



Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

João Paulo Lopes Agostinho

Otimização e Compressão de Páginas Web para Sistemas Embebidos e desenvolvimento de Equipamentos IOT

Relatório de Estágio

Orientado por:

Renato Eduardo da Silva Panda, Instituto Politécnico de Tomar

Júri (caso seja conhecido) + Instituição

Relatório de Estágio
apresentada ao Instituto Politécnico de Tomar
para cumprimento dos requisitos necessários
à obtenção do grau de Mestre
em Engenharia Informática – Internet das Coisas

AGRADECIMENTOS

Quero agradecer à minha família e amigos, pelo apoio dado nos bons e maus momentos não só durante a vida académica, mas durante toda a minha vida.

Aos meus colegas de curso e professores pelos bons tempos que foram passados nas aulas deste Mestrado.

Quero agradecer igualmente ao meu orientador, o professor Renato Panda por ser meu orientador e estar sempre disponível para ajudar nesta última fase do Mestrado.

À empresa Captemp pela possibilidade de realizar o estágio para conclusão de mais uma etapa da minha vida.

RESUMO

Este relatório de estágio foi realizado no âmbito do estágio inserido no Mestrado em Engenharia Informática-Internet das Coisas da Escola Superior de Tecnologia de Tomar do Instituto Politécnico de Tomar, e tem como objetivo colocar em ambiente real os conhecimentos adquiridos no percurso académico.

O estágio tem inerente 4 projetos associados à área do IOT e da monitorização de ambientes e/ou objetos com recurso a diversas soluções. Um dos projetos, "Nidus" já existia e é necessário a análise do projeto para dar continuidade ao suporte do *Front-end*. Neste projeto é necessário estudar e alterar os métodos de desenvolvimento da página referentes á compressão dos ficheiros para reduzir o espaço ocupado pelas páginas WEB, a adoção de um melhor método para utilização de imagens nas interfaces WEB em equipamentos com baixos recursos e o desenvolvimento de novas funcionalidades tais como um sistema de internalização de modo a disponibilizar o equipamento noutras idiomas. Os restantes projetos serão desenvolvidos de raiz durante o estágio quer por pedido de soluções à medida pelos clientes quer pela inovação/evolução dos produtos da empresa, e consistem em novos equipamentos para monitorização, nomeadamente monitorização ambiental e *Tracking* de objetos.

O segundo projeto é referente ao desenvolvimento do de um equipamento IOT que tira partido da nova tecnologia de comunicação o *NB-IOT*. Neste equipamento é possível adicionar vários sensores, e o equipamento é capaz de realizar as leituras dos diversos sensores, o armazenamento em Log para posterior envio para um portal *Cloud* para consulta futura ou criação de alertas.

Paralelamente ao projeto anterior e com o emergir de novas tecnologias tais como as *Beacons Bluetooth Low Energy* e pela requisição por diversos clientes de uma nova solução de monitorização portátil e simples, foi criado o projeto "Kea Tracker" composto com várias *Beacons* e uma aplicação *Mobile* responsável por ler os sensores presentes nas *Beacons* e o envio para o portal *Cloud*. Neste projeto o *Beacon* deve igualmente possuir um sistema de Log interno para a falha de comunicação com a aplicação *Mobile*.

O último projeto do estágio, solicitado pelo cliente, baseia-se no desenvolvimento de uma solução composta por Sensores e *Gateways* e uma plataforma WEB que seja possível realizar o *Tracking* de pessoas e de objetos em ambientes *indoor* e assim gerir tempos de acesso criar alertas, ou simplesmente gerir o stock de armazéns. Este projeto usa como tecnologia de suporte *Beacons Bluetooth Low Energy* (BLE) para fazer o tracking em tempo real.

