. Realizando 300 medições em 20 ms, obtém-se uma medição a cada 0.067 ms, o que é uma precisão bastante alta.

Finalmente, o ESP faz a medição das temperaturas dos sensores NTC. Conforme referido anteriormente, para fazer a medição da temperatura foram usadas placas de conversão A/D externas que comunicam por I2C com o microcontrolador. A medição da temperatura consiste na execução de alguns passos:

1. Conversão A/D da tensão aos terminais do sensor NTC;
2. Cálculo da respetiva resistência elétrica do sensor;
3. Utilização da curva temperatura vs resistência do sensor para chegar à temperatura correta.

O primeiro passo consiste num pedido enviado, por I2C, pelo ESP às placas ADS1115 pelos valores digitais nas entradas analógicas das placas. Esta interação foi implementada através da biblioteca "Adafruit_ADS1X15.h"[19] desenvolvida pela Adafruit. Caso a

[19]https://github.com/adafruit/Adafruit$_ADS1X15$

interação com as placas não seja estabelecida, o ESP não tenta pedir os valores digitais e atribui um código de erro às temperaturas que seriam lidas por essa placa. Se a interação for estabelecida, para cada uma das entradas analógicas da placa, o ESP chama a função ".readADC_SingleEnded()" para pedir o valor digital proveniente da conversão A/D para cada um dos sensores. De seguida é utilizada a equação 4.9 para calcular a resistência do sensor.

$$R_2 = R_1 \times \frac{D \times \frac{0.1875}{1000}}{3.33 - D \times \frac{0.1875}{1000}} \tag{4.9}$$

Na equação 4.9, D corresponde ao valor digital fornecido pelas placas ADS1115 ao ESP32, R_1 corresponde à resistência utilizada no divisor resistivo (foram utilizadas resistências elétricas de $10k\Omega$) e R_2 corresponde à resistência do sensor. Neste cálculo também é calculada a tensão a partir do valor digital utilizando o parâmetro 0.1875 mV, que corresponde à resolução da placa. O valor 3.33 corresponde à tensão de alimentação da placa. Depois de obtida a resistência do sensor, o cálculo da temperatura é feito pela equação 4.10.

$$T_{\mathrm{NTC}} = \frac{1}{\frac{1}{298.15} - \frac{log_{10}(\frac{R_1}{R_2})}{log_{10}(e) \times B_{\mathrm{value}}}} - 273.15 \tag{4.10}$$

Na equação anterior, $B_{value} = 3435$. Esta equação foi obtida a partir da datasheet do sensor e dos respetivos valores de temperatura associados a cada resistência.

Depois da recolha dos dados a partir de todos os sensores externos, é realizada a descodificação das mensagens recebidas na interface UART, vindas das ECUs às quais o data logger está ligado. Para isto foram considerados três equipamentos diferentes, com os quais é possível comunicar com a ECU. Estes equipamentos foram o EWI DNA, um esquentador totalmente elétrico, o KME e o FP2, ambos esquentadores a gás ventilados. As mensagens enviadas pelas ECUs destes equipamentos são diferentes. Por isso, foi necessário fornecer ao ESP32 a capacidade de distinguir com qual dos equipamentos está a comunicar. A estratégia adotada para resolver esta questão será apresentada posteriormente.

No protocolo utilizado pelas ECUs, está previsto que as ECUs enviem mensagens de segundo a segundo a um periférico que se ligue à interface UART, com um conjunto de parâmetros de operação do equipamento. Alguns destes parâmetros são: temperaturas de entrada e saída da água, caudal e potência. A estrutura da mensagem é semelhante de ECU para ECU, sendo que a principal diferença está nos dados que são enviados dentro da mensagem e na sua descodificação. Além das mensagens desejadas, as ECUs também enviam outros tipos de mensagens, com outros propósitos. Todavia, para este trabalho, pretende-se apenas utilizar as mensagens que fornecem parâmetros de operação do equipamento. Para distinguir as mensagens desejadas de outras é necessário avaliar um dos bytes que, de acordo com o protocolo, indica o tipo de mensagem. Por motivos de confidencialidade, detalhes mais específicos do protocolo não poderão ser abordados.

Para desenvolver o software necessário para processar as mensagens recebidas na interface UART utilizou-se a biblioteca Serial[20] do Arduino IDE. As funções utilizadas foram:

- Serial.begin(), que permitiu inciar uma comunicação série entre o ESP e a ECU,

[20]https://www.arduino.cc/reference/en/language/functions/communication/serial/

utilizando como parâmetros de entrada o baudrate de 9600 bps (definido pelo protocolo) e os pinos RX e TX do ESP.

- Serial.available(), que permite verificar se existem dados no buffer. Quando o ESP recebe bytes pela interface UART, esses bytes são armazenados num buffer antes da respetiva leitura.

- Serial.read(), que permite ler os bytes armazenados no buffer. Esta função, sempre que é chamada, faz a leitura do último byte guardado no buffer.

O ESP lê byte a byte enquanto houver bytes disponíveis no buffer. Caso encontre os dois bytes que representam o início de uma mensagem, começa a guardá-los num array. A partir daí, caso o ESP encontre os bytes que correspondem ao final da mensagem, pára a leitura e analisar o conteúdo da mensagem. O primeiro passo consiste em determinar se a mensagem recebida é ou não a mensagem esperada, através de um dos bytes. Se a mensagem for a correta, o ESP guarda alguns dos parâmetros, como temperaturas de entrada e saída da água, caudal, potência e setpoint em variáveis, que depois regista no cartão de memória. A estrutura das mensagens das ECUs de cada um dos equipamentos é semelhante, no entanto as posições e a forma de calcular determinados parâmetros varia. O código desenvolvido para retirar os parâmetros mais relevantes foi desenvolvido com base na documentação providenciada pela empresa. Para garantir que a leitura dos bytes armazenados no buffer não causaria um "overload" do ESP, foi também implementada uma condição em que, caso sejam guardados mais de 120 bytes, o ESP reinicia a leitura das mensagens enviadas pela ECU e ignora os bytes anteriores. Este valor foi definido com base nas dimensões das mensagens previstas neste protocolo.

A última etapa do ciclo de recolha de dados corresponde ao armazenamento dos mesmos no cartão de memória. Para simplificar a organização dos dados foi feita uma divisão por vários ficheiros de texto, nomeadamente ficheiros apenas para o registo de dados relativos aos sensores externos, ficheiros para armazenar toda a mensagem da ECU e ficheiros para os parâmetros mais relevantes de cada uma das diferentes ECUs. Desta forma, cada tipo de ficheiro terá uma estrutura específica com base no tipo de dados que deverá reter. A forma de registo de dados nos ficheiros passa por ter uma linha por cada ciclo de medições. Todos os dados guardados serão do tipo "float", com exceção do timestamp. Por este motivo, a melhor forma encontrada para organizar os ficheiros foi registar as várias "floats" separadas por vírgulas (formato csv) e no final da linha o respetivo timestamp.

Para implementar o registo dos dados nestes ficheiros de texto foram utilizadas as bibliotecas "SPI.h"[21] e "SD.h"[22], sendo que a primeira permite estabelecer as comunicações com a placa SPI onde é colocado o cartão de memória e a segunda permite a gestão dos vários ficheiros no cartão SD. O cartão SD foi organizado em dez ficheiros de texto, dois para cada tipo de dados. Esta organização foi desenvolvida com o objetivo de tornar o datalogger mais flexível. Assim, caso não haja comunicação com uma ECU então só os ficheiros dos sensores serão considerados. Além disso, como os vários equipamentos têm diferentes parâmetros relevantes, desta forma é possível definir uma estrutura única para cada um deles. A razão para terem sido criados dois ficheiros para cada tipo de dados está relacionada com a interação entre a escrita e leitura dos dados no cartão SD.

[21] https://reference.arduino.cc/reference/en/language/functions/communication/spi/
[22] https://github.com/arduino-libraries/SD

O dispositivo de armazenamento atua como intermediário entre os dados recolhidos pelo núcleo 0 e os dados enviados para o servidor pelo núcleo 1. À medida que os dados são enviados para o servidor, estes são apagados para que não se acumulem dados no cartão de memória que já estarão na base de dados. A gestão implementada consiste em: quando o núcleo 1 iniciar a leitura dos dados de um ficheiro, o outro núcleo saberá e então começará a escrever os dados no outro ficheiro. Deste modo, quando o núcleo 0 começar a ler um ficheiro, já não haverá mais dados a serem escritos e quando terminar de ler os dados poderá apagar o ficheiro sem provocar perdas de dados. Por sua vez, no ciclo seguinte, quando o núcleo 1 iniciar a leitura do outro ficheiro, o núcleo 0 volta a trocar onde escreve os dados. Caso o ficheiro não exista (porque foi apagado), o ESP cria um ficheiro com o mesmo nome logo de seguida.

Além dos parâmetros relevantes de cada um dos equipamentos decidiu-se também registar a totalidade da mensagem enviada pela ECU, independentemente do equipamento ao qual está ligado. O objetivo desta função é permitir que o data logger possa ser instalado num novo equipamento, que ainda não conheca os parâmetros, mas mesmo assim permita o registo de dados. Desta forma, como o ESP não sabe que parâmetros deve retirar da mensagem, simplesmente regista a mensagem toda e depois esse processamento poderá ser realizado do lado do servidor, caso seja necessário. Além disso, isto permite que caso um utilizador queira ver algum parâmetro específico que não foi considerado entre os mais relevantes, também tem essa possibilidade.

No final de cada ciclo de recolha de dados um dos LEDs pisca de forma sinalizar que o ciclo de recolha de dados terminou.

Implementação da leitura e envio de dados do cartão SD para o servidor

No modo de funcionamento normal, o núcleo 1 do ESP executa a tarefa de ler os dados que estão no cartão SD, registados pelo outro núcleo, e enviá-los para o broker por MQTT. Por este motivo, núcleo 1 tem ainda a função de manter a ligação WiFi com o router e a ligação MQTT com o broker. Conforme representado no diagrama da figura 4.16, o ESP começa por chamar três funções: "connectWiFi()", "printTimeStamp()" e "connectMQTT()". A primeira função tem como objetivo verificar o estado da ligação WiFi com o router. Caso esta não esteja estabelecida, deve conectar-se. Conectado ao router, o ESP atualiza a data e hora atual através de um pedido enviado a um servidor NTP (network time protocol). Depois disso, estabelece uma ligação MQTT com o broker. Para implementar a componente MQTT foi utilizada a biblioteca "PubSubClient.h"[23]. Dentro da função "connectMQTT()", caso a ligação ao broker esteja "OFF", o ESP tenta conectar-se ao broker. Nesta fase haverá tentativas sucessivas até que a ligação seja estabelecida. Depois de estabelecidas as ligações necessárias, o ESP analisa quanto tempo passou desde o último envio de dados ao servidor. Caso já tenham passado mais de 10 segundos, inicia a leitura dos dados dos sensores. Nesta função, a primeira tarefa é decidir que ficheiro ler. Sempre que a função é chamada, é lido o ficheiro que ficou por ler na última vez que a função foi chamada. Nesta fase é importante lembrar que o registo de dados é efetuado com parâmetros separados por vírgulas, e que são registados linha a linha. Por este motivo, foi utilizada a função "readStringUntil('\n')", que permite ler todos os carateres e armazená-los numa variável "String", até que encontre uma mudança de linha. Quando encontrar a mudança de linha significa que já guardou todos os dados

[23]https://github.com/knolleary/pubsubclient

relativos a um ciclo de recolha de dados. De seguida, foi implementada uma função que permite separar os parâmetros escritos nessa variável "String" por vírgulas de forma a ter um "array" de dados. Com esse array é criado um objeto JSON, através da biblioteca "ArduinoJson.h", onde são guardados os vários parâmetros com a respetiva chave de identificação. Posteriormente, esse objeto JSON é enviado para o servidor através da função ".publish()" da biblioteca "PubSubClient.h". Esta mensagem é publicada num tópico designado por "DL/sensores". Esta biblioteca prevê duas formas de o envio da mensagem dar erro. A primeira é se a mensagem for demasiado longa, sendo que para mitigar essa limitação foi utilizada a função "setBufferSize()", capaz de aumentar o limite máximo do número de bytes da mensagem. Além disso, se houver uma falha de conectividade, o envio também poderá falhar. Caso essa falha ocorra, o ESP chama as funções "connectWiFi()" e "connectMQTT()" para reestabelecer a conectividade e, depois disso, volta a tentar enviar a mensagem. A partir daí volta ao início para verificar se ainda há mais linhas de dados para ler e, caso haja, continua este ciclo de operação. Caso não haja mais dados, significa que já enviou todos para o servidor e pode, então, apagar o ficheiro que acabou de ler. Para que localmente se possa perceber que o envio de dados para o servidor está a funcionar como pretendido, de cada vez que uma mensagem é publicada no broker um dos LEDs pisca. Quando termina de ler os dados e apaga o ficheiro, também é realizado um sinal através dos LEDs.

Depois de ler os dados dos sensores é realizada uma função semelhante tanto para as mensagens das ECUs como para os parâmetros mais relevantes do equipamento ao qual está ligado.

4.2 Gestão dos dados em backend

No final da primeira parte do trabalho foi possível ter um protótipo de um data logger capaz de estabelecer ligações à internet, recolher dados de um conjunto de sensores e das ECUs dos aparelhos e publicar esses dados num broker MQTT. Tendo ínumeros dispositivos destes a enviarem dados para um servidor, é necessário implementar uma estrutura que permita a gestão desses dados. Nesta secção será apresentada a implementação das tarefas que serão executadas em backend[24].

4.2.1 Arquitetura

A solução implementada inclui um conjunto de ferramentas, que são integradas numa plataforma a correr na cloud. Deste modo, todas as funções do backend são executadas na internet, sem necessidade de haver um servidor local sempre ligado. Na figura 4.18 está representada a arquitetura implementada em backend. Nesta arquitetura a informação chega através do broker MQTT, onde há um agente que subscreve todos esses dados e os envia para a base de dados. Além disso, há uma sequência de funções de processamento que interagem com a base de dados. Há ainda um conjunto de ficheiros de configuração que são chamados pelas várias funções, quando for necessário. Por último, há uma ferramenta de visualização que permite enviar queries à base de dados, para os apresentar em dashboards.

[24]conjunto de tarefas que são executadas em segundo plano sem interagir com um utilizador

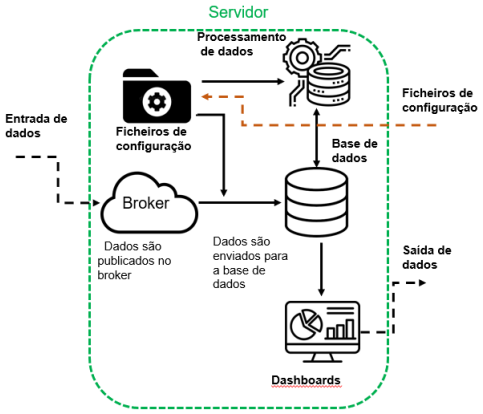

Figura 4.18: Arquitetura da solução proposta para a gestão de dados em backend

Ferramentas selecionadas

Para cumprir os requisitos foi selecionado um conjunto de ferramentas para desempenhar cada uma das seguintes funções: agente intermediário entre os dataloggers e a BD, linguagem de programação para implementar o processamento, base de dados, formato para ficheiros de configuração e ferramenta de visualização.

Máquina Virtual Linux na Microsoft Azure

Para integrar toda a arquitetura em backend e permitir que todas as funções sejam executadas na cloud selecionou-se a Microsoft Azure. Esta plataforma oferece um conjunto de serviços na cloud, como máquinas virtuais (MV), armazenamento de dados, análise de dados, dashboards, etc. Neste trabalho foi apenas utilizada uma máquina virtual alojada na cloud através desta plataforma. Nesta MV foram instaladas todas as ferramentas que serão apresentadas de seguida. A grande vantagem da utilização de uma máquina virtual assenta na sua alta configurabilidade e segurança.

Broker Mosquitto

Para permitir enviar dados para o servidor foi selecionado o broker Mosquitto. Esta ferramenta atua como agente intermediário entre os data loggers e o servidor onde podem ser publicados e subscritos dados. Esta comunicação utiliza o protocolo MQTT. Nesta arquitetura, os vários data loggers publicam dados em tópicos no Mosquitto, e há um programa que subscreve esses dados. Para desempenhar esta função havia outras opções. Porém, com base na análise do estado de arte, ferramentas como Apache Kafka seriam demasiado poderosas para as necessidades desta aplicação, e o Mosquitto tem a capacidade necessária para as exigências que lhe serão impostas.

InfluxDB

Quanto ao armazenamento de dados foi selecionada a base de dados InfluxDB. Esta ferramenta está muito otimizada para a gestão de séries temporais e tem alta flexibilidade no envio dos dados para a BD. Com base no estado da arte, outras bases de dados têm algumas limitações, tanto ao nível da gestão temporal como ao nível da flexibilidade

que proporcionam. Por exemplo, bases de dados MySQL têm regras de integridade que impedem a falta de valores ou que estes sejam inválidos. No caso de outras ferramentas, como a MongoDB, não há um foco na gestão temporal dos dados. O InfluxDB permite definir um período de retenção de forma que, os dados são retidos na BD apenas durante esse período. Consoante o tipo de dados podem ser criados diferentes "buckets" com diferentes períodos de retenção. Além disto, esta base de dados tem a própria linguagem de programação "Flux" que é utilizada para interagir com a BD. Esta linguagem permite executar praticamente qualquer query que se pretenda e pode ainda ser utilizada em diversas ferramentas de programação, através de bibliotecas, e em diversas ferramentas de visualização. Deste modo, selecionando o InfluxDB como base de dados, a solução torna-se muito escalável e versátil.

Python

Para interligar os vários agentes intervenientes no backend será utilizada a linguagem de programação Python, devido à sua versatilidade e possibilidade de incluir inúmeras bibliotecas que permitem desempenhar inúmeras funções. Assim haverá um conjunto de *scripts* que serão executados com o objetivo de subscrever tópicos no broker, enviar os dados para a base de dados, enviar queries ao InfluxDB e processar as séries temporais.

YAML

Para desenvolver os ficheiros de configuração foi utilizado o formato YAML. Este formato tem uma syntax muito fácil de ler e de escrever, constitundo a sua principal vantagem relativamente ao formato JSON. Existem bibliotecas desenvolvidas especificamente para chamar ficheiros em formato YAML e extrair as configurações para variáveis.

Grafana

Para apresentar os dados em dashboards foi selecionado o Grafana, dada a sua grande compatibilidade com a base de dados InfluxDB e possibilidade de criar dashboards dinâmicas e interativas. Através do Grafana serão enviadas queries à base de dados InfluxDB, utilizando também a linguagem Flux.

Fluxo de informação

Na secção anterior foram apresentadas as ferramentas e a arquitetura implementada. Contudo, não fornece uma boa perceção da forma como a informação flui entre os agentes intervenientes. No backend há, essencialmente, dois fluxos de informação: um que consite no envio dos dados dos data loggers para a base de dados e outro que consiste numa interação entre os scripts de processamento de dados e a base de dados. Na figura 4.19 está representado o diagrama UML que representa o fluxo de informação desde o data logger até à base de dados. Conforme representado no diagrama 4.19, a informação é criada e depois enviada pelo data logger para o broker. De imediato, passa para o script que envia os dados para a BD, onde os dados são formatados e inseridos na BD.

Além disto, na figura 4.20 está representado um diagrama UML com o fluxo de informação geral para o processamento dos dados. No processamento, há várias rotinas independentes, cada uma com alguns detalhes específicos, contudo, o fluxo de informação geral é compatível com todas. Conforme apresentado no diagrama da figura 4.20, o fluxo de informação começa com o envio de uma query à base de dados. Esta query tem o

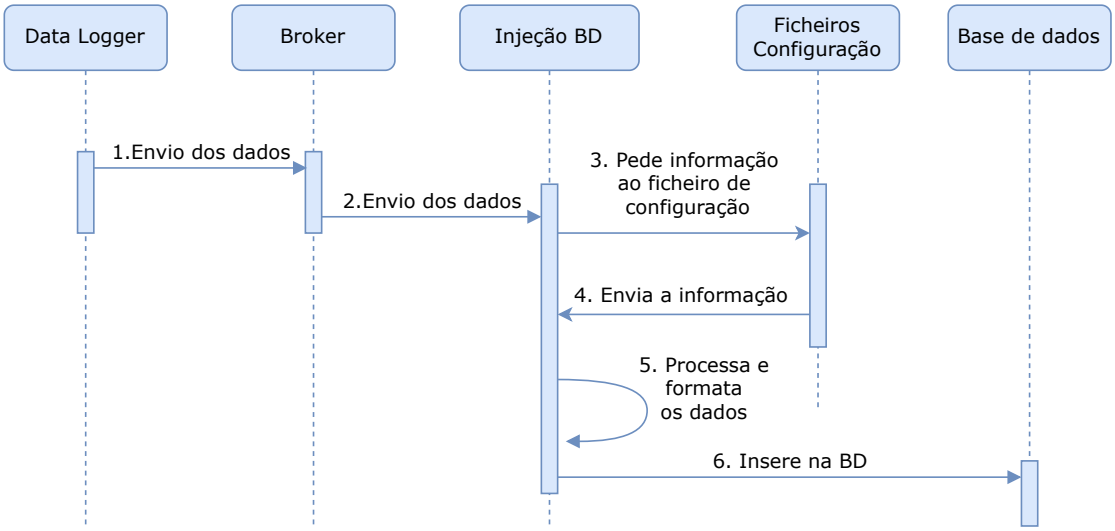

Figura 4.19: Diagrama UML do fluxo de informação para o envio dos dados do data logger para a base de dados

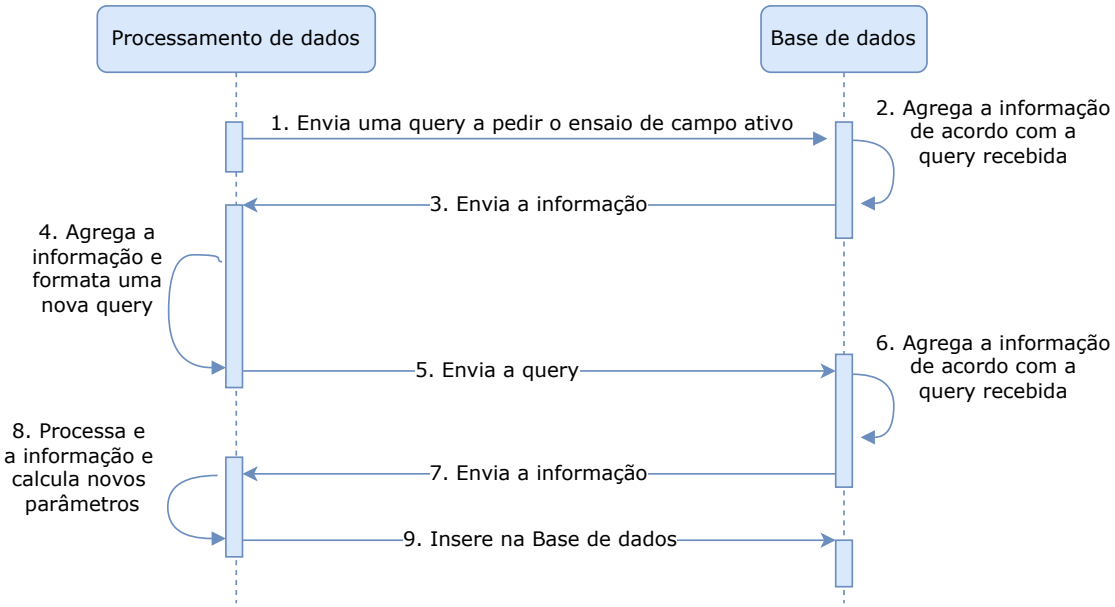

Figura 4.20: Diagrama UML do fluxo de informação entre o processamento e a base de dados

objetivo de solicitar o ensaio de campo ativo para o respetivo data logger. Depois, a base de dados envia a informação de acordo com essa query. De seguida, toda essa informação é agregada para formatar uma nova query que é enviada à BD, que envia de volta a informação solicitada. Nessa altura, é aplicado um método de processamento de dados, (que pode ser cálculo de métricas, deteção de outliers ou cálculo de extrapolações) cujo resultado é inserido na base de dados.

4.2.2 Ficheiros de configuração

Os ficheiros de configuração têm como objetivo efetuar quaisquer configurações aos data loggers que possam ser feitas apenas do lado do servidor e não tenham de ser realizadas localmente. Uma das configurações presente nestes ficheiros é a informação acerca de quais os sensores que foram implementados no data logger. Esta informação tem como objetivo formatar os valores enviados pelo data logger para um código específico, caso no ficheiro de configuração esteja indicado que esse sensor não está a ser utilizado. Além disso, tem como objetivo fornecer parâmetros e informações necessárias para a gestão dos dados em backend. Um dos parâmetros mais relevantes permite definir se o cálculo de métricas deve ser feito através dos dados dos sensores ou dos parâmetros recolhidos a partir da ECU. Além disso, o ficheiro de configuração também permite dar uma identidade ao ensaio de campo realizado e especificar o modelo do esquentador. Sempre que são enviados dados para a base de dados, estes dois parâmetros são incluídos. Isto tem o objetivo de permitir a separação das séries temporais entre equipamentos e ensaios.

Para cada data logger haverá um ficheiro de configuração. Desse modo, quando o data logger envia dados ao servidor, envia também a sua identidade e, consequentemente, o script de inserção de dados na BD saberá qual o ficheiro de configuração a chamar. Na figura 4.21 está apresentado um exemplo de um ficheiro de configuração desenvolvido para testes efetuados durante a fase de implementação. O nome do ficheiro de configuração

```yaml
sensores:
  T1: 1
  T2: 1
  T3: 1
  T4: 1
  T5: 1
  T6: 1
  T7: 1
  T8: 1
  T9: 1
  T10: 1
  T11: 1
  T12: 1
  F: 1
  P_h: 1
  T_h: 0
  I: 1
  atm: 1
  Tamb: 1
  T_max: 1

Appliance:
  powerFP2: 21
  powerEWI: 27
  powerKME: 27

field_test:
  status: 1
  app: "FP2_15_L"
  id: "Cap5_testeNormal_B"

metrics:
  ecu: 1
```

Figura 4.21: Exemplo de um ficheiro de configuração utilizado

apresentado na figura anterior é "DL2". Este é o nome de um data logger e é a identidade que o data logger envia para o servidor, junto das mensagens de dados. Em backend, enquanto os dados são publicados no broker, há um cruzamento das mensagens recebidas com o ficheiro de configuração. Contudo, para que isso aconteceça, a identidade atribuída ao data logger terá de corresponder a um dos ficheiros de configuração no servidor.

4.2.3 Organização da base de dados

A estratégia da organização dos dados no InfluxDB é extremamente importante para que todo o sistema funcione corretamente. Esta organização afeta praticamente tudo o resto, desde as queries enviadas pela ferramenta de visualização, ao processamento dos dados e ao script que subscreve os dados publicados no broker. Antes de apresentar a abordagem implementada é importante introduzir alguns conceitos relativos ao modelo da base de dados InfluxDB.

- **Organização:** área de trabalho partilhada por um conjunto de utilizadores. Numa organização podem ser geridos buckets, tarefas, membros, etc.
- **User:** membro de uma organização. Cada user terá um username e uma password. Os users podem ter diferentes funções e daí diferentes permissões.
- **Token:** chave de acesso gerada por um user que substitui a autenticação por username e password.
- **Bucket:** elemento principal de armazenamento de dados. Todos os dados são armazenados em buckets. Este elemento pode ser descrito como uma base de dados com um determinado período de retenção.
- **Measurement:** segundo maior elemento. Atua como um "container" de fields, tags e timestamps, dentro de um bucket. Permitem agrupar dados consoante características comuns a grandes quantidades de dados.
- **Field:** elemento base, que consiste numa chave, guardada na coluna "_field" e o respetivo valor, guardado na coluna "_value".
- **Tag:** permite definir colunas para os dados, ou seja, atua como uma etiqueta que permite separar os dados consoante características mais voláteis.
- **Time:** todos os dados guardados no InfluxDB têm de estar associados a uma data e hora, que serão guardadas na coluna "_time".

Com base no modelo da base de dados InfluxDB foi criado um diagrama de classes UML para representar a organização da base de dados deste trabalho (ver figura 4.22). Conforme o diagrama da figura 4.22, foi criada uma organização, cujo nome atribuído foi "Projeto Data Logger". Esta organização terá um ou mais *Users*. Os *Users* têm uma role que pode ser de utilizador, cujas permissões passam apenas por ler dados da base de dados, ou administrador, que têm acesso total à base de dados. Além dos *Users*, a organização contém sete buckets, cada um deles com um nome e um período de retenção. Destes sete buckets, dois fazem parte do sistema e, por isso, apenas cinco serão utilizados para o projeto. Neste caso, os buckets serão utilizados para separar os dados consoante a categoria. Haverá um bucket para cada uma das seguintes categorias: séries temporais, métricas, outliers, mensagens das ECUs e extrapolações. Deste modo, para cada categoria de dados, pode ser definido um período de retenção diferente.

Dentro de cada bucket haverá várias *measurements*, às quais estão associados os diferentes data loggers. O nome de cada *measurement* será o nome de cada data logger. Deste modo, é possível agrupar os dados que têm o respetivo data logger em comum. Dentro de cada *measurement* serão armazenados os "pontos" enviados por cada data

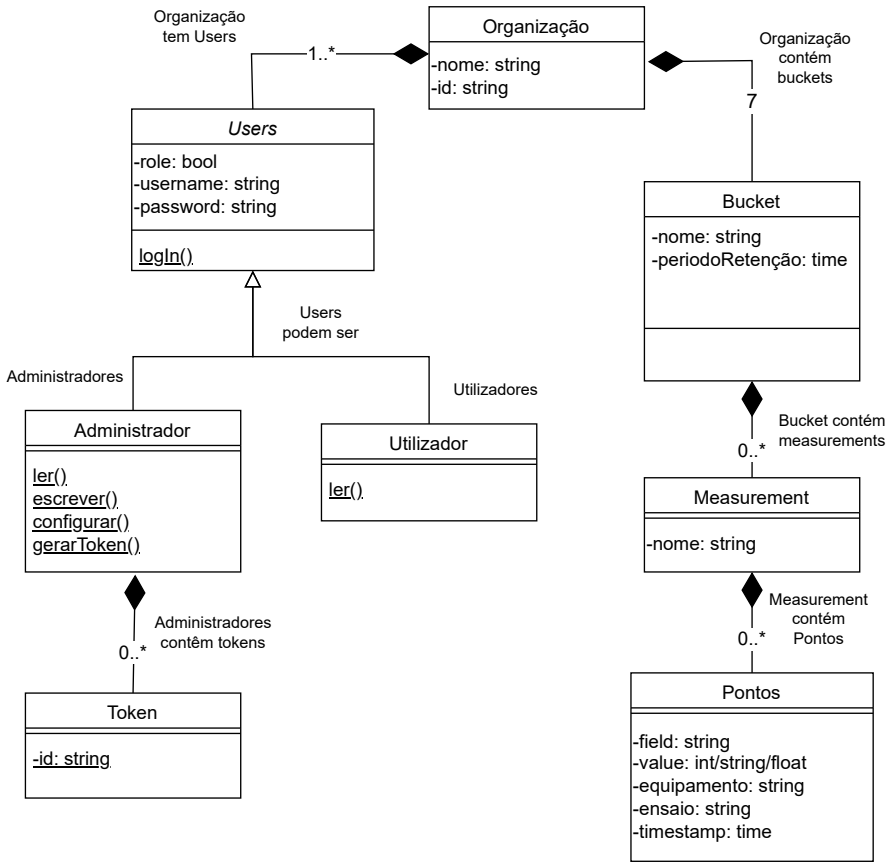

Figura 4.22: Diagrama de classes UML para representar a estrutura da base de dados utilizada neste trabalho

logger. A cada ponto corresponde: um timestamp único, uma "appliance" (equipamento do qual está a recolher dados) e um "fieldtest" (ensaio de campo). Para cada ponto pode haver um ou mais *fields*. Cada um destes *fields* é formado por um par *key-value*. O resultado desta organização será uma estrutura em que um utilizador pode enviar queries à base de dados aplicando vários tipos de filtros. Assim, será possível escolher todos os dados de um determinado data logger (*measurement*), ou de uma determinada "appliance", apenas um determinado "field", ou mesmo juntar vários filtros para chegar a um objetivo específico. No apêndice D está apresentado um exemplo de uma query, bem como o respetivo resultado fornecido pela BD.

4.2.4 Receção dos dados e envio para a base de dados

Para permitir a receção dos dados publicados no broker e o seu envio para a base de dados foi desenvolvida um rotina em Python que terá essencialmente duas funções. Por um lado, esta rotina atua como um cliente MQTT, que subscreve todos os tópicos no broker, e por outro lado, atua como agente de inserção de dados no InfluxDB.

Para que este script subscreva os dados publicados no broker foi utilizada a biblioteca "paho.mqtt.client". Ao ser executada, é estabelecida uma ligação ao broker e são subscritos todos os tópicos. Para este efeito, foi utilizado o método ".subscribe('#')",

onde o caracter "#" dá a indicação que devem ser subscritos todos os tópicos. Depois, é também criada uma função de "callback", que será chamada sempre que for publicada uma mensagem no broker. Essa função terá duas tarefas. A primeira é descodificar os dados recebidos e transformar num objeto JSON, e a segunda é inseri-los na base de dados.

Conforme explicado anteriormente, cada data logger pode enviar várias mensagens diferentes, nomeadamente dados de sensores, mensagens das ECUs ou parâmetros relevantes das ECUs de um KME, EWI DNA ou FP2. Estas mensagens devem ser tratadas de forma diferente no servidor. Durante o desenvolvimento da primeira parte do trabalho, definiu-se que cada mensagem seria publicada num tópico diferente, do tipo "DL/xxxx". Deste modo, em função do tópico em que a mensagem foi publicada, o script python saberá como tratar a mensagem. Depois de saber o tipo de mensagem, é preciso também cruzar os dados recebidos com os parâmetros de configuração do respetivo ficheiro YAML. Para este efeito, em todas as mensagens é enviado um parâmetro com a identidade do data logger, permitindo ao script chamar o ficheiro de configuração correspondente. Antes de inserir os dados na base de dados há a formatação dos dados recebidos. Este processo inclui essencialmente duas funcionalidades:

- Caso o ficheiro de configuração tenha a informação de que um determinado sensor não está a ser utilizado no data logger, qualquer valor que seja enviado para esse parâmetro será substituído por um código específico.

- No caso das mensagens com parâmetros relevantes da ECU, é apenas enviada a potência em % do valor nominal. Por esse motivo, o ficheiro de configuração é chamado para fornecer a potência nominal do equipamento para que seja calculada a potência em kW.

Depois disso, é criado um dicionário com os vários parâmetros que serão inseridos na base de dados e é chamada a função responsável por enviá-los para a BD. Para implementar estas funções, foi utilizada a biblioteca "InfluxDBClient" desenvolvida para permitir aos scripts python interagir com a base de dados InfluxDB. Nesse dicionário, é selecionado o bucket "Time Series", uma "measurement" com nome igual à identidade do data logger, um conjunto de "fields" onde são atribuídos os dados recebidos e "tags" que têm como objetivo separar os dados por ensaio de campo e equipamento. Para estas tags, é usado o ficheiro de configuração. Todas as mensagens enviadas pelos data loggers têm um timestamp, que é enviado para a BD em conjunto com os dados.

4.2.5 Processamento dos dados

Do ponto de vista do processamento dos dados, o principal objetivo do trabalho é agregar as séries temporais de forma a ser calculado um conjunto de métricas necessárias para traçar os perfis de carga dos equipamentos. No entanto, com base na análise do estado da arte, nomeadamente a figura 2.3, para se obterem resultados precisos é necessário que o processamento inclua algumas etapas de tratamento das séries. As etapas mais relevantes para este trabalho são o preenchimento de dados em falta (é expectável que haja falhas de dados), deteção de outliers (sensores estão muito sujeitos a fatores externos que comprometem as medições) e a fusão de dados (uso de informação de vários data loggers em simultâneo poderá melhorar a fiabilidade dos resultados). Concluiu-se também que

a deteção de outliers através de algoritmos necessita de cuidados especiais, quer seja ao nível dos hiperparâmetros como do tipo de dados que são usados ou se o algoritmo é supervisionado ou não. Por esse motivo, o processamento dos dados foi dividido em duas principais áreas focais. Primeiramente, foi implementado um algoritmo de deteção de outliers que teve como objetivo estudar a deteção de outliers em séries temporais recolhidas por sensores de baixo custo, sendo que foi selecionado um único algoritmo que já tenha sido testado e validado por outros autores. Este estudo permitiu treinar o algoritmo e afiná-lo para o tipo de dados deste trabalho de modo que pudesse ser usado como forma de controlo do pré-processamento que é aplicado na segunda rotina, que consiste no pré-processamento das séries temporais, a respetiva agregação e posterior extrapolação. O pré-processamento inclui o preenchimento de dados em falta e a deteção de outliers. A fusão dos dados de múltiplos data loggers é aplicada, posteriormente, no frontend.

É necessário que o processamento dos dados ocorra autonomamente mesmo que haja alterações de ensaios de campo ou equipamentos. O sistema deve conseguir separar as séries temporais, métricas e outliers consoante as várias "tags". Para isso, foi implementada uma abordagem na qual o sistema é executado para cada data logger e em cada execução procura determinar qual é o ensaio de campo ativo, a partir dos dados já armazenados na base de dados. Depois de determinado o ensaio, pode enviar queries que retornem apenas os dados pretendidos. Será de esperar que cada ensaio de campo realizado tenha uma identidade única, utilizada pelas rotinas de processamento para filtrar os dados.