Palavras chave: IOT, Monitorização, Compressão WEB, Compressão de Imagens, NB-IOT, Beacon, BLE, Indoor Tracking, Bluetooth Tracking

ABSTRACT

This traineeship report was made in the scope of the traineeship inserted in the master's degree in Computer Engineering -Internet of Things at the School of Technology of Tomar from the Polytechnic Institute of Tomar, and has a goal place in real environment the knowledge acquired in the academic path.

The traineeship has inherent 4 associated projects at the area of IOT and the monitoring of environments and/or objects using different solutions. One of the projects, "Nidus" already existed and it is necessary to analyze the project to continue the support of Front-end. In this project, it is necessary to study and change the page development methods related to file compression to reduce the space occupied by the WEB pages, the adoption of a better method for using images on the WEB interfaces on equipment with low resources and the development of new features such as an internalization system in order to make the equipment available in other languages. The remaining projects will be developed from scratch during the traineeship, or by clients requesting custom solutions or by the innovation/evolution of the company's products, and consist of new equipments for monitoring, namely environmental monitoring and Tracking of objects.

The second project is relative of the development of an IOT device that takes advantage of the new communication technology NB-IOT. In this equipment it is possible to add several sensors, and the equipment is able to perform the readings of the various sensors, storage in Log for later upload to a Cloud portal for future preview or make alerts.

In parallel with the previous project and with the emergence of new technologies such as Bluetooth Low Energy Beacons and the request by several customers for a new portable and simple monitoring solution, the "Kea Tracker" project was created, composed with several Beacons and an application Mobile responsible for reading the sensors present in Beacons and upload them to the Cloud portal. In this project,

Beacon must also have an internal Log system for communication failure with the Mobile application.

The last project of the traineeship, requested by the client, is based on the development of a solution composed of Sensors and Gateways and a WEB platform that makes it possible to Tracking people and objects in indoor environments and thus manage access times, create alerts, or simply manage the stock of warehouses. This project uses as support technology Bluetooth Low Energy Beacons (BLE) to do real-time tracking.

Key words: IOT, Monitoring, WEB Compression, Image Compression, NB-IOT, Beacon, BLE, Indoor Tracking, Bluetooth Tracking

"Persistence is the shortest path to success"

— Charles Chaplin

ÍNDICE

Agradecimentos	iii
Resumo	v
Abstract	viii
Acrónimos	xvi
Índice de Figuras	xvi
Índice de Tabelas	xix
Índice de Algoritmos	xx
1 Introdução	1
1.1 Contextualização	1
1.2 A Empresa	1
1.3 Motivação e Objetivos	4
1.3.1 Nidus	5
1.3.2 NB-Iot	5
1.3.3 Kea Tracker	5
1.3.4 dot.Tracker	6
1.4 Problemas identificados	6
1.5 Organização do relatório	8
2 Estado da Arte	9
2.1 Introdução	9
2.2 Coletor de dados - Nidus	9
2.2.1 Páginas do coletor de dados Nidus	10
2.3 NB-Iot & Digi XBee 3	11
2.3.1 MicroPython	13

2.3.2	NB-Iot/ LTE-M	13
2.4	Kea Tracker	15
2.4.1	<i>Beacons</i> BLE	15
2.4.2	Ruuvi <i>beacons</i>	16
2.4.3	Apps smartphones	16
2.5	dot.Tracker	16
2.5.1	<i>Beacons</i> e <i>Gateway</i>	17
2.6	Soluções e tecnologias disponíveis	17
2.6.1	Tecnologias disponíveis	17
2.6.2	Produtos similares	23
3	Trabalho Desenvolvido	26
3.1	Introdução	26
3.2	Coletor de dados - Nidus	26
3.2.1	Sistema de tradução automática	26
3.2.2	Compressão de ficheiros	30
3.2.3	Compressão de imagens	31
3.2.4	Desenvolvimento a pedido de cliente	33
3.2.5	Correção de <i>bugs</i>	33
3.2.6	Melhoramento da página	36
3.3	NB-Iot & Digi XBee 3	37
3.3.1	Envio de dados para o portal	37
3.3.2	Gestão de memória	38
3.3.3	Sincronismo de leitura e envio	41
3.3.4	Encriptação dos dados	42
3.4	Kea Tracker	43
3.4.1	App - Alterações necessárias	43
3.4.2	App - Novas funcionalidades	44
3.4.3	Desenvolvimento de uma aplicação multi-plataforma	45
3.5	dot.Tracker	46
3.5.1	Portal Cloud - <i>Front-End</i>	47
3.5.2	Portal Cloud - <i>Back-end</i>	49
3.5.3	Reutilização de <i>beacons</i>	55
4	Testes e Avaliação	56
4.1	Nidus	56
4.2	Nb-Iot	56
4.3	Kea Tracker	57

4.4 dot.tracker	57
5 Conclusões e Trabalho Futuro	59
Bibliografia	59
Apêndice	63
A Exemplo de um SVG	64
B Exemplo de um SVG comprimido	67
C Fluxograma dos processos de Registo -App	68
D Fluxograma do processo de Envio - App	69
E Fluxograma de receção de Pacotes BLE	70
F Fluxograma do processo de receção de pacotes	71
G Código da obtenção da posição - LSE	72

ACRÓNIMOS

ÍNDICE DE FIGURAS

1.1	CapTemp SQL	2
1.2	Coletor de dados Nidus-C	2
1.3	Universo Nidus	3
1.4	TH3 e Airo	3
1.5	Portal Senslive	4
1.6	Programação com a ferramenta Scratch	7
1.7	Data Logger iButton	7
2.1	Layout página da Nidus IT no início do estágio	11
2.2	Módulo Xbee 3 e placa de expansão desenvolvida pela Captemp	11
2.3	Gráfico com relação Distancia vs Largura de Banda[1]	14
2.4	<i>BLE broadcast packet</i> [2]	15
2.5	Funcionamento da Janela Deslizante	19
2.6	Sequência não comprimida	19
2.7	Sequência comprimida com LZ77 (apenas palavras)	19
2.8	Sequência comprimida com LZ77(palavras e sequências)	20
2.9	Posição utilizando o método <i>Centroid</i> com 3 e 4 receptores	22
3.1	Processo da obtenção da tradução de um valor	27
3.2	Estrutura JSON do conjunto de dicionários	28
3.3	Gráfico com comparação do tamanho entre versões (em Bytes)	29
3.4	Comparação entre os <i>headers</i> HTTP GZIP e Brotli[3]	30
3.5	Gráfico com comparação do tamanho entre original e otimizado(em Bytes)	32
3.6	Exemplo do SVG utilizado na Solução	33
3.7	Estrutura obtida no JavaScript - 1	35
3.8	Estrutura obtida no JavaScript - 2	36
3.9	Gráfico com comparação do tamanho do plugin Blockly (em Bytes) . .	37
3.10	Estrutura do Pacote NB-IOT	38
3.11	Comportamento da gestão de Memória do MicroPython	39

3.12 Array Circular com leituras	42
3.13 Token JWT codificado e descodificado [4]	49
3.14 Diferença entre distâncias Planta vs realidade	52
3.15 Exemplo da aplicação do Filtro Kalman [5]	54

ÍNDICE DE TABELAS

2.1	Especificações do Módulo RCM6760	10
2.2	Especificações do Módulo XBee 3	12
2.3	Comparação entre <i>beacons</i> [6][7][8]	25
3.1	Comparação entre Brotli e GZIP	31
3.2	Requisitos da solução	47

ÍNDICE DE ALGORITMOS

3.1	Definição do Sistema de Internacionalização	28
3.2	Estrutura parcial do XML da Nidus(Sensores)	34
3.3	Exemplo do XML antes da eliminação de Sensores	34
3.4	Exemplo do XML após eliminação do Sensor	35
3.5	Função findObjectByKey	36
3.6	Exemplo com a concatenação da operadora	40
3.7	Adaptação do Código para o equipamento Digi	42
A.1	Exemplo de um SVG	64
B.1	Exemplo de um SVG comprimido	67
G.1	Implementação do algoritmo LSE em Javascript	72

Capítulo 1

Introdução

1.1 Contextualização

Atualmente a sociedade vive rodeada de tecnologias indispensáveis e de ambientes que se intitulam inteligentes (*smart*), mas nem sempre foi assim.