Um princípio chave aplicado no processamento dos dados é que o sistema agrega autonomamente os dados em intervalos de tempo reduzidos e depois funde-os aos anteriores. Esta forma de implementação tem algumas vantagens, nomeadamente:

- Pouco tempo após um data logger ser instalado será possível começar a visualizar resultados do processamento;
- Caso haja alguma falha no processamento, as perdas não terão tanto impacto, pois o processamento ocorre muitas vezes com poucos dados, em vez de poucas vezes com muitos dados;

O intervalo de tempo selecionado para o processamento dos dados foi horário. Ou seja, as várias rotinas são executadas 1 vez a cada hora, em todas as horas e pedem dados à base de dados sempre com intervalos de tempo horários. Para implementar este agendamento de tarefas foi utilizada a funcionalidade "crontab" na máquina virtual. Devido às relações de dependência entre os vários tipos de dados, as várias rotinas de processamento são executadas com um desfasamento entre elas de uma hora, conforme representado na figura 4.23, onde está representado um esquema que mostra a sequência de eventos pelos quais um determinado "data point", enviado por um data logger, passa. Conforme representado neste mesmo esquema, quando um data logger envia dados para o broker e estes são inseridos na base de dados, dá-se o instante de tempo t_0, que corresponde ao timestamp desse ponto. Ao fim de 2 horas, na próxima vez que a rotina de deteção de outliers for executada este ponto será chamado. Por sua vez, ao fim de 3 horas, da próxima vez que a rotina de cálculo de métricas for executada, este ponto também será chamado. Depois de 4 horas, a próxima vez que a rotina de extrapolações for executada as métricas calculadas com esse ponto também serão chamadas. Desta forma garante-se um atraso de 2 horas no processamento, relativamente à recolha, que também tem como objetivo permitir ao data logger enviar os dados todos, na eventualidade de uma falha de ligação WiFi por um

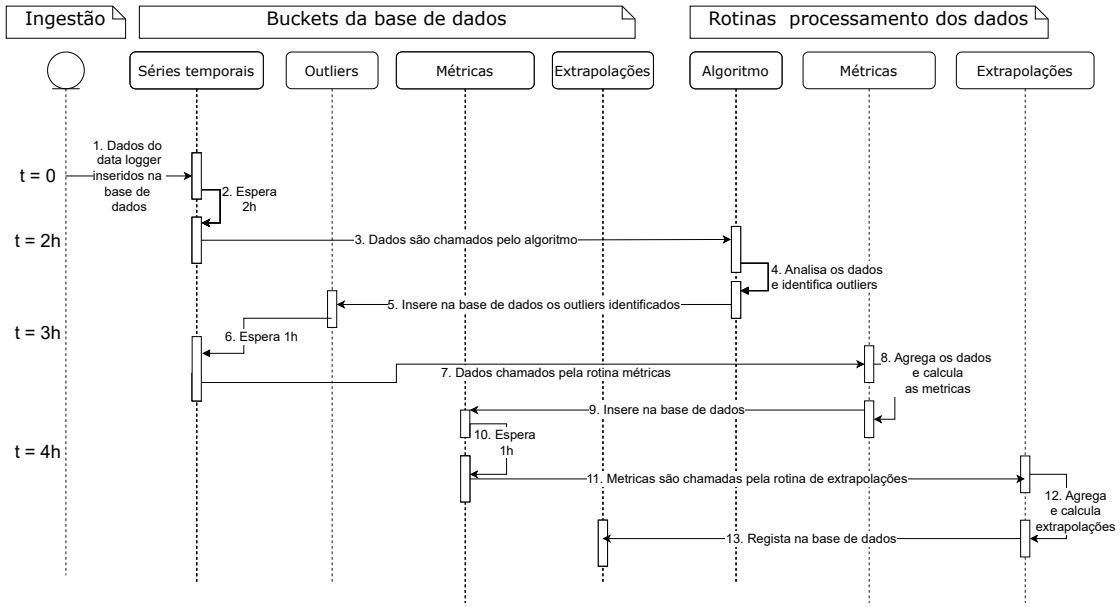

Figura 4.23: Diagrama de atividades pelas quais um data point enviado por um data logger para o servidor passa durante o processamento

período de tempo mais significativo. O processamento começa a ser executado para um determinado ensaio com 2 horas de atraso, sendo que depois, cada uma das 3 rotinas está desfasada uma hora, devido às relações de dependência. De seguida segue-se a explicação detalhada das rotinas de processamento.

Deteção de outliers

A rotina implementada para a deteção de outliers nas séries temporais consiste na seguinte sequência de funções:

1. Ativação da comunicação com a base de dados InfluxDB, através de um token.
2. Envio de uma query à base de dados a pedir a lista dos diferentes data loggers.
3. Seleção de um dos data loggers e leitura do respetivo ficheiro de configuração.
4. Envio de uma query à base de dados a pedir a lista dos ensaios de campo e equipamentos, para o mesmo data logger e considerando um intervalo de tempo entre as três e as duas horas anteriores.
5. Da lista recebida, extração de alguns parâmetros da última linha de dados (ensaio, equipamento, timestamp).
6. Envio de uma nova query à base de dados a pedir dados cujo ensaio é igual ao anterior.
7. Agregação dos dados recebidos num dataframe.
8. Envio de uma nova query à base de dados a pedir os mesmos dados mas com um intervalo de tempo mais reduzido.

9. Agregação dos dados recebidos num dataframe.
10. Definição da lista de variáveis que serão analisadas pelo algoritmo.
11. Para cada variável é treinado um modelo do algoritmo com os dados da query da etapa 6.
12. Para a mesma variável, análise dos dados da query 8 para a deteção de outliers.
13. Registo dos outliers identificados na base de dados.
14. Repetição das etapas 6-8 para as restantes variáveis.
15. Repetição das etapas 3-15 para os restantes data loggers.

Os algoritmos de deteção de outliers dividem-se essencialmente em supervisionados e não supervisionados. O primeiro passo na seleção do algoritmo foi definir a sua tipologia. Com base na análise do estado da arte os algoritmos não supervisionados são mais indicados para aplicações em que se pretende analisar grandes volumes de dados em tempo real, visto que não há necessidade de fazer uma pré-classificação dos dados para treinar o algoritmo. Como nesta aplicação se pretende que o sistema autonomamente identifique outliers praticamente logo a seguir a serem recolhidos os dados, então não seria viável optar por um algoritmo supervisionado. Com base na análise do estado da arte, os algoritmos supervisionados permitem a identifcação de outliers com uma fiabilidade maior, no entanto o nível de exigência desta aplicação não traz a necessidade desse nível de fiabilidade visto que não se pretende identificar propriamente anomalias dos equipamentos, mas sim nos dados recolhidos. Por estes motivos decidiu-se implementar um algoritmo não supervisionado.

Como neste trabalho não se pretende realizar um estudo comparativo entre algoritmos, mas sim avaliar o desempenho de um já bastante aceite, testado e validado para esta aplicação específica, com base nas conclusões retiradas na análise do estado da arte, o algoritmo implementado foi o Isolation Forest. Este é um algoritmo não supervisionado, visto que não é dada a indicação dos pontos normais e dos pontos anormais dentro do dataset fornecido. Com o conjunto de dados que lhe é dado, o algoritmo deverá por conta própria procurar identificar quais são os pontos anormais. A desvantagem deste algoritmo é que é necessário fornecer um hiperparâmetro, designado por contaminação que traduz a percentagem de anomalias previstas dentro do conjunto de dados, pelo que continua a ser necessário ter uma noção da quantidade de outliers esperados nos dados.

A forma como este algoritmo atua baseia-se em gerar uma "isolation forest" que consiste num conjunto de decisões binárias, designadas por "isolation trees". O primeiro passo consiste em selecionar aleatoriamente uma amostra do dataset e definir para essa amostra uma "decision tree". Para a expansão dessa árvore é necessário selecionar um ponto da amostra e um limite. Esse limite é selecionado aleatoriamente entre os valores máximos e mínimos da variável. Se o ponto selecionado estiver acima ou abaixo do limite selecionado então o algoritmo irá expandir para um lado ou para o outro, sendo assim uma decisão binária. Este processo, depois da seleção da amostra, é repetido até que todos os pontos estejam isolados. De seguida é selecionada outra amostra de dados, para construir outra "isolation tree" e é realizado o mesmo procedimento. Depois de construidas várias "isolation trees" haverá uma "isolation forest" e o algoritmo encontra-se treinado. Para o algoritmo determinar se um determinado ponto é uma anomalia, esse

ponto tem de passar por um processo de "scoring". Esse processo consiste em passar esse ponto por todas as "decision trees" geradas e avaliar o grau de profundidade necessário para isolar esse ponto em cada árvore. O princípio aqui presente é que se o ponto for uma anomalia mais rapidamente será isolado e menos ramos terá de ter a árvore. Este princípio encontra-se representado na figura 4.24, retirada do artigo [33].

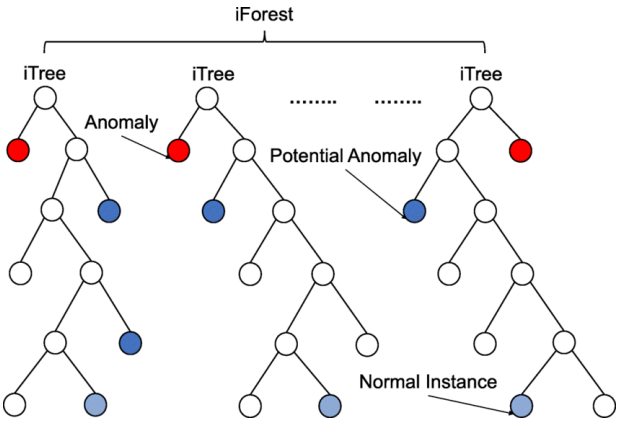

Figura 4.24: Princípio de funcionamento do algoritmo Isolation Forest

Desta forma, fornecendo um conjunto de dados que queremos que sejam analisados ao algoritmo, ele vai correr todos os pontos pela floresta e depois fornecer uma lista de scores para cada um desses pontos. Com base no score de cada ponto e no hiperparâmetro "contaminação", o algoritmo decide se cada ponto é uma anomalia dando um score de -1 ou um ponto normal, tendo um score de 1.

Para implementar o algoritmo de deteção de anomalias na solução proposta foi utilizada a linguagem python e a respetiva biblioteca "scikit-learn". Esta rotina é executada a cada hora. A rotina de deteção deoutliers consiste em utilizar uma grande quantidade dos dados recolhidos, neste caso das últimas 24 horas, para treinar o algoritmo. Depois a cada hora a rotina vai utilizar os dados da última hora para serem analisados pelo algoritmo, de modo a que todos os dados passam uma vez pela análise. Tendo em conta que este algoritmo é não supervisionado este método é adequado, porque são usados dados reais sem dar a informação de quais são normais e quais são anormais para treinar o algoritmo. O parâmetro de contaminação, no entanto, pode ter uma influência muito significativa nos resultados do algoritmo, pelo que é muito importante ter uma noção da percentagem de anomalias esperadas.

O algoritmo foi implementado de forma a analisar três variáveis individualmente, a potência, a temperatura ambiente e pressão hidráulica.

1. No caso da potência, a estratégia consistiu numa redução dimensional, visto que a série temporal "potência" é sempre calculada e está dependente das seguintes variáveis: temperatura de entrada, temperatura de saída e caudal. Deste modo, qualquer outlier provocado por um destes três sensores provocará um outlier na potência. Por este motivo é possível reduzir a carga computacional do algoritmo utilizando apenas esta variável.

2. No caso da temperatura ambiente, o sensor responsável por adquirir este parâmetro é o BME280. Este sensor será colocado no exterior do data logger, por isso estará

suscetível a fatores externos.

3. No caso da pressão hidráulica, durante a implementação percebeu-se que havia um número significativos de outliers nos dados devido a falhas de interação entre o ESP e o sensor.

A afinação do algoritmo foi um processo iterativo que consistiu em variar os vários parâmetros, nomeadamente a contaminação e o número de "isolations trees". Deste modo, chegou-se a uma configuração que permite o melhor desempenho do algoritmo. Para este processo iterativo foram implementadas algumas funções de geração de outliers através do script de inserção de dados na base de dados. Também foram impostas anomalias forçadas nos equipamentos durante a operação. No final deste processo, o algoritmo foi implementado com um parâmetro de contaminação de 0.2 %.

As anomalias assinaldas pelo algoritmo são, posteriormente, enviadas para uma outra zona da base de dados e serão apresentada ao utilizador em conjunto com as séries temporais, no frontend.

Agregação das séries temporais

As métricas têm como objetivo apresentar a um utilizador um conjunto da parâmetros que resume horas/dias de funcionamento de um equipamento. Analisar séries temporais pode ser uma tarefa muito demorada, por isso é importante que o sistema consiga automaticamente resumir todos esses dados em algumas métricas relevantes. Com base na análise do estado da arte, (ver figura 2.3) concluiu-se que para realizar esta tarefa há várias etapas cruciais de tratamento dos dados, como inserir dados em falta, deteção de outliers e, só depois, a agregação das séries temporais. Deste modo, foi desenvolvido um código em Python que executa a seguinte sequência de funções:

1. Ativação da comunicação com a base de dados InfluxDB, através de um token.
2. Envio de uma query à base de dados a pedir a lista dos diferentes data loggers.
3. Seleção de um dos data logger e leitura do respetivo ficheiro de configuração.
4. Envio de uma query à base de dados a pedir a lista dos ensaios e equipamentos, para o mesmo data logger.
5. Para um dos ensaios de campo, determinação do respetivo equipamento, indicação de uso de ECU ou não e timestamp do último dado encontrado.
6. Envio de uma nova query à base de dados a pedir as séries temporais cujo ensaio é igual ao anterior.
7. Agregação dos dados recebidos num dataframe.
8. Preenchimento dos dados em falta no dataframe.
9. Análise dos dados para identificação de outliers.
10. Substituição dos outliers por valores adequados relativamente aos restantes dados
11. Cálculo de um conjunto de métricas

12. Envio das métricas para a base de dados
13. Repetição das etapas 5-12 para os restantes ensaios encontrados.
14. Repetição das etapas 3-13 para os restantes data loggers.

A sequência de funções descrita está representada na figura 4.25.

Para o cálculo de todas as métricas, as séries temporais solicitadas à base de dados foram: temperatura de entrada, temperatura de saída e caudal da água e potência do equipamento. Estas são as séries temporais necessárias para calcular as métricas, mas também foram solicitados outros campos. Estes têm como objetivo filtrar e separar os dados entre ensaios e equipamentos.

Preenchimento dos dados em falta

O cálculo das métricas é efetuado com uma base horária, por isso de cada vez que é executado deveria receber da BD 3600 linhas de dados, um por segundo. Na prática, é recebido um número de linhas que pode variar entre 3400-3600, devido à perda de dados. Este intervalo foi definido com base em resultados de testes realizados durante a implementação da solução. Posto isto, há uma função para preencher todas estas falhas e permitir ter um conjunto de dados completo. A técnica utilizada consiste em determinar as linhas em falta e atribuir valores "NaN". De seguida, a cada valor "NaN" será atribuída a média entre o próximo valor não "NaN" e o anterior.

Algoritmo para identificação de outliers

Depois de completo e sem qualquer valor "NaN", o dataframe passará por um algoritmo de deteção de outliers. Para este caso continuou-se a utilizar o algoritmo Isolation Forest com o mesmo parâmetro de contaminação (0.2%). Todas as séries temporais foram analisadas pelo algoritmo com uma abordagem "univariate data" e os outliers identificados são substituídos por "NaN".

Algoritmo para substituição dos "NaN" por valores adequados

Depois da aplicação do algoritmo de deteção de outliers, o dataset fica com um conjunto de dados em falta. Enquanto que, inicialmente, os dados em falta eram linhas completas de dados, nesta fase os dados em falta são apenas valores únicos no interior de uma linha de dados. Para preencher estes valores foi utilizado um algoritmo de Machine Learning, o "KNNImputer". Para cada valor "NaN", este algoritmo procura as linhas de dados mais semelhantes e calcula a média entre os valores dessas linhas. Essa média corresponderá ao valor "NaN". Para esse efeito, o algoritmo utiliza as variáveis que não têm valor "NaN" para determinar os vizinhos mais próximos. Determinados os vizinhos, utiliza os valores da variável, que tinha valor "NaN" na linha inicial, para calcular a média. Por este motivo, este algoritmo é utilizado em situações cujos conjuntos de dados têm várias variáveis. Na implementação do algoritmo é possível selecionar o número de vizinhos mais próximos que o algoritmo irá procurar, através de um hiperparâmetro. Neste caso foram selecionados dois. Na figura 4.26 está representado o princípio de funcionamento deste algoritmo. Na figura 4.26 há três conjuntos de dados (A, B e C) e um valor da coluna B está em falta. Nessa linha, os valores de A e C são 1 e 1050, respetivamente. Partindo destes valores, o algoritmo procura os vizinhos mais próximos. Neste caso, há três vizinhos que também têm o valor 1 na coluna A. Desses três, um

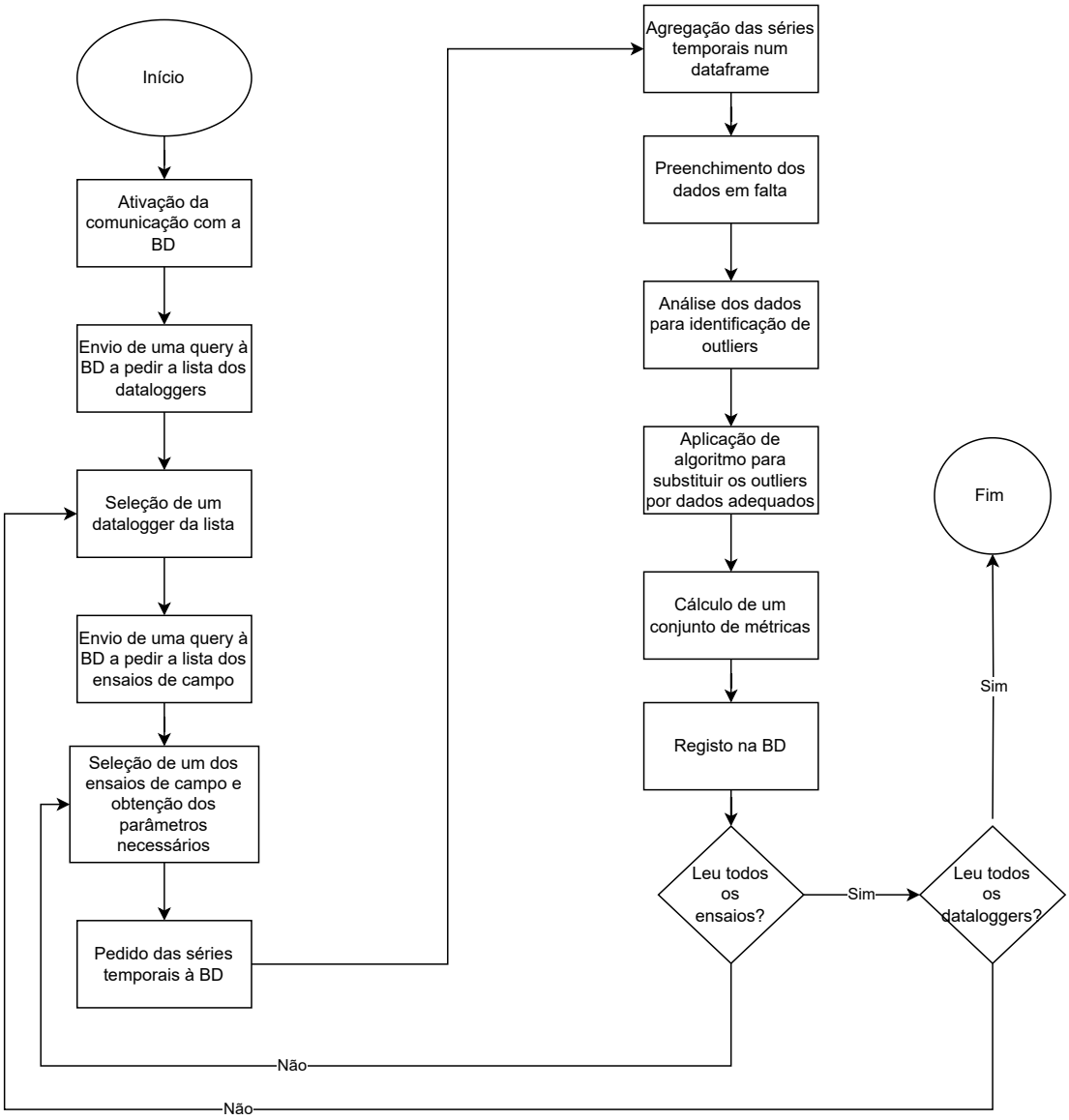

Figura 4.25: Fluxograma da sequência de funções executadas pelo script "metricas.py" desenvolvido

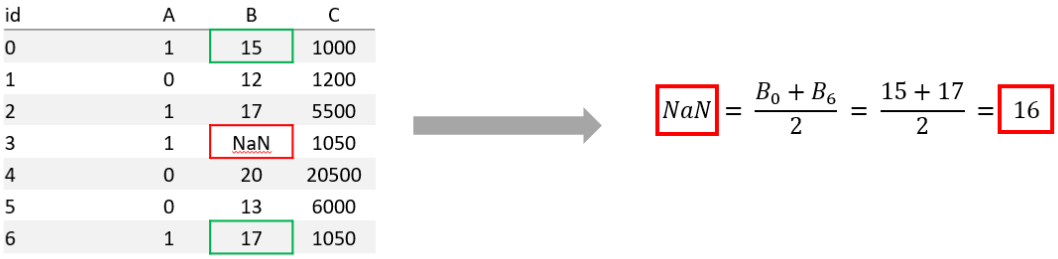

Figura 4.26: Exemplo de funcionamento do algoritmo KNNImputer

deles tem o valor de C bastante afastado de 1050. Por esse motivo, as linhas 0 e 6 são os vizinhos mais próximos. De seguida, é calculada a média entre os valores da coluna B dessas linhas. Essa média é atribuída ao valor "NaN".