Desde cedo, mesmo antes de existir tecnologia, o homem tendeu a procurar e encontrar coisas que melhorassem a sua vida e bem-estar pessoal e da sociedade, mas para chegar a humanidade está hoje é necessário recuar na história algum tempo para marcos importantes da tecnologia.

Um dos marcos muito importantes para o desenvolvimento dos sistemas embebidos e de sistemas de monitorização foi a invenção dos processadores. Com o surgimento dos processadores começaram a surgir os primeiros sistemas embebidos e sistemas de monitorização. Com o passar dos anos até aos dias de hoje a tecnologia tem vindo a evoluir e por consequência os sistemas também se adaptaram para os padrões de atualmente.

Uma das partes mais importantes num sistema embebido é a sua interface disponível para o utilizador, as principais e mais usadas nos dias de hoje são a linha de comandos e a WEB, comuns para configurações á distância e as interfaces dos próprios equipamentos como os ecrãs com *software* proprietário.

1.2 A Empresa

A empresa CapTemp, Lda localizada em Pombal, Leiria é uma empresa, focada em desenvolvimento de soluções de monitorização, controlo, supervisão e de soluções à medida consoante os requisitos do cliente. Para criar um sistema de monitorização é necessário o sistema possuir sensores, atuadores, colectores de dados e software para analisar os dados provenientes dos sensores de modo a possuir capacidade de atuar com base nesses valores. A Captemp é responsável pelo desenvolvimento de todos estes componentes passando pelos sensores até ao *software* responsável por analisar e armazenar os dados.

1.2. A EMPRESA

Uma das subáreas da empresa é a disponibilização de um registador de temperatura e respetivo *software* certificado para Meteorologia Legal. A Meteorologia Legal é aplicada a todas as câmaras com uma volumetria superior a 10 metros cúbicos, onde a regulamentação indica que tem de existir um sistema certificado para o registo das temperaturas. Faz parte deste conjunto o *Software* “CapTemp SQL”, representado na figura 1.1, responsável por guardar os dados provenientes dos sensores ligados ao registador.

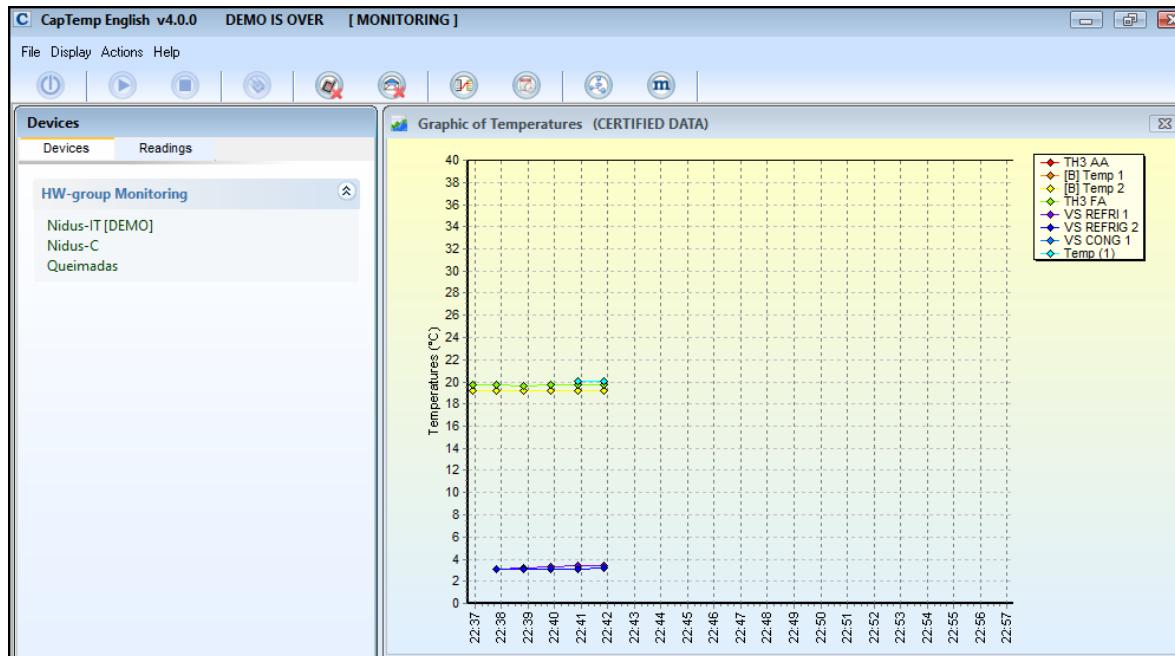


Figura 1.1: CapTemp SQL

O registador desenvolvido pela Captemp, representado na figura 1.2 denomina-se por Nidus-C, um registador que suporta até 32 sensores.



Figura 1.2: Coletor de dados Nidus-C

Com a necessidade de mais funcionalidades, a Captemp criou diversas variantes da Nidus-C, representadas na Figura 1.3 para aplicar em outras áreas para além da Meteorologia Legal. Das quais surgiram a Nidus-C+, uma versão similar da Nidus-C acrescentando a possibilidade de adicionar sensores *Wireless*. A Nidus-IT e Nidus-IT+

duas versões com as funcionalidades da Nidus-C e Nidus-C+ respetivamente, acrescentando Inputs e Outputs ao sistema de monitorização. Para soluções exclusivamente *Wireless* nasce a Nidus-W suportando apenas sensores *Wireless*. Por último é desenvolvido a Nidus-R, baseada na Nidus-IT especialmente desenhada a pensar em ambientes IT com suporte para montagem em bastidores.



Figura 1.3: Universo Nidus

No setor dos sensores foi desenvolvido o TH3, um conversor RS485 permitindo às diversas Nidus, ligar por RS485 a sensores 1Wire além dos dois inputs possuídos no TH3. Nos sensores *Wireless*, foi desenvolvido o Airo à semelhança do TH3 possui dois inputs, um ecrã e possibilita a ligação de sensores 1Wire. Permite ainda a leitura de todos os Airo adicionados na Nidus ao mesmo tempo, tecnologia desenvolvida pela Captemp denominada por Captemp AST [9]. Ambos os sensores estão representados na Figura 1.4



Figura 1.4: TH3 e Airo

Em desenvolvimento encontram-se sensores com recurso a tecnologias NB-Iot, *beacons* BLE e Lora entre outras soluções tais como sistemas de rastreamento de Febre em tempo real com recurso a inteligência artificial.

A Captemp desenvolve igualmente um portal *Cloud* denominado Senslive (Figura

1.3. MOTIVAÇÃO E OBJETIVOS

1.5) que possibilita a centralização dos sistemas de monitorização numa única plataforma *Cloud*.