Seguem-se algumas notas sobre a abordagem tomada.

1. Para efetuar o pré-processamento dos dados, decidiu-se utilizar uma técnica baseada na média (entre os valores seguintes e anteriores) para completar as linhas de dados em falta. Além disso, utilizou-se um algoritmo de machine learning para atribuir valores adequados aos outliers. Como o algoritmo KNNImputer utiliza um critério de semelhança entre os dados existentes da linha que tem valores "NaN" e as outras linhas, não se considerou o melhor método para inserir linhas completas de dados. Por outro lado, a técnica que utiliza a média entre o valor seguinte e o anterior tem uma limitação: quando os valores "NaN" aparecem no final ou início do conjunto de dados, visto que não há valores seguintes ou anteriores, respetivamente. Para garantir que, antes do cálculo das métricas, o conjunto de dados não tem qualquer valor "NaN" decidiu-se aplicar o algoritmo KNNImputer por último.

2. Alguns estudos [34,35] suportaram a seleção do algoritmo KNNImputer. *Seu et al.* demonstram a superioridade deste algoritmo relativamente a outros [34]. *Mahboob et al.* provam a eficácia do algoritmo KNNImputer no preenchimento de dados em falta sem qualquer erro [35].

3. A abordagem "univariate" para ientificação de outliers é mais pesada para o sistema, porém testes realizados permitiram obter uma maior eficácia na deteção dos outliers.

Cálculo de métricas

Depois de efetuado o tratamento do conjunto de dados são calculadas as métricas, as quais são:

- **Número de arranques:** calculado contabilizando o número de vezes que o caudal passa de zero para um valor positivo, maior do que um.

- **Tempo ON/OFF:** calculado contabilizando o número de linhas de dados com caudal nulo e o número com caudal não nulo.

- **Energia consumida:** calculada efetuando a média entre todas as medições da potência. Como o intervalo de tempo considerado é de 1 h, então a média da potência num intervalo de tempo de 1h [kW], é igual à energia consumida [kWh].

- **Distribuições de temperatura, caudal e potência:** com os valores de cada um destes parâmetros é calculado o tempo (absoluto e em rácio) que o equipamento esteve em determinadas condições de operação. Ou seja, quantos dados têm temperatura entre 15ºC e 25ºC , potência entre 20 e 25 kW, etc.

Estas métricas são depois inseridas na base de dados, incluindo o nome do ensaio e equipamento, e o timestamp da última linha de dados considerada.

Extrapolações

As métricas anteriores permitem realizar uma descrição bastante detalhada do perfil de carga do equipamento. Estes parâmetros, no entanto serão influenciados pela duração do ensaio de campo. Naturalmente, um data logger com um mês de recolha de dados, não terá um perfil diretamente comparável com um de seis meses. Para criar uma uniformização das métricas calculadas realizou-se um cálculo de extrapolação das métricas calculadas com frequência horária para um intervalo de tempo de um ano. Para isso foi desenvolvido um script em python com o nome "extrapolation.py", cuja sequência de tarefas é a seguinte:

1. Ativação da comunicação com a base de dados InfluxDB, através de um token.

2. Envio de uma query à base de dados a pedir a lista dos diferentes data loggers.

3. Seleção de um dos data logger e leitura do respetivo ficheiro de configuração.

4. Envio de uma query à base de dados a pedir a lista dos ensaios de campo e equipamentos, para o mesmo data logger e considerando um intervalo de tempo entre as cinco e as quatro horas anteriores.

5. Da lista recebida é retirado o último ensaio, equipamento e timestamp.

6. Para as condições registadas na etapa anterior é enviada uma query à base de dados a pedir a lista de todas as métricas já calculadas desde sempre para essas condições. Neste caso, é solicitado o nome das métricas e não o respetivo valor.

7. Seleção de uma das métricas da lista recebida e envio de uma nova query à base de dados a pedir todos os valores registados na base de dados para essa métrica e para as condições registas na etapa 5.

8. Para a lista de valores recebidos é efetuada a extrapolação para um período de um ano.

9. Repetição das etapas 7 e 8 para as restantes métricas.

10. Agregação de todas as extrapolações numa lista e envio para a base de dados, para um bucket com o nome "Extrapolações".

Esta rotina será executada a cada hora, no entanto considera todas as métricas já calculadas desde sempre. Esta abordagem permite que pouco tempo após o data logger ser instalado seja possível visualizar no frontend resultados das extrapolações. Todavia, à medida que o tempo de operação cresce o resultado das extrapolações torna-se cada vez mais fiável, visto que considerará cada vez mais dados.

4.3 Frontend

O Frontend deste trabalho traduz o ponto final do sistema implementado. Interage com o utilizador apresentando dados em dashboards e foi implementado através da ferramenta Grafana. Para a construção do frontend houve duas ideias-chave: os dados devem ser separados consoante a sua função e objetivo, e o utilizador deve poder interagir com as dashboards selecionando variáveis que irão, dinâmicamente, influenciar a dashboard. Com base nestes dois princípios foram construídas algumas dashboards:

1. Página inicial - tem como objetivo apresentar ao utilizador um ponto inicial para exploração dos dados. Apresenta um conjunto de alertas de data loggers que deveriam estar ativos mas não estão e tabelas com informação dos data loggers que estão no terreno, como por exemplo data de início e fim de ensaio, equipamento no qual foi aplicado e estado atual. Nesta página são também apresentadas tabelas com as listas de métricas e séries temporais que o sistema recolhe e processa.

2. Séries temporais - esta dashboard tem como objetivo permitir ao utilizador selecionar um determinado ensaio de campo e avaliar gráficos das séries temporais que o datalogger está a enviar. Nesta dashboard é possível visualizar as variações dos vários parâmetros ao longo do tempo, identificar falhas e perdas de dados e identificar padrões na variação das variáveis em qualquer uma das séries temporais pretendidas. Nesta dashboard são ainda apresentadas, para cada série temporal, o respetivo valor máximo, mínimo e médio. Num dos inputs da dashboard é possível selecionar a resolução temporal com que os dados serão apresentados. A resolução máxima é de 1 segundo, no entanto, esta só deverá ser usada para intervalos de tempo curtos. Caso contrário, a quantidade de dados apresentados é muito grande e as queries tornam-se lentas.

3. Mensagens UART - tal como apresentado anteriormente, os data loggers enviam as mensagens recebidas pela interface UART dos equipamentos aos quais estão ligados. Esta dashboard apresenta uma tabela com os vários parâmetros das mensagens separados em colunas, para um ensaio à escolha do utilizador. Estes parâmetros são apresentados em "raw", sem qualquer descodificação, visto que o objetivo desta dashboard é atuar como "backup" para situações particulares.

4. Outliers - esta dashboard apresenta ao utilizador os outliers identificados pelo algoritmo para um determinado ensaio de campo. Estes outliers serão apresentados na forma de tabelas e gráficos. Nesta dashboard é possível ainda visualizar a quantidade de pontos registados num intervalo de tempo a definir pelo utilizador, e a percentagem desses pontos que foi identificada como outlier.

5. Métricas - esta dashboard apresenta a totalidade das métricas calculadas a partir das séries temporais para um determinado ensaio de campo. Nas métricas incluém-

se distribuições de potência, temperaturas e caudal, número de arranques, energia consumida e tempo de operação.

6. Extrapolações - esta dashboard apresenta os resultados da extrapolação das métricas anteriores para o período de um ano. Nesta dashboard haverá um conjunto de tabelas onde o utilizador poderá selecionar um ou vários ensaios de campo.

7. Perfis de carga - esta dashboard é o culminar de todo o trabalho apresentado até agora. Com base na análise do estado da arte, nomeadamente a figura 2.3 além do processamento individual para cada data logger, é importante fazer a fusão dos dados provenientes das várias fontes. Partindo desse princípio, esta dashboard tem objetivo permitir a um utilizador selecionar um determinado equipamento e visualizar resultados médios de todos os ensaios de campo já registados com esse equipamento. Esta dashboard utiliza as extrapolações e as métricas relativas para que os valores médios das métricas sejam calculados com base em valores uniformizados.

A dashboard relativa aos perfis de carga está representada na figura 4.27. Na dashboard

Figura 4.27: Dashboard com os perfis de carga, onde o utilizador poderá selecionar o tipo de equipamento do qual deseja visualizar os resultados e aplicar filtros aos ensaios de campo

apresentada na figura 4.27 o utilizador pode selecionar (no canto superior esquerdo) o equipamento do qual pretende visualizar métricas. Ao selecionar um equipamento, a dashboard atualiza para apresentar resultados médios de todos os ensaios de campo cujo equipamento foi igual ao selecionado. O utilizador pode ainda filtrar os ensaios (campos ao lado do canto superior esquerdo), de forma que sejam incluidos apenas os ensaios que desejar. Do lado esquerdo da dashboard são apresentadas métricas individuais como: número de arranques, tempo ligado e desligado, energia consumida e condições atmosféricas médias. Além disto, são apresentadas as distribuições intervaladas das temperaturas de entrada e saída da água, caudal e potência. Estas distribuições são apresentadas

em rácios (gráficos circulares) e em horas (diagramas de colunas). Através dos campos de entrada na zona superior da dashboard, o utilizador pode ainda variar o passo das distribuições intervaladas. Por exemplo, para a temperatura de entrada da água, pode selecionar um passo de 2, 5 ou 10 ^{o}C. As métricas apresentadas na dashboard da figura 4.27 são valores médios das extrapolações efetuadas para cada ensaio. No apêndice A podem ser consultadas as restantes dashboards que fazem parte do frontend construído.

Capítulo 5

Análise dos resultados

Neste capítulo serão apresentados os resultados obtidos e a respetiva análise. Esta análise foi dividida em alguns tópicos, mais relevantes para avaliar o desempenho do data logger desenvolvido.

1. **Instalação e interação local com o data logger:** foram realizados um conjunto de testes para avaliar a forma como um instalador poderá interagir com o data logger, nomeadamente o funcionamento da ligação em modo AP (Ponto de acesso WiFi), interpretação dos sinais fornecidos pelos LEDs, visualização de mensagens na interface WEB Serial, etc.

2. **Visualização das séries temporais:** foram realizados testes para demonstrar a visualização das séries temporais em tempo real e a coerência entre elas.

3. **Precisão dos dados:** foram realizados testes de calibração a alguns sensores para determinar o erro relativo dos dados recolhidos.

4. **Perdas de dados:** foram realizados testes com o objetivo de verificar se o data logger é capaz de funcionar durante longos períodos de tempo sem perder muitos dados. Este ponto é crucial para avaliar o desempenho do data logger.

5. **Identificação de outliers:** foram realizados testes para avaliar o desempenho e a eficácia do algoritmo, visto que uma das etapas fundamentais do processamento dos dados foi a aplicação de um algoritmo de deteção de outliers.

6. **Precisão das métricas e extrapolações:** foram realizados testes para verificar se os resultados da agregação das séries temporais estavam coerentes com as séries temporais e com as condições de teste. Este ponto é crucial para que haja confiança nos resultados fornecidos pelo sistema implementado.

Para analisar todos estes tópicos, o protótipo desenvolvido foi colocado num laboratório da empresa a recolher dados de esquentadores em funcionamento. Para cobrir o máximo de possibilidades, o data logger foi implementado de forma a recolher dados de dois equipamentos distintos em simultâneo. Por um lado foi estabelecida uma comunicação com a ECU de um esquentador e neste caso, foi apenas utilizada a ECU para recolha de dados. Por outro lado, foram colocados sensores de temperatura à entrada e saída da água e um caudalímetro num outro esquentador. Nas figuras 5.1 e 5.2 estão apresentadas fotografias da implementação do protótipo no laboratório.

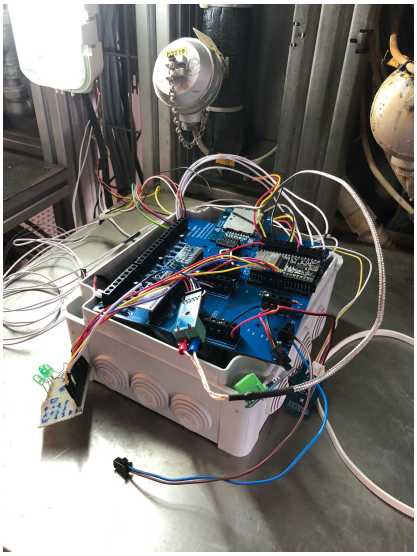

Figura 5.1: Protótipo durante a fase de testes

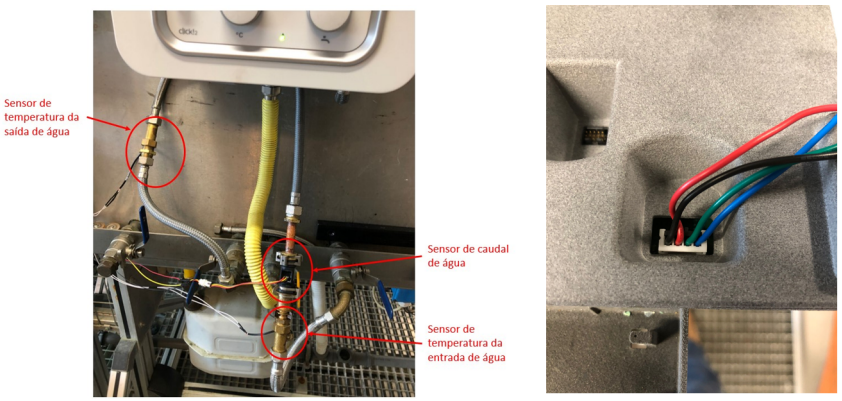

Figura 5.2: Recolha de dados de um esquentador utilizando sensores de temperatura à entrada e saída da água e um sensor de caudal (à esquerda) e recolha de dados de um esquentador através da comunicação com a ECU (à direita)

5.1 Instalação e interação local com o data logger

Na instalação de um data logger numa nova localização, a principal questão é a forma como lhe são fornecidas as credenciais para estabelecer uma conexão WiFi com o router. Para isto, o ESP liga-se em modo AP e o instalador tem de se conectar à rede do ESP através do telemóvel. Depois terá de aceder ao browser, inserir o IP "192.168.4.1" e preencher o formulário, que está representado na figura 5.3 (à esquerda). Realizaram-se testes para avaliar a rapidez e robustez deste procedimento. Quando o ESP é ligado à corrente, se ao fim de 20 segundos não se conseguir ligar à rede WiFi, então ativa o modo AP. Neste momento, o instalador poderá conectar-se ao ESP. Esta tentativa de conexão foi testada múltiplas vezes, retirando-se as seguintes conslusões:

1. Para que um telemóvel se possa conectar à rede do ESP terá de se encontrar a uma distância bastante curta, sendo que para distâncias superiores a 1 metro, a conexão

nunca é estabelecida. No entanto, esta limitação deveu-se ao facto de que o ESP32 utilizado não continha uma antena PCB e por isso o alcance era bastante reduzido.

2. Obstáculos entre o ESP e o telemóvel também podem dificultar a ligação.

3. Se o telemóvel estiver bastante próximo do ESP, a ligação é quase sempre estabelecida. Por vezes, demore apenas alguns segundos, mas também pode demorar 30 segundos a 1 minuto.

4. O ESP foi programado para que se ao fim de 20 segundos em modo AP nenhum equipamento se tiver conectado, então irá reinicar. No entanto, caso algum telemóvel se tente conectar ao ESP, mesmo que a ligação demora a ser estabelecida, desde que o telemóvel continue a tentar estabelecer a conexão, o ESP não irá reiniciar.

5. Por vezes podem ser necessárias algumas tentativas para que seja estabelecida a conexão entre o telemóvel e o ESP.

Estabelecida a ligação ao ESP, o IP que o utilizador deverá colocar no browser é sempre o mesmo. Os testes revelaram que se a ligação for estabelecida, o envio das credenciais WiFi por HTTP para o ESP funciona com um bom desempenho e a resposta do ESP aos pedidos HTTP é quase imediata. Durante este processo de configuração, o instalador deve manter o telemóvel próximo do ESP, caso contrário a ligação será perdida e o ESP irá reiniciar de imediato.

Além da interação com o data logger ao nível do modo AP, também é importante que tanto o instalador como posteriormente o cliente, consigam receber feedback do data logger simplesmente através dos LEDs. Para isso, foram implementados múltiplos sinais que o ESP irá fornecer através de quatro LEDs.

1. ESP em modo AP - o primeiro e segundo LEDs deverão estar acessos, indicando, respetivamente, alimentação e ligação em modo AP.