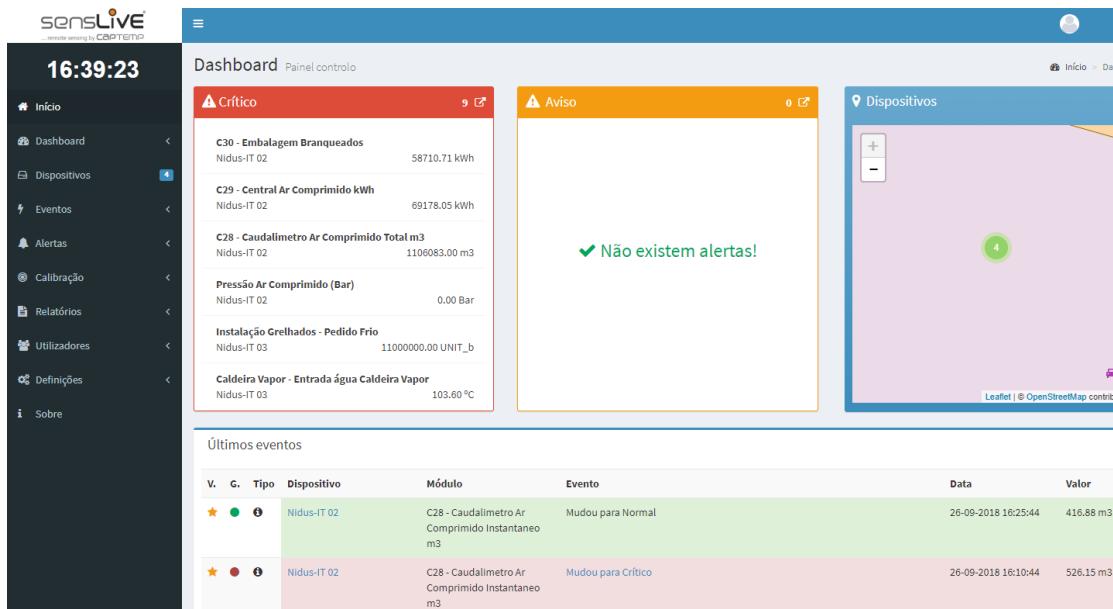


Figura 1.5: Portal Senslive

1.3 Motivação e Objetivos

O estágio é uma forma do estudante colocar numa situação de contexto profissional os conceitos adquiridos em contexto académico. A realização de um estágio é também uma mais valia pois possibilita o adquirir de experiência profissional que não é possível obter em contexto escolar.

A Captemp com os projetos a desenvolver pretende melhorar constantemente os seus equipamentos e as suas interfaces, por questões de *Marketing* quer pelo próprio evoluir da tecnologia e necessidade de novas funcionalidades e o seu desenvolvimento requer o alojamento de mais recursos como por exemplo a memória ou disco, estes limitados nestes tipos de equipamentos e onde é preciso fazer uma correta escolha de soluções e de implementação.

Ao longo do estágio, o objetivo será estudar variados mecanismos que permitam adotar recursos web modernos nos equipamentos de baixos recursos. Serão aplicados vários conhecimentos adquiridos durante o percurso académico de modo a melhorar a interface para o utilizador, técnicas de otimização de código, compressão de ficheiros, manipulação de imagens de modo a ocupar o mínimo de espaço permitindo futuros desenvolvimento e melhorias, dando continuidade ao suporte do projeto Nidus. Igualmente serão criados três novos projetos de desenvolvimento de novos equipamentos e

soluções que tiram partido de novas tecnologias como o NB-Iot e *beacons* BLE onde é necessário devido à escassez de recursos fazer a correta gestão dos mesmos e a implementação de variados algoritmos.

Nas seguintes secções são apresentadas uma breve descrição de cada projeto, dos equipamentos já existente e desenvolvido, e as funcionalidades a desenvolver em cada projeto.

1.3.1 Nidus

O projeto "Nidus" tem como objetivo dar suporte ao *Front-end* das Nidus já existentes para correções de bugs encontrados em versões anteriores, otimização de código, de modo a ocupar o mínimo espaço, possibilitando deixar memória livre para desenvolvimentos futuros, desenvolver versões customizadas com *layouts* a pedido do cliente com funcionalidades específicas, ou simplesmente melhorar a página seguindo a tendência de equipamentos concorrentes.