2. Operação normal do data logger - o primeiro e terceiro LEDs estão acesos, dando indicação de alimentação e WiFi, respetivamente. O quarto LED pisca de segundo a segundo (indicando recolha de dados) e o segundo pisca muitas vezes a cada 10 segundos (indicando envio de dados ao servidor).

3. Data logger em modo de funcionamento normal e a ligação WiFi está OFF - o primeiro LED está aceso, porém o terceiro (Led do estado WiFi) está desligado. O quarto LED mantém-se a piscar sempre que houver um ciclo de recolha de dados.

4. Envio de mensagens para o servidor - sempre que uma mensagem é enviada com sucesso o LED na segunda posição pisca.

5. Recolha de dados e registo no cartão SD - sempre que o ciclo de recolha de dados funcionar corretamente o LED na quarta posição pisca.

6. Caso haja uma avaria no cartão SD, o LED na quarta posição pisca duas vezes a cada segundo.

7. Quando há uma falha no envio de uma mensagem ao servidor, todos os LEDs piscam simultaneamente.

Os testes realizados revelaram que o funcionamento dos LEDs cumpriu sempre com as expetativas e permitem que o instalador saiba exatamente o estado de funcionamento do data logger sem precisar de qualquer equipamento. Alguns screenshots do estado dos LEDs durante alguns dos estados de funcionamento anteriores podem ser consultadas no apêndice B.

Contudo, há informações para as quais os LEDs não podem fornecer feedback. Um exemplo disso é os valores das leituras dos sensores. No momento da instalação é importante verificar se os valores recolhidos pelos sensores são consistentes com a realidade. Para isso, foi implementada uma forma de debug em que o instalador recebe mensagens, através do browser do telemóvel. Na figura 5.3 (à direita) está representada a interface que o instalador poderá ver e uma mensagem enviada pelo ESP durante os testes realizados. Na figura 5.3 é possível visualizar a temperatura de entrada e saída da água, bem

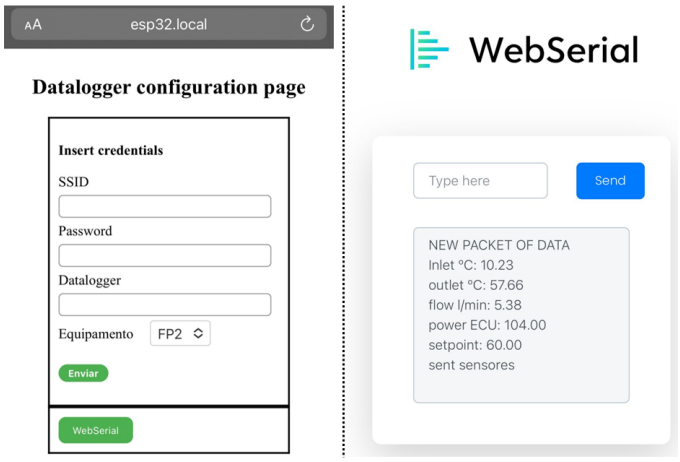

Figura 5.3: Interface Web que será apresentada ao instalador localmente no browser do telemóvel incluindo a página de configuração (à esquerda) e a página de monitorização (à direita)

como o caudal que o ESP recolheu a partir dos sensores. O instalador pode comparar estes valores com os valores expectáveis para avaliar se o data logger está a funcionar corretamente.

5.2 Visualização das séries temporais

O frontend construído inclui uma dashboard de visualização das séries temporais. Na figura 5.4 está apresentada esta dashboard para um teste em que apenas foram retirados dados a partir dos sensores externos. Na dashboard da figura 5.4 o utilizador pode escolher os data loggers e as aplicações, sendo que estes campos atuam como filtros para o campo que de facto tem impacto na dashboard, o ensaio de campo. Neste caso, foi selecionado o ensaio "Delta_testOF". O campo de entrada "Time Resolution" permite definir a janela de agregação dos dados. Como para intervalos de tempo maiores, a quantidade de dados é significativa, então este campo permite reduzir a quantidade de dados apresentados calculando a média de um conjunto de pontos de duração igual ao que o utilizador preencher neste campo.

Figura 5.4: Dashboard com séries temporais recolhidas a partir dos sensores externos

Na primeira linha de gráficos da figura 5.4 estão representadas as principais variáveis relativas à operação do equipamento (temperaturas de entrada e saída, caudal e potência). Conforme é possível verificar através da variação da potência o equipamento realizou vários ciclos ON/OFF, em que a cada hora ligava ou desligava. É possível verificar a coêrencia dos dados visto que quando o caudal é zero, a potência também é zero e a temperatura de saída é mais baixa. Para os intervalos de tempo em que o equipamento esteve ligado, o caudal variou entre os 4-6 L/min e a potência entre os 15-20 kW. A temperatura de entrada é mais alta para os casos em que o equipamento está ligado, pois nestes casos fornece a temperatura da água da rede. Nos gráficos também é possível acompanhar as condições atmosféricas do ar ambiente, nomeadamente a temperatura, pressão e humidade. Neste caso, a temperatura média foi de 20.3 ºC, humidade média de 53.8 % e pressão 1.06 bar. Algumas subidas inesperadas nos gráficos apresentados representam outliers, causados por falhas na comunicação com os sensores. Esta dashboard permite ainda visualizar valores máximos e mínimos para cada série temporal. Esta dashboard atua como um suporte que permite visualizar o panorama geral de operação de um equipamento, a evolução ao longo do tempo, investigar algum momento ou variável específica e avaliar se o data logger está a funcionar corretamente.

Foi também realizado um teste em condições controladas para avaliar a precisão das séries temporais. Neste teste, o caudal e potência de operação estiverem sempre dentro dos limites 3.7-3.9 L/min e 20.8 - 21.8 kW, respetivamente. O aparelho foi ligado e desligado durante alguns intervalos de tempo, conforme as condições apresentadas na tabela 5.1.

Face às condições descritas na tabela 5.1 os resultados das séries temporais associadas à temperatura de entrada, saída, caudal e potência estão apresentadas na figura 5.5.

Na figura 5.5 é possível visualizar os vários ciclos de operação, nomeadamente os intervalos em que o equipamento foi ligado e desligado. Nos períodos em que esteve desligado a potência (gráfico no canto inferior direito da figura 5.5) e o caudal (gráfico no canto inferior esquerdo da figura 5.5) são nulos, enquanto quando está ligado os valores encontram-se dentro dos intervalos definidos anteriormente. No caso da temperatura de entrada (gráfico no canto superior esquerdo) é possível verificar que, quando o equipamento está ligado, a temperatura de entrada é mais baixa, correspondendo à temperatura

Tabela 5.1: Condições de teste para efeitos de validação das séries temporais

Horário	Estado do equipamento
10h - 10h30	ON
10h30 - 11h	OFF
11h - 11h45	ON
11h45 - 12h45	OFF
12h45 - 13h35	ON
13h35 - 13h59	OFF
13h59	ON

Figura 5.5: Resultados das séries temporais para as condições de testes descritas na tabela 5.1

da água da rede. Quando o equipamento é desligado, a temperatura de entrada começa a subir, porque começa a tender para a temperatura do ar ambiente. A temperatura de saída da água (gráfico no canto superior direito) tem um valor praticamente constante, enquanto o equipamento está ligado. Este resultado está de acordo com o esperado, visto que a potência, o caudal e a temperatura de entrada são praticamente constantes (enquanto o equipamento está em funcionamento). Quando o aparelho é desligado, como a água fica no interior das tubagens que passam pela câmara de combustão e o sensor de temperatura está bastante próximo da câmara, então a água que fica no interior desses tubos começa a aquecer durante alguns minutos. Entretanto, à medida que dissipa calor para o ar ambiente, a câmara de combustão começa a arrefecer e a temperatura de saída também começa a diminuir. Os resultados obtidos nas séries temporais estão de acordo com o teste que foi realizado.

5.3 Análise da precisão dos dados recolhidos

O data logger recolhe essencialmente dois tipos de dados: leituras de sensores externos e parâmetros enviados pela ECU. No caso dos sensores externos é importante garantir uma

boa calibração dos sensores, nomeadamente os cálculos efetuados pelo data logger para converter os outputs dos sensores nas grandezas pretendidas. Por esse motivo, foram realizados testes de calibração para validar as medições realizadas pelo data logger.

Foram realizados testes individuais a alguns dos sensores mais suscetíveis a problemas na calibração, nomeadamente os sensores de temperatura NTC, o sensor de pressão hidráulica, o sensor de corrente e o caudalímetro. Nas próximas subsecções será apresentado o procedimento experimental e os respetivos resultados.

Sensores de temperatura NTC

No caso dos sensores de temperatura NTC foram realizados testes utilizando a conversão A/D (analógica digital) do próprio ESP e das placas ADS1115 para avaliar qual teria melhores resultados. Estes testes consistiram em colocar os sensores no interior de um forno com temperatura regulada, juntamente com termopares de um picologger, que é um dispositivo usado para a calibração de sensores, fornecendo os valores de referência a partir de sensores de temperatura calibrados. Na figura 5.6 está representada a visualização dos dados recolhidos pelo picologger e pelo ESP durante o teste realizado. O

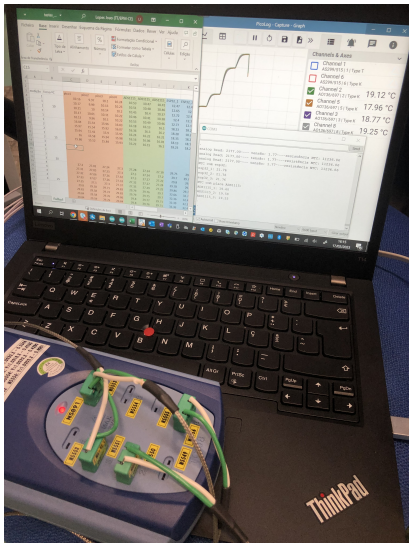

Figura 5.6: Visualização dos dados recolhidos durante o teste experimental dos sensores NTC

procedimento consistiu no registo de cinco valores de temperatura medidos diretamente pela conversão A/D do ESP32 e outros cinco valores a partir das placas ADS1115. Foi definido um intervalo de medição de 10ºC até 80ºC com um passo de 5ºC. Estes valores foram definidos com base nas temperaturas esperadas para a entrada e saída da água num esquentador. Os resultados deste teste encontram-se resumidos na figura 5.7, que apresenta a temperatura medida pelos sensores em relação à temperatura do pico logger. Para avaliar o erro das medições é necessário avaliar o declive da reta de regressão linear, sendo que quanto mais próxima de 1 menor é o erro das medições. Conforme representado na figura 5.7 os resultados mostram que a conversão analógica das placas ADS1115 permite ter dados muito mais precisos que a conversão A/D do ESP. Considerando os pontos apresentados na figura e o declive das respetivas retas de regressão linear obtidas,

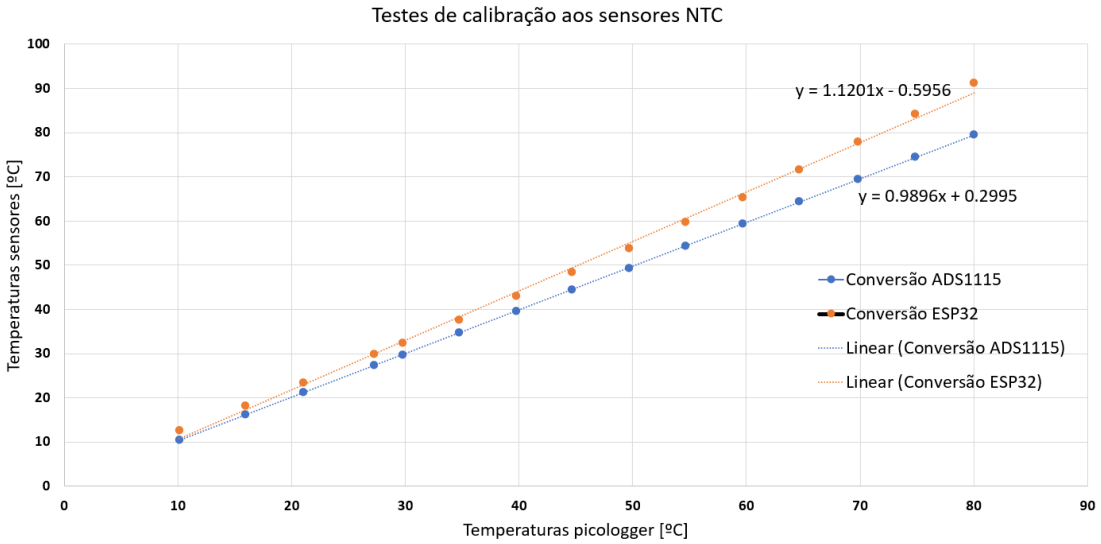

Figura 5.7: Resultados obtidos para o teste experimental dos sensores NTC

o erro obtido com as placas ADS1115, relativamente aos valores de referência recolhidos a partir do pico logger, é de cerca de 1%, enquanto que utilizando apenas o ESP o erro é de 12%. Esta foi a principal razão que motivou a utilização das placas externas para realizar a leitura dos sensores de temperatura. Face aos requisitos deste trabalho, um erro de cerca de 1% é aceitável.

Caudalímetro

Não foram realizados testes individuais ao caudalímetro, porém a verificação da precisão dos valores adquiridos foi realizada durante os testes finais efetuados ao data logger. Foi comparado o valor adquirido pelo data logger com o valor de referência fornecido pela banca do laboratório. Os resultados revelaram que o erro de medição foi cerca de 0.2 L/min. Os valores obtidos foram: 4.00 vs 4.19 L/min e 4.50 vs 4.73 L/min (valor de referência vs valor do data logger). Para os requisitos deste trabalho, este erro de medição é perfeitamente aceitável.

Sensor de pressão hidráulica

O sensor de pressão hidráulica foi alvo de testes individuais, numa banca de testes de circuitos hidráulicos. Nestes testes, houve dois tipos de medição: pressão estática e pressão dinâmica. Implementado o sensor de pressão hidráulica no circuito, fez-se variar a pressão e comparou-se o valor medido pelo sensor com o valor de referência da banca. Os resultados encontram-se resumidos na figura 5.8, que representa a pressão hidráulica medida pelo sensor em relação à de referência. Conforme mostram os resultados da figura 5.8, os pontos medidos estão praticamente coincidentes com a reta $Y = X$, o que significa que praticamente não há desvio entre o valor de referência e o valor medido pelo sensor. O desvio entre os pontos e a reta foi sempre inferior a 2%.

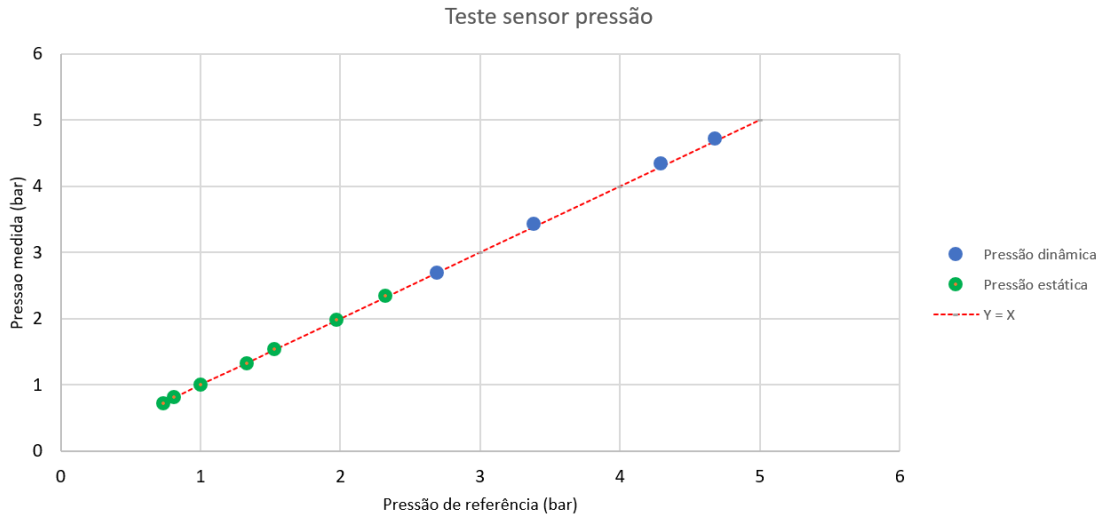

Figura 5.8: Resultados obtidos para o teste do sensor de pressão hidráulica

Sensor de corrente

O sensor de corrente foi alvo de testes individuais. O objetivo destes testes foi encontrar a melhor forma de implementar a leitura da corrente e também avaliar o grau de precisão das medições com a solução implementada. Tal como foi apresentado no capítulo anterior o sensor de corrente tem um sinal de saída analógico, o qual deverá ser convertido num sinal digital pelo ESP. Uma das fases do teste realizado ao sensor de corrente foi avaliar qual a melhor solução: utilizar o ESP para a conversão A/D ou as placas ADS1115. O tempo que as placas ADS1115 demoraram a realizar cada medição foi demasiado longo para que se conseguissem obter os valores de pico de corrente. Por esse motivo, a utilização destas placas para medir corrente AC é inviável. Para medir corrente alternada, é crucial realizar o máximo de medições possível num curto período de tempo. O teste realizado revelou uma diferença demasiado significativa entre a rapidez de conversão A/D do ESP e das placas ADS1115. Conseguiram-se realizar mais de 300 medições em 20 ms com o ESP e 2 ou 3 medições em 20 ms com as placas externas.

Os testes aos sensores de temperatura revelaram que a conversão A/D do ESP está sujeita a baixa precisão e exatidão. Por esse motivo, foi importante avaliar se a solução implementada conseguiria fornecer dados consistentes o suficiente para cumprir com os requisitos deste trabalho. Para isso, foi realizado um teste experimental, no qual foram utilizados equipamentos com alto consumo de corrente. Desse modo, conseguiu-se comparar o valor medido com o sensor com o valor fornecido por um amperímetro. Na figura 5.9 está representada a montagem experimental utilizada. A montagem consistiu em colocar o amperímetro e o sensor em série no circuito de alimentação do equipamento (no caso da figura 5.9 um secador industrial). Deste modo, conseguiu-se medir a corrente que passava pelo sensor e pelo amperímetro. Para este teste, foram utilizados secadores industriais e aquecedores. Os aquecedores permitiram ter um consumo de corrente de 1.92 A constante e os secadores corrente máxima entre 7 e 8 A. Desta forma, foi possível obter quatro valores de referência que cubriam toda a gama de medição que se pretende neste trabalho. Estes valores de referência foram obtidos durante o funcionamento dos equipamentos através de um amperímetro. A recolha de resultados deste teste incluiu dez

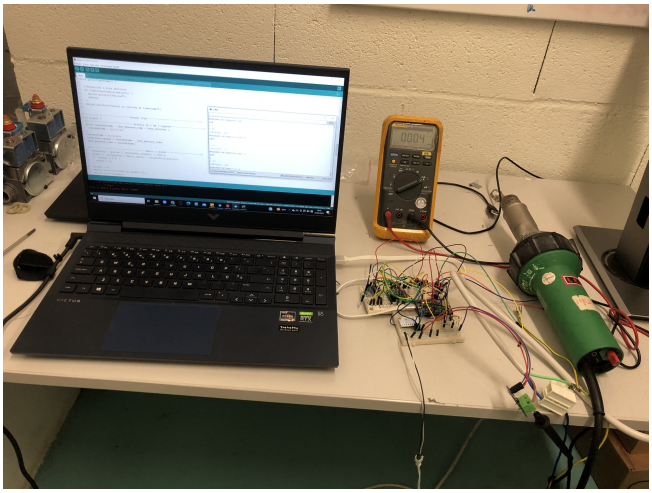

Figura 5.9: Montagem experimental utilizada para testar o sensor de corrente

medições, as quais foram utilizadas para estabelecer um valor médio que foi comparado com o valor de referência. Como era de esperar a variação entre as várias medições é significativa, dada a instabilidade da conversão A/D do ESP. Posto isto, os resultados estão apresentados na tabela 5.2. Conforme mostram os resultados da tabela 5.2, em termos

Tabela 5.2: Resultados obtidos no teste experimental ao sensor de corrente

	Referência [A]	Média Sensor [A]	Desvio máximo [A]
Aquecedores	1.92	1.89	0.25
Secador A	6.82	6.85	0.18
Secador B	8.12	8.16	0.34
Secadores A e B	14.94	14.81	0.38

médios o valor medido pelo sensor está próximo do valor real, com um erro inferior a 0.2 A. No entanto, se for analisada cada medição individualmente os desvios podem ser na ordem dos 0.4 A. Mesmo assim considerou-se que este erro de medição é aceitável para as exigências desta aplicação.