1.3.2 NB-Iot

Com o surgimento da nova tecnologia NB-Iot surgiu a necessidade de serem criados equipamentos que tirem partido dessa tecnologia e as suas vantagens. Para tal durante o estágio será desenvolvido um dos equipamentos que tira partido da tecnologia. Este projeto tem como por objetivo criar uma versão de raiz, simplificada e mais barata de um outro equipamento de NB-Iot em desenvolvimento pela Captemp, através do módulo Xbee da DIGI e da sua programação em Micropython. Durante o projeto será necessário garantir a correta gestão de memória, gestão de Logs internos, comunicação com os sensores físicos, comunicação bidirecional e encriptação com o portal Senslive.

1.3.3 Kea Tracker

O "Kea Tracker" é um projeto de *beacons* BLE que comunicam com o *smartphone*, onde é possível definir alertas locais no *smartphone* e envio dos dados obtidos dos sensores dos *beacons* e envio para a plataforma Senslive. Tal como o projeto anterior será necessário além de criar uma aplicação para *smartphone*, criar *Firmware* específico para os *beacons* que na ausência de comunicação com o *smartphone* devem armazenar em Log as leituras dos sensores e quando este está ao alcance descarregar para o *smartphone*.

1.3.4 dot.Tracker

A pedido de um cliente foi solicitado o desenvolvimento de uma plataforma para localização de pessoas e objetos em ambientes *inndoor*. O cliente pretende ter uma plataforma onde seja capaz de ver em tempo real a posição de pessoas e objetos definidos previamente, definir zonas de alerta, e consultar o histórico de movimentos. Neste projeto irão ser usados *beacons* BLE e vários *Gateways* BLE estrategicamente colocados no edifício e responsáveis por receber o *broadcast* dos *beacons* que por sua vez transmitem para o servidor através da rede informática do cliente do serviço. O projeto é constituído pelo desenvolvimento da plataforma de gestão e visualização, pelo recetor dos pacotes provenientes dos equipamentos e respetivos cálculos segundo o algoritmo a adotar.

1.4 Problemas identificados

Foram identificados diversos problemas em cada um dos projetos a desenvolver durante o estágio. Uma breve descrição é apresentada de seguida.

A página WEB da Nidus desde a sua criação já sofreu muitas alterações para seguir os padrões e tendências da concorrência e, portanto, está em constante atualização. Atualmente com a mundialização quase todas as pessoas sabem inglês, mas existem algumas pessoas que ou não sabem ou preferem usar a língua nativa. Para tal a Captemp pretende desenvolver uma página WEB com um sistema de tradução e diversas línguas que seja possível de alojar na memória do equipamento, devido aos problemas já referidos para o utilizador escolher a linguagem e assim cativar mais clientes e expandir a Captemp para outros países. Com o acréscimo do serviço de internalização surge o problema de uma interface com necessidade de mais armazenamento. Terão de ser estudadas otimizações que se possam implementar no código já existente. Existe a necessidade de estudar o melhor método de compressão da página mantendo o GZIP utilizado atualmente ou migrar para outro mais recente como o Brotli e a compressão de imagens migrando as imagens existentes para imagens SVG, possibilitando outras soluções para a página com sistemas mais interativos e ocupando o menor espaço disponível. Além dos problemas referidos anteriormente poderão surgir novas funcionalidades, a pedido de clientes, como por exemplo páginas com layout específicos ou novos sensores e a simples correção de possíveis Bugs encontrados.

Outro problema a resolver detetado pelo *feedback* recebido dos clientes é a complexidade para a criação de eventos, ações e reações, que controlam o Sistema Nidus. Para isso a Captemp pretende reformular a estrutura de gestão de eventos para um

sistema mais visual e atual similar ao Scratch, um *software* utilizado atualmente para ensinar a crianças as bases da programação e elas mesmos criarem alguns programas sem saber nenhuma linguagem de programação. Na figura 1.6 é apresentado um exemplo de programação usando a ferramenta Scratch, onde o utilizador com um sistema de blocos pode criar condições e eventos a despoletar consoante algumas condições.