5.4 Perdas de dados

Uma das formas utilizada para avaliar o desempenho do data logger foi a quantidade de dados perdidos, quer seja durante o funcionamento normal ou em eventuais falhas de WiFi. Para que não haja perdas de dados também é importante que o data logger consiga realizar cada ciclo de aquisição e registo no cartão SD rapidamente, para que nunca seja ultrapassado o tempo de 1 segundo da taxa de aquisição. No caso da rapidez de recolha de dados os testes realizados revelaram que o ESP demora cerca de 400 ms por cada ciclo, considerando que regista informação de todos os sensores e da ECU. Deste modo, no cenário mais exigente possível o ESP demora, em média, menos de metade do tempo que tem para recolher todos os parâmetros. Por vezes, há ciclos que necessitam de um pouco mais tempo (cerca de 550-600 ms), contudo, a margem é suficiente para garantir bom desempenho em eventuais atrasos.

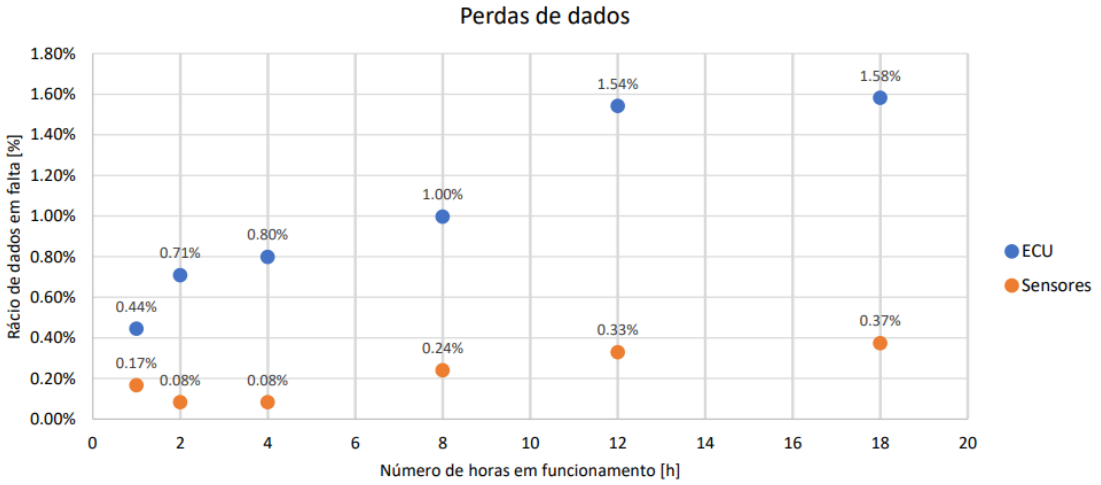

Figura 5.10: Evolução do rácio de dados perdidos ao longo de horas de funcionamento

Para verificar a quantidade de falhas nas séries temporais enviadas para o servidor, foi implementado um contador nos gráficos do Grafana para apresentar também a quantidade de dados que foram recolhidos. Posto isto, foram realizados testes com o data logger em funcionamento normal durante algumas horas, sendo que o caso ideal seria recolher 3600 dados por hora. Na figura 5.10 são apresentados os resultados obtidos ao longo de várias horas de funcionamento. A figura apresenta, para os sensores e para a ECU, o rácio de dados em falta ao longo de vários intervalos de tempo. Conforme representado na figura 5.10 há uma diferenca considerável entre os dados dos sensores e da ECU. No caso dos sensores, o rácio de dados perdidos mantém-se inferior a 0.5 % mesmo ao fim de 18 horas de funcionamento. Não se nota uma subida relevante neste rácio ao longo do tempo, o que é um bom resultado e indica que o data logger não terá um aumento de problemas de desempenho ao longo do tempo. Por outro lado, no caso dos parâmetros da ECU, o rácio de dados perdidos é cerca de 1.6 % ao fim de 18 horas. Um rácio de 1.6 % traduz-se em cerca de 60 linhas de dados em falta por cada hora de operação (uma linha por minuto. Visto que, no caso dos sensores a quantidade de dados perdidos é menor, conclui-se que a causa da perda maior para a ECU não se deve ao registo e envio dos dados do cartão SD para o servidor, mas à forma como o ESP processa os bytes enviados pela ECU. Como o ESP só aceita determinadas mensagens, uma possibilidade é haver dessincronização entre o ESP e a ECU, fazendo com que não seja lida a totalidade da mensagem em cada ciclo. Mesmo assim, de um modo geral os resultados foram bastante razoáveis, visto que os rácios de dados perdidos são bastante reduzidos.

Por outro lado, durante os ensaios de campo haverá por vezes falhas de WiFi. Para avaliar como é que o data logger se irá comportar se perder a conexão ao servidor durante vários minutos/horas foi feito um teste no qual se desligou o router durante vários intervalos de tempo. Nesses intervalos, deixou-se o data logger a recolher dados e a tentar reestabelecer a ligação e ao fim do intervalo de tempo definido voltou-se a ligar o router. O primeiro objetivo foi avaliar se o ESP conseguiria restabelecer a ligação sem a necessidade de reiniciar. Todos os testes realizados validaram esta funcionalidade. Os resultados deste testes estão apresentados na figura 5.11, que representa o rácio de dados em falta para cada intervalo de tempo sem ligação ao servidor.

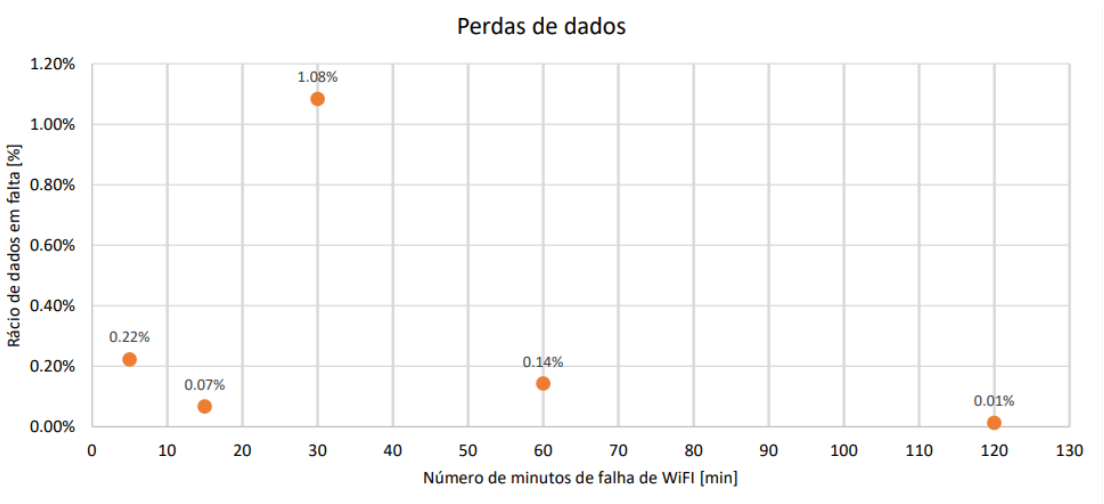

Figura 5.11: Evolução do rácio de dados perdidos em função do intervalo de tempo sem WiFi

Para todos os testes apresentados na figura 5.11, o data logger conseguiu recuperar a ligação WiFi quando o router foi novamente ligado. Além disso, conseguiu recuperar todos os dados que recolheu durante esse intervalo sem qualquer intervenção física no dispositivo. Os rácios de perdas de dados, apresentados na figura 5.11, não têm como causa a falha de WiFi, mas sim a própria recolha dos dados. Quando recupera a ligação ao servidor, o data logger começa a enviar dados a partir da zona onde ocorreu a última publicação de mensagens com sucesso. Quanto mais tempo sem ligação ao servidor, maior será a quantidade de dados acumulados e por isso mais tempo irá precisar para enviar todos os dados armazenados e voltar ao funcionamento normal. No envio de grandes quantidades de dados é comum haver falhas momentâneas na ligação ao servidor, contudo o código implementado revelou-se capaz de lidar com estas situações sem que houvesse perdas de dados.

Os testes realizados permitiram concluir que o data logger consegue operar durante longos períodos de tempo sem perder quantidades de dados significativas. Além disso, em caso de falhas de conectividade, consegue recuperar a ligação e recuperar todos os dados que foram registados durante esse intervalo de tempo. As causas das perdas de dados estão relacionadas, quase na totalidade, com eventos inesperados na recolha dos dados.

Para avaliar a velocidade de envio de dados para o servidor foi desligado o WiFi durante 5 minutos para deixar o ESP acumular dados no cartão SD. Ao fim desse tempo voltou-se a ligar o WiFi e registou-se o tempo que o ESP demorou a recuperar todos os dados que foram recolhidos durante esse intervalo de tempo. Este teste permitiu concluir que, para enviar dados dos três tipos (sensores, mensagens ECU e parâmetros ECU) relativos a um intervalo de tempo de 5 minutos, o ESP demora cerca de 4 minutos, incluindo o tempo que precisa para reestabelecer a ligação, enviar os dados anteriores e enviar os dados que entretanto também recolhe. Este intervalo de tempo é importante porque, caso a ligação à internet seja mesmo muito instável, o ESP poderá começar a ter problemas relacionados com a quantidade de dados acumulados. Algo a melhorar em trabalho futuro será migrar toda a parte de descodificação das mensagens recebidas pela

interface UART para o script de inserção de dados na base de dados. Não só fornece mais versatilidade, como também permitirá reduzir signifcativamente a carga de funções e memória exigida ao ESP.

5.5 Desempenho do algoritmo

O algoritmo foi alvo de um conjunto de testes para avaliar a capacidade de identificação de outliers quando eles existem e não identificar pontos normais como outliers. Nesse sentido, foram realizados dois testes com natureza distinta: inicialmente utilizou-se o script de inserção de dados na BD para, de forma aleatória, inserir alguns outliers nos dados enviados para a base de dados. Foi selecionada uma determinada probabilidade de gerar esses pontos aleatórios, que foi também usada como parâmetro de contaminação do algoritmo. Posteriormente, foi realizado um teste no qual foram forçados outliers localmente nos sensores interferindo fisicamente no hardware. A principal motivação, para a utilização desta abordagem, foi a quantidade reduzida de outliers produzidos pelo data logger durante os testes realizados.

O primeiro teste realizado consitiu em gerar pontos aleatórios com uma probabilidade de 0.1 %. Por esse motivo, foi fornecido ao algoritmo um parâmetro de contaminação de 0.1 %. Na figura 5.12 estão apresentados os resultados para a temperatura ambiente, onde está apresentada a série temporal e os outliers identificados (pontos azuis). Conforme

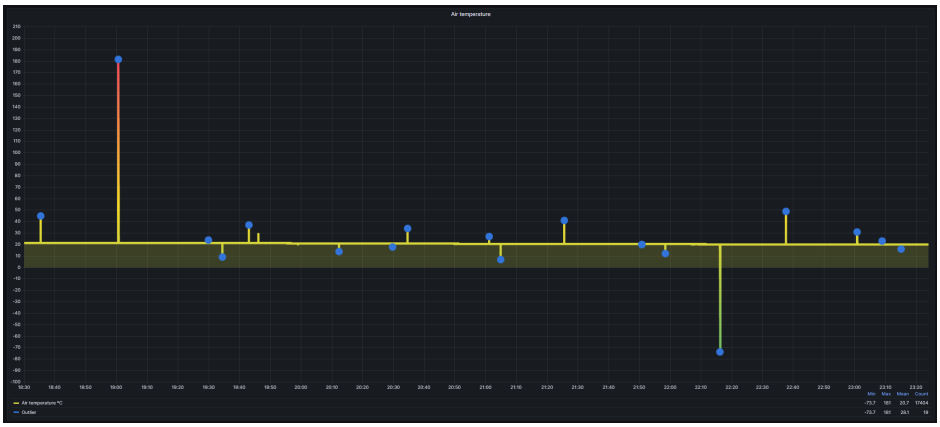

Figura 5.12: Identificação de outliers pelo algoritmo Isolation Forest para o primeiro teste realizado, gerando outliers aleatoriamente, para a temperatura ambiente

está apresentado na figura 5.12, ao longo de cerca de 5 horas, o algoritmo detetou 19 outliers em 17404 dados, ou seja 0.109 % de outliers. Analisando a série de dados, a identificação dos outliers acertou praticamente em todos os casos exceto num. Houve também um ponto normal que foi identificado como outlier. No apêndice C podem ser consultados os resultados para outros parâmetros.

Com base nos resultados do primeiro teste o algoritmo conseguiu detetar quase todos os outliers gerados e praticamente não identificou pontos normais erradamente. No entanto, é importante notar que este teste utilizou uma probabilidade de gerar outliers fixa e um parâmetro de contaminação igual a essa probabilidade, o que facilitou o processo. Numa aplicação real, nomeadamente no caso desta aplicação não será possível fornecer um parâmetro de contaminação tão próximo da quantidade de outliers geradas, visto que

esses eventos serão gerados pontualmente e de forma bastante aleatória. Por este motivo, foi importante realizar um teste em que se manteve o parâmetro de contaminação mas deixou-se de gerar outliers através do script de inserção de dados na BD. Na figura 5.13 estão representados os resultados deste teste para a temperatura ambiente. Na figura 5.13 todos os outliers representados foram dados que o próprio data logger enviou para o servidor, sem qualquer interferência. Estes outliers são gerados quando há maus contactos entre o microcontrolador e o sensor, por exemplo. Para o intervalo de tempo considerado (perto de 7 horas) o algoritmo identificou 13 outliers, dos quais 12 foram bem identificados. No entanto, houve 3 pontos anormais que não foram identificados. No apêndice C podem ser consultados os resultados para as leituras do sensor de pressão hidráulica.

Figura 5.13: Outliers identificados pelo algoritmo durante o teste realizado em que não se inseriram outliers aleatórios

Com base nestes resultados chegou-se à conclusão de que o algoritmo Isolation Forest é uma opção bastante eficaz para identificar outliers quando é necessário utilizar um algoritmo não supervisionado. É um algoritmo bastante sensível ao parâmetro de contaminação. Contudo, tendo conhecimento suficiente dos dados recolhidos ao ponto de poder aferir uma estimativa do rácio esperado de outliers, o algoritmo consegue identificar os outliers com uma eficácia por volta dos 90 %. Caso não haja qualquer outlier, o algoritmo mostrou-se capaz de não identificar qualquer ponto normal como outlier e na presença de pontos anormais conseguiu identificar praticamente todos.

5.6 Visualização e precisão métricas

Nesta secção será avaliada a visualização e precisão das métricas calculadas a partir da agregação das séries temporais. Para isso, foi utilizado o mesmo teste cujas condições foram descritas na tabela 5.1 e as respetivas séries temporais, representadas na figura 5.5 para validar as métricas calculadas pelo sistema. Face ao teste realizado, foram realizadas algumas estimativas (com uma margem de erro de cerca de 2-3%):

- Número de arranques: 3
- Energia consumida: $\approx 44\,\text{kWh}$

- Tempo ligado: ≈ 2.1 h
- Tempo desligado: ≈ 1.9 h

Posto isto, apresentam-se na figura 5.14 algumas das métricas calculadas pelo próprio sistema, para serem comparadas com as estimativas anteriores. É importante notar que o cálculo das distribuições dos vários parâmetros, considera apenas os intervalos de tempo em que o equipamento esteve ligado, ou seja é de esperar que o somatório de todas as distribuições seja igual ao tempo que o equipamento esteve ON e não ao tempo total do teste. Conforme se pode verificar na figura 5.14 o número de arranques corresponde ao

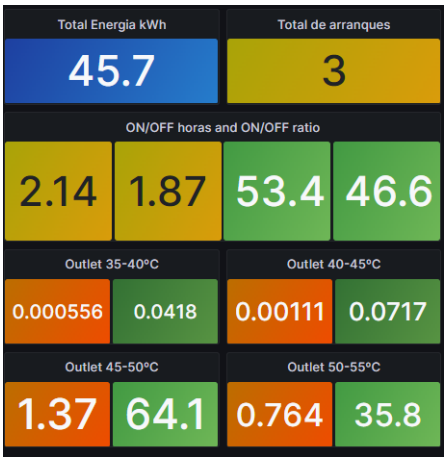

Figura 5.14: Métricas calculadas pelo sistema para o teste descrito na tabela 5.1

que era esperado e a energia consumida está muito próxima do valor estimado. Nas cerca de 4 horas de teste, as métricas relativas ao tempo que o equipamento esteve ligado e desligado também estão de acordo com o esperado. O sistema calculou um tempo ON de 2.14 horas e um tempo OFF de 1.87 horas. Analisando o gráfico da temperatura de saída da figura 5.5, pode-se verificar que, neste teste, a temperatura de saída esteve entre 45 e 50 ºC pouco mais de metade do tempo (que esteve em funcionamento) e menos de metade entre 50-55 ºC. Os resultados da figura 5.14 estão de acordo com estas condições. Apresenta 64.1 % do tempo ON com uma temperatura de saída no intervalo 45-50ºC e 35.8 % no intervalo 50-55 ºC. Somando o número de horas apresentadas na figura 5.14 para cada um destes intervalos o resultado é $1.37 + 0.764 = 2.134$, que também está de acordo com a estimativa de 2.1 h em funcionamento.

Além das métricas também foi avaliada a precisão das extrapolações, calculadas a partir das métricas. Para os resultados apresentados na figura 5.14, as respetivas extrapolações calculadas pelo próprio sistema foram:

- Número de arranques: 6564
- Energia consumida: 99897 kWh
- Tempo Ligado: 4677 horas
- Horas com temperatura de saída entre:
  - 45 - 50 ºC: 3000 horas

– 50 - 55 ºC: 1673 horas

Para validar os cálculos efetuados pode ser utilizada a seguinte relação matemática:

$$Valor\ extrapolado = \frac{horas_{ano} \times metrica}{\Delta t} \tag{5.1}$$

A equação anterior permitiu validar as extrapolações efetuadas, a partir das métricas calculadas. Face às métricas apresentadas na figura 5.14, as extrapolações apresentadas pelo sistema estão de acordo com os resultados calculados utilizando a equação anterior.

Desta forma, concluiu-se que as métricas e extrapolações calculadas pelo sistema implementado traduzem a realidade da operação dos equipamentos.

Capítulo 6

Conclusão e trabalhos futuros

A solução implementada mostrou-se capaz de satisfazer os requisitos iniciais. O sistema desenvolvido pode ser instalado em qualquer lugar com ligação à internet, e sem a necessidade de conhecimentos ou equipamentos especiais. O sistema consegue recolher dados de uma vasta gama de equipamentos, através dos sensores ou através da comunicação com a ECU e enviá-los para um servidor na cloud. No servidor, um frontend dinâmico e interativo apresenta ao utilizador um conjunto de métricas, que são calculadas de forma autónoma, a partir das séries temporais. Na gestão do sistema implementado, apenas há necessidade de intervenção manual nos momentos de configuração dos data loggers.

A capacidade de processamento do ESP32 revelou-se capaz de gerir toda a recolha e envio dos dados para o servidor. A programação multi-core permitiu separar as tarefas de recolha e envio, tornando-as independentes. Os mecanismos implementados para mitigar possíveis falhas no data logger cumpriram com os requisitos: em caso de falhas WiFi, o data logger consegue recuperar a ligação, e os dados que não foram enviados enquanto esteve sem ligação ao servidor. Em caso de falha de energia, o data logger consegue recuperar o funcionamento normal, assim que o router também voltar ao normal. A utilização do cartão de memória, como agente intermediário entre a recolha e envio de dados ao servidor, permitiu o armazenamento de dados offline e a separação das duas principais tarefas. Isto tornou o data logger mais eficiente e robusto.

A interface local implementada permite a um instalador configurar parâmetros cruciais do data logger. Através de LEDs ou de uma interface WEB, pode receber feedback do seu estado de operação, incluindo a visualização de dados, que estão a ser recolhidos pelo dispositivo.

A base de dados, InfluxDB, apresentou resultados positivos para a gestão das séries temporais. Esta BD facilitou bastante os processos de leitura e escrita na base de dados. Por outro lado, para informações que não dependem da variável tempo, nomeadamente os perfis de carga, esta base de dados não é tão adequada e tornou a solução mais complexa.

Os ficheiros de configuração mostraram-se eficazes na configuração dos data loggers. Na eventualidade de um data logger ser mudado de localização, ensaio ou equipamento é possível configurar um novo ensaio remotamente, apenas alterando alguns parâmetros nesses ficheiros.

O processamento dos dados que foi implementado permite obter os perfis de carga dos equipamentos automaticamente. Além disso, não há mistura de dados de ensaios diferentes e é sempre coberta a totalidade do ensaio. As métricas apresentadas traduzem, com precisão, a realidade das condições, a que os equipamentos foram sujeitos. O

processamento foi implementado com um atraso, relativamente à recolha dos dados, com o objetivo de permitir que o data logger recupere de uma eventual falha de WiFi. Este atraso permite que falhas de algumas horas não tenham impacto no cálculo das métricas, contudo qualquer dado que seja enviado com um atraso superior ao previsto não será considerado.

O algoritmo Isolation Forest mostrou-se capaz de detetar a maioria dos outliers. O pré-processamento implementado permitiu melhorar a precisão da agregação das séries temporais, removendo outliers e preenchendo quaisquer dados em falta.

O frontend construído no Grafana permite que o utilizador interaga com as dashboards dinâmicamente. Algumas das funcionalidades que o frontend oferece são: filtrar os dados por ensaios de campo, equipamentos ou data loggers, configurar intervalos de tempo e resolução dos dados apresentados e visualizar o estado de operação dos data loggers.

Apesar de a solução cumprir com os requisitos ainda há várias possibilidades para melhorar a solução implementada. Algumas limitações que poderão ser trabalhadas no futuro são:

- Migrar a descodificação dos dados, realizada localmente pelo data logger, para o script de inserção de dados na base de dados. Até ao momento, o data logger envia para o servidor a maior parte dos dados já descodificados. Para este efeito, necessita que as retas de calibração dos sensores e a descodificação das mensagens das ECUs façam parte do código transferido para o ESP. Embora o data logger tenha a capacidade de descodificar dados de várias ECUs, qualquer alteração que tenha de ser realizada implica alterar o código de todos os data loggers. Além disso, no caso dos sensores, se se pretender utilizar um sensor NTC diferente daqueles que foram previstos inicialmente, é necessário alterar o programa do ESP. Por estes motivos, o principal trabalho futuro que se propõe é migrar toda a descodificação dos dados para o script que envia os dados para a base de dados. Desta forma, qualquer modificação, que seja necessária, poderá ser realizada facilmente, através do servidor.

- A base de dados InfluxDB não é a mais adequada para armazenar os perfis de carga dos equipamentos. Por este motivo, pode ser implementado um novo ramo na arquitetura, responsável por recolher os perfis de carga e guardá-los numa base de dados mais adequada.

- A PCB desenvolvida até ao momento foi uma primeira versão, com o objetivo de validar um conceito. Como trabalho futuro, poderá ser desenhada uma placa PCB mais compacta e que incluia espaço para uma fonte de alimentação. Nessa nova versão podem ser incluidas as alterações que foram efetuadas ao esquema elétrico, já depois de construída a PCB.

Apêndice A

Dashboards Grafana

Neste apêndice serão apresentadas as várias dashboards desenvolvidas. Os dados apresentados em cada figura devem ser vistos apenas como exemplo. A figura A.1 representa

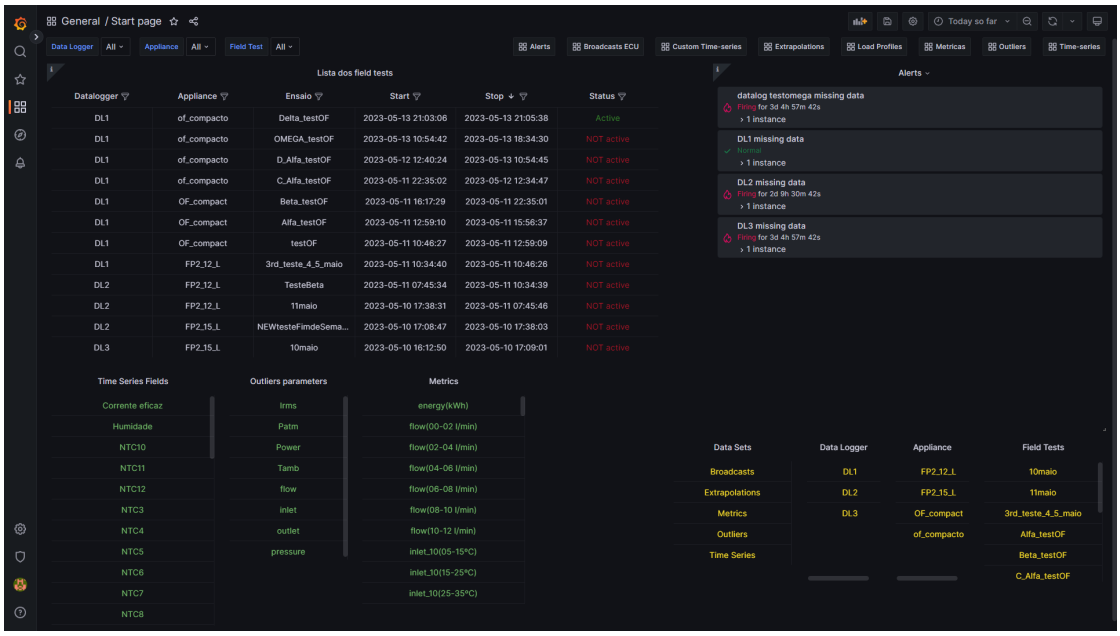

Figura A.1: Página inícial do frontend

a página inicial, onde um utilizador poderá obter um conjunto de informação geral acerca dos data loggers e ensaios. Nesta dashboard são ainda apresentados alertas, caso algum data logger não envie dados ao servidor durante 1 hora. As figuras A.2 e A.3 apresentam a dashboard com as séries temporais, incluindo parâmetros associados aos sensores e à ECU dos equipamentos FP2. A figura A.4 apresenta a dashboard relativa aos outliers identificados pelo algoritmo utilizado.

As figuras A.5 e A.6 apresentam as dashboards associadas às métricas calculadas e às extrapolações, respetivamente. Por último, a figura A.7 apresenta a dashboard com as mensagens recolhidas das ECUs dos equipamentos, por descodificar.

Figura A.2: Dashboard de séries temporais relativa à ECU dos equipamentos FP2

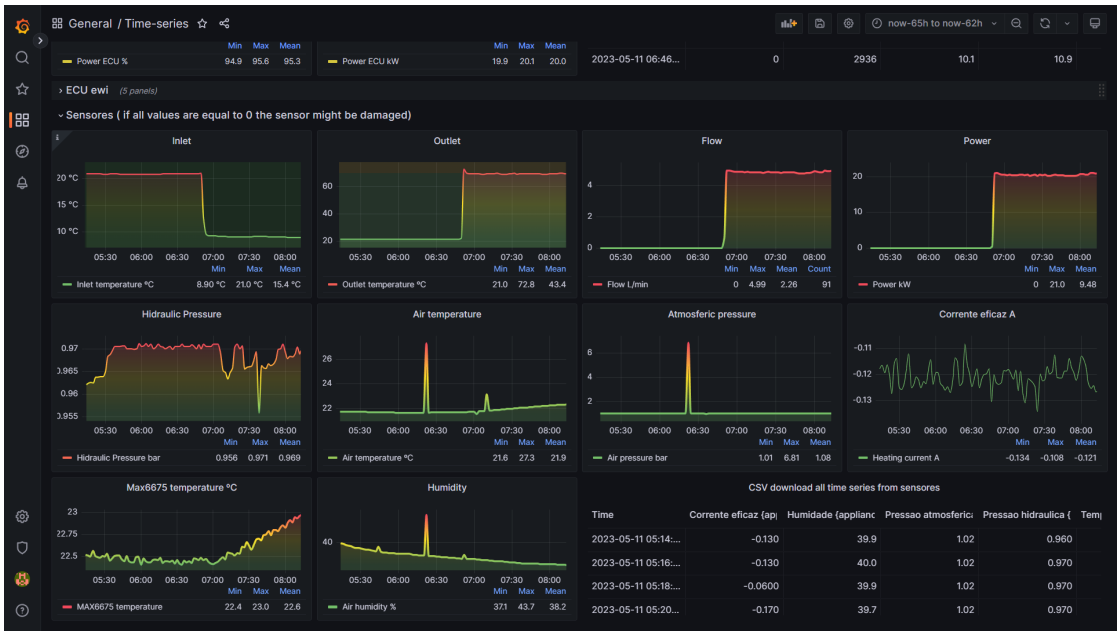

Figura A.3: Dashboard de séries temporais relativa aos sensores externos

Figura A.4: Dashboard relativa aos outliers identificados pelo algoritmo Isolation Forest

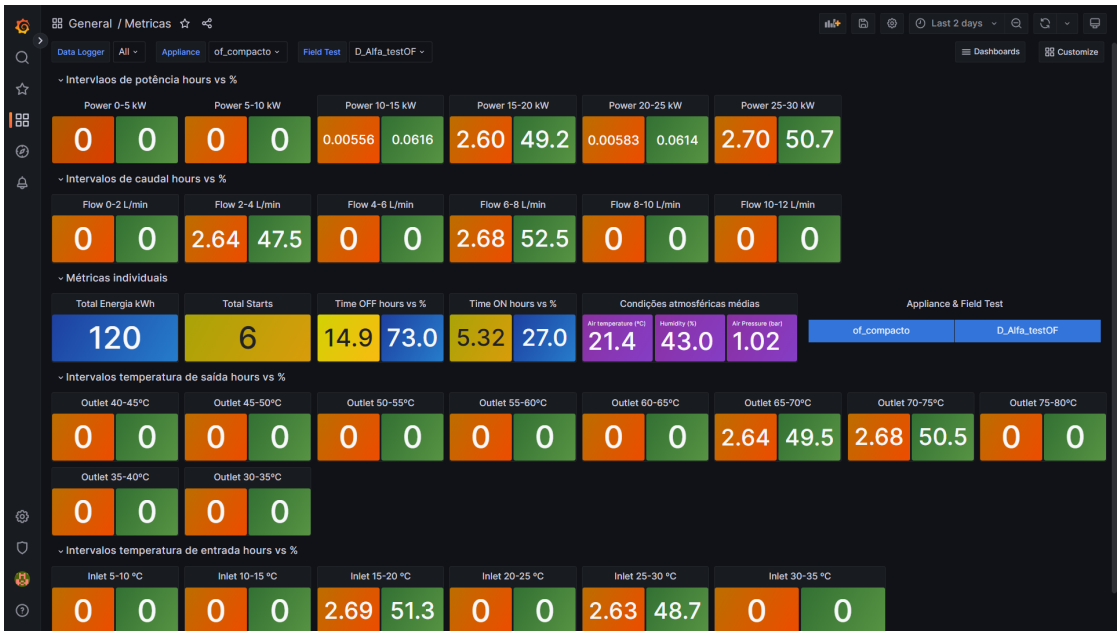

Figura A.5: Dashboard relativa às métricas calculadas no processamento das séries temporais

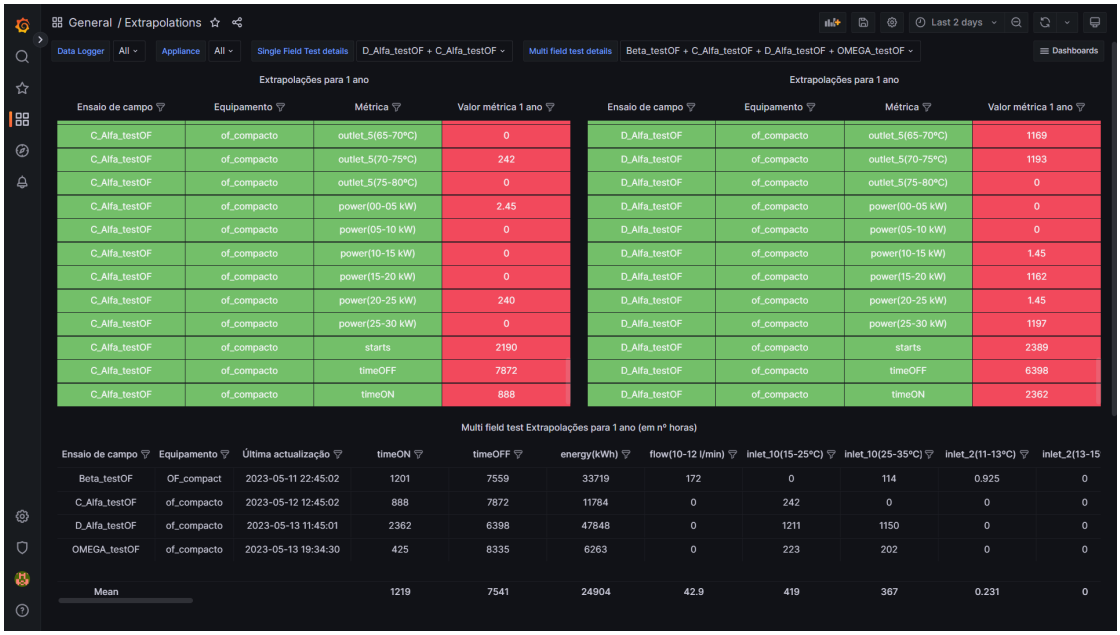

Figura A.6: Dashboard relativa às extrapolações das métricas

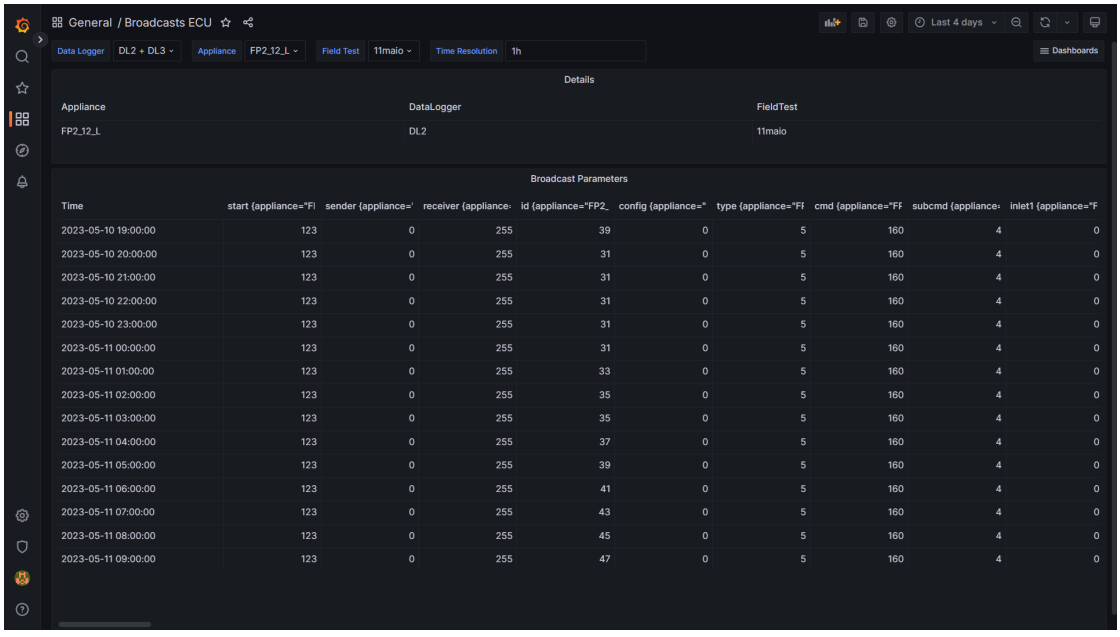

Figura A.7: Dashboard relativa às mensagens recebidas da ECU do equipamento, por descodificar

Apêndice B

Estado dos LEDs

Nesta secção são apresentadas fotografias do estado dos LEDs durante vários estados de funcionamento do data logger.

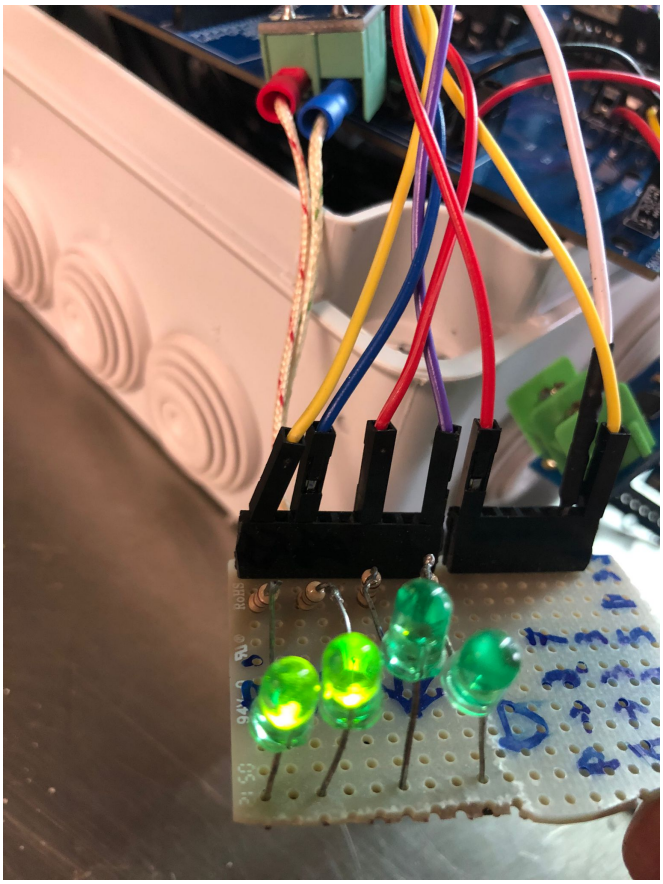

Figura B.1: Estado dos LEDs durante a ligação do ESP em modo AP

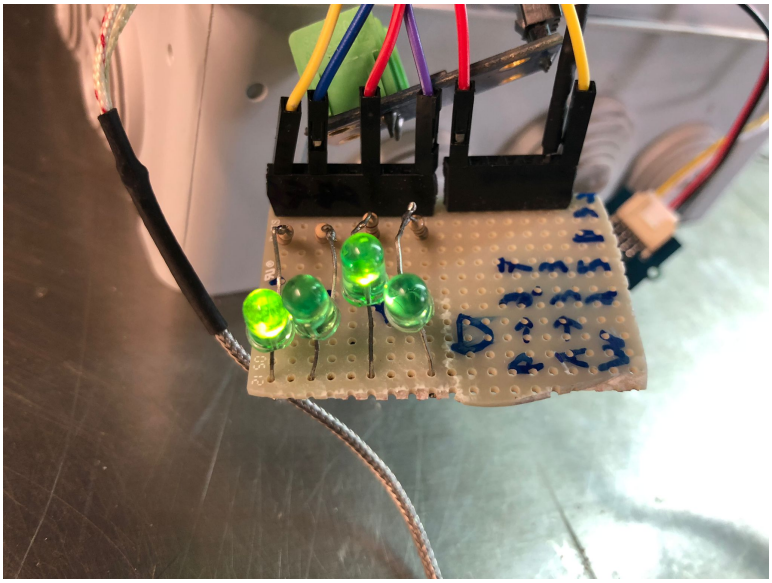

Figura B.2: Estado dos LEDs durante a operação normal do ESP

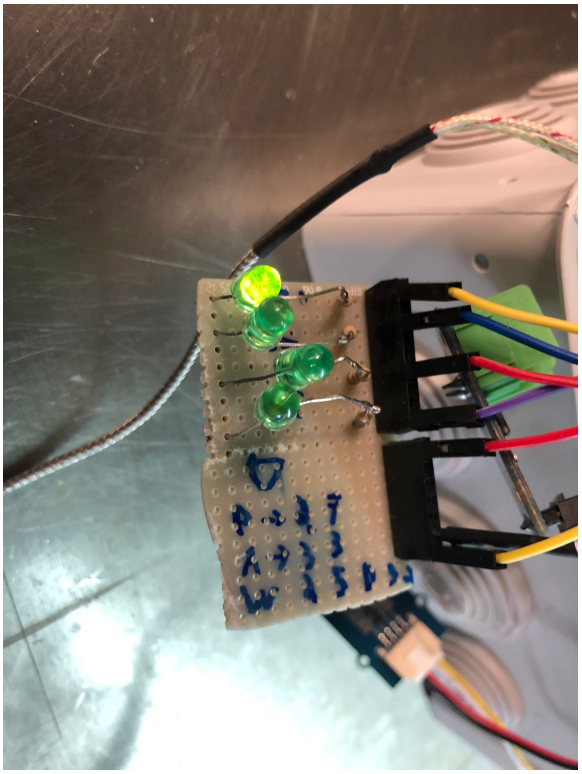

Figura B.3: Estado dos LEDs durante a operação normal do ESP mas quando há uma perda de WiFi

Apêndice C

Resultados do algoritmo Isolation Forest

Neste apêndice são apresentados resultados obtidos na análise do desempenho do algoritmo.

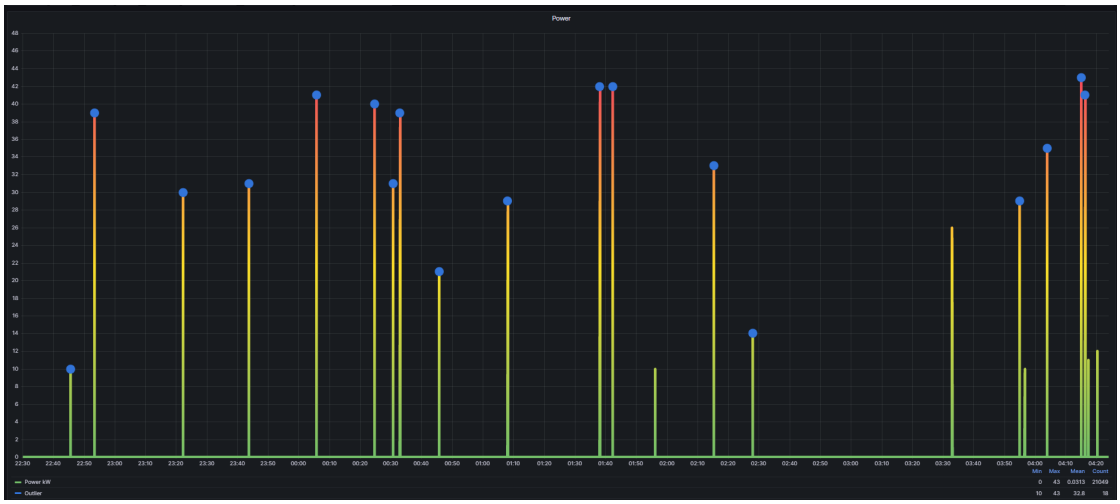

Figura C.1: Identificação de outliers na série temporal potência, onde foram gerados outliers aleatoriamente

Figura C.2: Identificação de outliers na série temporal pressão hidráulica, onde foram gerados outliers aleatoriamente

Figura C.3: Outliers identificados pelo algoritmo na série pressão hidráulica, durante o teste em que não foram gerados outliers aleatoriamente

Apêndice D

Exemplo do processo de leitura de dados da BD InfluxDB

Na figura D.1 está representada um exemplo de uma query que começa por selecionar a fonte de dados (bucket), depois aplica um filtro para escolher o data logger, depois continua a filtrar para só serem fornecidos dados de uma determinada "appliance". Depois disso aplica outro filtro para definir qual o field que deseja ler. Na query anterior

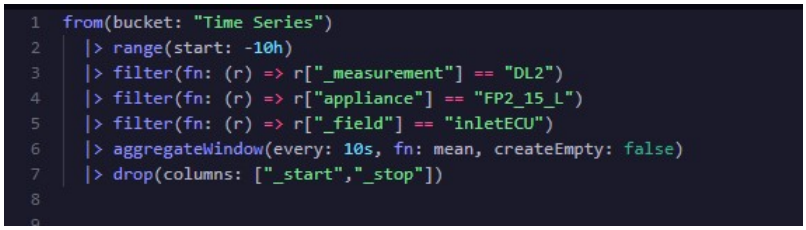

Figura D.1: Exemplo de uma query enviada à base de dados InfluxDB

são ainda aplicadas três funções: "range" permite definir o intervalo de tempo do qual se pretende ler dados e "aggregateWindow" que permite agregar os dados lidos de vários formas para diminuir a quantidade de dados. Quando o número de dados é demasiado grande, os sistemas podem ficar lentos e as queries demoradas e esta função permite resolver esse problema. A outra função utilizada é a "drop", que permite remover determinadas colunas dos dados devolvidos pela base de dados. O resultado desta query está representado na figura D.2. Conforme está representado na figura D.2 a query da figura D.1 retornou uma tabela em que as linhas são separadas por timestamp, de 10s em 10s. Em cada linha tem-se um field (neste caso "inletECU", um valor (que neste caso foi sempre igual a 21), a ID do data logger (DL2), o equipamento (FP2_15_L) e o ensaio (NewtesteFimdeSemana).

_time	_value	_field	_measure...	appliance	fieldTest
2023-05-10 0...	21	inletECU	DL2	FP2_15_L	NEWtesteFim...
2023-05-10 0...	21	inletECU	DL2	FP2_15_L	NEWtesteFim...
2023-05-10 0...	21	inletECU	DL2	FP2_15_L	NEWtesteFim...
2023-05-10 0...	21	inletECU	DL2	FP2_15_L	NEWtesteFim...
2023-05-10 0...	21	inletECU	DL2	FP2_15_L	NEWtesteFim...
2023-05-10 0...	21	inletECU	DL2	FP2_15_L	NEWtesteFim...
2023-05-10 0...	21	inletECU	DL2	FP2_15_L	NEWtesteFim...
2023-05-10 0...	21	inletECU	DL2	FP2_15_L	NEWtesteFim...
2023-05-10 0...	21	inletECU	DL2	FP2_15_L	NEWtesteFim...
2023-05-10 0...	21	inletECU	DL2	FP2_15_L	NEWtesteFim...

Figura D.2: Dados retirados a partir da query da figura D.1. Print retirado diretamente da interface Web do InfluxDB

Referências

[1] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview understanding the issues and challenges of a more connected world," *The Internet Society*, 2015. [Online]. Disponível em: https://www.internetsociety.org/resources/doc/2015/iot-overview/.

[2] C. Kaur, "The cloud computing and internet of things (iot)," *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 7, pp. 19–22, 1 2020. [Online]. Disponível em: https://www.researchgate.net/publication/338490865_The_Cloud_Computing_and_Internet_of_Things_IoT.

[3] H. J. Ding, "Traffic flow data collection and signal control system based on internet of things and cloud computing," *Advanced Materials Research*, vol. 846-847, pp. 1608–1611, 2014. [Online]. Disponível em: https://www.scientific.net/AMR.846-847.1608.

[4] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, pp. 685–695, 9 2021. [Online]. Disponível em: https://link.springer.com/article/10.1007/s12525-021-00475-2.

[5] A. R. Santiago, M. Antunes, J. P. Barraca, D. Gomes, and R. L. Aguiar, "Scotv2: Large scale data acquisition, processing, and visualization platform," in *Proceedings - 2019 International Conference on Future Internet of Things and Cloud, FiCloud 2019*, pp. 318–323, Institute of Electrical and Electronics Engineers Inc., 8 2019. [Online]. Disponível em: https://ieeexplore.ieee.org/document/8972819.

[6] D. J. Camarneiro, "Internet das coisas na indústria, e a conectividade na bosch," Dissertação de mestrado, DEM, UA, Aveiro, PT, 2021.

[7] D. F. S. Coelho, "A predictive maintenance approach based on time series segmentation," Dissertação de mestrado, DEM, UA, Aveiro, PT, 2021.

[8] "Temperature sensors: Types, uses, benefits, design." [Online]. Disponível em: https://www.iqsdirectory.com/articles/thermocouple/temperature-sensors.html (Acedido a 5 de mar. 2023).

[9] J. Morais, "Esp32 - analisando e corrigindo o adc interno - embarcados." [Online]. Disponível em: https://embarcados.com.br/esp32-adc-interno/(Acedido a 5 de mar. 2023).

[10] "Turbine flow meter: Technology and working principle | eltra trade." [Online]. Disponível em: https://eltra-trade.com/blog/turbine-flow-meter (Acedido a 15 de fev. 2023).

[11] "How to measure current using current sensors | dewesoft." Disponível em: https://dewesoft.com/blog/how-to-measure-current-using-current-sensors (Acedido a 15 de fev. 2023).

[12] P. Gackowiec and M. Podobińska-Staniec, "Iot platforms for the mining industry: An overview," *Journal of the Polish Mineral Engineering Society*. [Online]. Disponível em: http://doi.org/10.29227/IM-2019-01-47.

[13] N. Naik, "Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http," in *2017 IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 10 2017. [Online]. Disponível em: https://ieeexplore.ieee.org/document/8088251.

[14] G. Sasikala, Y. P. S. Chandra, N. Siva, al, Z. Wang, D. Han, Y. Gong, N. Tyutyundzhiev, C. Angelov, T. Arsov, K. Lovchinov, H. Nitchev, A. Mutafov, and G. Alexieva, "Remote datalogging of solar uv irradiation using open-source esp32 platform and mqtt protocol," *Journal of Physics: Conference Series*, vol. 2436, p. 012003, 1 2023. [Online]. Disponível em: https://iopscience.iop.org/article/10.1088/1742-6596/2436/1/012003.

[15] A. Zare and M. T. Iqbal, "Low-cost esp32, raspberry pi, node-red, and mqtt protocol based scada system," in *IEMTRONICS 2020 - International IOT, Electronics and Mechatronics Conference, Proceedings*, Institute of Electrical and Electronics Engineers Inc., 9 2020. [Online]. Disponível em: https://ieeexplore.ieee.org/document/9216412.

[16] B. Demircan and E. Akyüz, "Iot and cloud based remote monitoring of wind turbine," *Celal Bayar University Journal of Science*, vol. 15, pp. 337–342, 12 2019. [Online]. Disponível em: https://dergipark.org.tr/en/pub/cbayarfbe/issue/50875/540812.

[17] O. Panagopoulos and A. A. Argiriou, "Low-cost data acquisition system for solar thermal collectors," *Electronics 2022, Vol. 11, Page 934*, vol. 11, p. 934, 3 2022. [Online]. Disponível em: https://www.mdpi.com/2079-9292/11/6/934.

[18] CavideBalkıGemirter, "A comparative evaluation of amqp, mqtt and http protocols using real-time public smart city data," in *6th International Conference on Computer Science and Engineering (UBMK)*, 2021. [Online]. Disponível em: https://ieeexplore.ieee.org/document/9559032.

[19] B. Wukkadada, K. Wankhede, R. Nambiar, A. Nair, A. Professor, and K. Somaiya, "Comparison with http and mqtt in internet of things (iot)," in *International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2018. [Online]. Disponível em: https://ieeexplore.ieee.org/document/8597401.

[20] R. Krishnamurthi, A. Kumar, D. Gopinathan, A. Nayyar, and B. Qureshi, "An overview of iot sensor data processing, fusion, and analysis techniques," *Sensors*

2020, Vol. 20, Page 6076, vol. 20, p. 6076, 10 2020. [Online]. Disponível em: https://www.mdpi.com/1424-8220/20/21/6076.

[21] A. A. Al-khatib, M. Balfaqih, and A. Khelil, "A survey on outlier detection in internet of things big data," *Big Data-Enabled Internet of Things*, pp. 225–272, 1 2020. [Online]. Disponível em: https://www.researchgate.net/publication/338019121_A_survey_on_outlier_detection_in_Internet_of_Things_big_data.

[22] P. Zhang, J. Wang, K. Guo, F. Wu, and G. Min, "Multi-functional secure data aggregation schemes for wsns," *Ad Hoc Networks*, vol. 69, pp. 86–99, 2 2018. [Online]. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1570870517302032.

[23] A. A. Cook, G. Misirli, and Z. Fan, "Anomaly detection for iot time-series data: A survey," *IEEE Internet of Things Journal*, vol. 7, pp. 6481–6494, 7 2020. [Online]. Disponível em: https://ieeexplore.ieee.org/document/8926446.

[24] G. Enderlein, "Hawkins, d. m.: Identification of outliers. chapman and hall, london – new york 1980, 188 s., £ 14, 50," *Biometrical Journal*, vol. 29, pp. 198–198, 1 1987. [Online]. Disponível em: https://onlinelibrary.wiley.com/doi/10.1002/bimj.4710290215.

[25] H. P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 444–452, 2008. [Online]. Disponível em: https://dl.acm.org/doi/10.1145/1401890.1401946.

[26] J. Janssens, "Stochastic outlier selection | jeroen janssens." [Online]. Disponível em: https://jeroenjanssens.com/sos/ (Acedido a 18 de feb. 2023).

[27] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, "Copod: Copula-based outlier detection," in *Proceedings - IEEE International Conference on Data Mining, ICDM*, vol. 2020-November, pp. 1118–1123, Institute of Electrical and Electronics Engineers Inc., 11 2020. [Online]. Disponível em: https://www.researchgate.net/publication/344306968_COPOD_Copula-Based_Outlier_Detection.

[28] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 413–422, IEEE, 2008. [Online]. Disponível em: https://ieeexplore.ieee.org/document/4781136.

[29] T. Mendes, P. J. Cardoso, J. Monteiro, and J. Raposo, "Anomaly detection of consumption in hotel units: A case study comparing isolation forest and variational autoencoder algorithms," *Applied Sciences 2023, Vol. 13, Page 314*, vol. 13, p. 314, 12 2022. [Online]. Disponível em: https://www.mdpi.com/2076-3417/13/1/314.

[30] D. Adhikari, W. Jiang, J. Zhan, Z. He, D. B. Rawat, U. Aickelin, and H. A. Khorshidi, "A comprehensive survey on imputation of missing data in internet of things," *ACM Computing Surveys*, vol. 55, 12 2022. [Online]. Disponível em: https://dl.acm.org/doi/10.1145/3533381.

[31] J. Poloczek, N. A. Treiber, and O. Kramer, "Knn regression as geo-imputation method for spatio-temporal wind data," *Advances in Intelligent Systems and Computing*, vol. 299, pp. 185–193, 2014. [Online]. Disponível em: `https://link.springer.com/chapter/10.1007/978-3-319-07995-0_19`.

[32] S. Fanourakis, K. Wang, P. McCarthy, and L. Jiao, "Low-cost data acquisition systems for photovoltaic system monitoring and usage statistics," *IOP Conference Series: Earth and Environmental Science*, vol. 93, p. 012048, 11 2017. [Online]. Disponível em: `https://iopscience.iop.org/article/10.1088/1755-1315/93/1/012048`.

[33] Y. Regaya, F. Fadli, and A. Amira, "Point-denoise: Unsupervised outlier detection for 3d point clouds enhancement," *Multimedia Tools and Applications*, vol. 80, pp. 1–17, 07 2021.

[34] K. Seu, M. S. Kang, and H. Lee, "An intelligent missing data imputation techniques: A review," *JOIV : International Journal on Informatics Visualization*, vol. 6, pp. 278–283, 5 2022. [Online]. Disponível em: `http://joiv.org/index.php/joiv/article/view/935`.

[35] T. Mahboob, A. Ijaz, A. Shahzad, and M. Kalsoom, "Handling missing values in chronic kidney disease datasets using knn, k-means and k-medoids algorithms," pp. 76–81, Institute of Electrical and Electronics Engineers Inc., 1 2019. [Online]. Disponível em: `https://ieeexplore.ieee.org/document/8632179`.