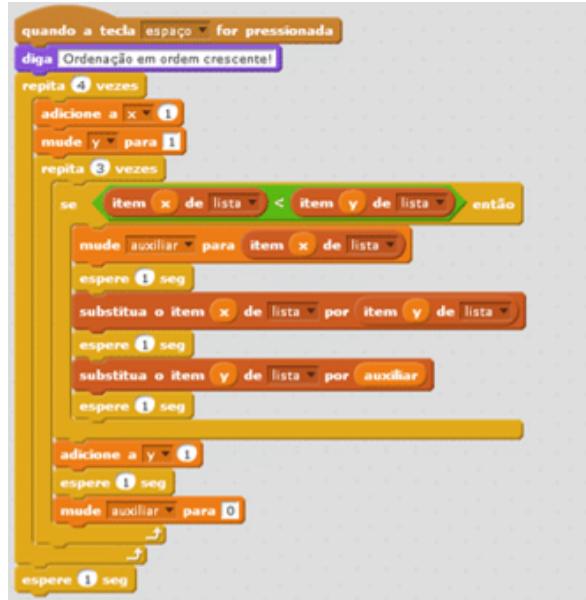


Figura 1.6: Programação com a ferramenta Scratch

Outros problemas existentes, a resolver durante o estágio, são a criação de sistemas *low-cost*, de outros equipamentos CapTemp, para clientes que não necessitem de tantas funcionalidades com a introdução da alternativa para NB-IoT com recurso ao módulo Xbee da Digi, e a substituição de produtos antigos descontinuados, os data-logger(Figura 1.7) e sua substituição por similares com as mesmas funções e mais tipos de sensores disponíveis, uma necessidade também já requisitada pelos clientes que pretendem monitorizar mais grandezas além da temperatura, mas com os padrões e tecnologias dos dias de hoje e com suporte para o novo Portal da Captemp o Senslive. Ou simplesmente o desenvolvimento de novos produtos a pedido dos clientes.



Figura 1.7: Data Logger iButton

Em resumo os problemas a solucionar durante o estágio podem ser encontrados na seguinte lista:

1.5. ORGANIZAÇÃO DO RELATÓRIO

- Melhorar a compressão da página WEB da Nidus;
- Melhorar a compressão das imagens presentes na página WEB da Nidus;
- Correção de Bugs da página Web da Nidus;
- Melhorar o processo de criação de eventos;
- Criação de uma página com sistema de tradução automático;
- Versões customizadas da página WEB a pedido do cliente;
- Seguir as tendências da concorrência;
- Criação de soluções/equipamentos de baixo custo;
- Substituição de produtos descontinuados;
- Desenvolvimento de produtos à medida do cliente.

1.5 Organização do relatório

Este presente relatório está dividido em 5 capítulos. O primeiro capítulo faz a introdução ao tema e é apresentado os objetivos, o enquadramento do estágio e alguns aspectos iniciais a considerar.

No capítulo seguinte é apresentada a tecnologia e hardware pesquisado com fim a dar suporte a este mesmo estágio e uma pequena pesquisa sobre projetos/produtos similares quer na finalidade quer nas tecnologias usadas e o estado atual de cada projeto.

O capítulo 3 apresenta o trabalho desenvolvido durante o estágio na empresa para a resolução dos problemas identificados. Este capítulo apresenta os detalhes técnicos das soluções escolhidas.

No capítulo 4 são descritos os resultados dos testes efetuados às soluções propostas e desenvolvidas no capítulo 3.

Por fim no capítulo 5 é apresentada uma breve conclusão de todo o trabalho, dificuldades e algumas sugestões para futuras implementações.

Capítulo 2

Estado da Arte

2.1 Introdução

Nesta secção é apresentado o estado da arte dos projetos realizados durante o estágio na empresa CapTemp. Nessa ordem é apresentado o funcionamento do sistema do Nidus desenvolvido pela empresa CapTemp e a sua página de configuração e visualização. Na secção 2.2.1 são apresentadas também as metodologias e tecnologias que o sistema implementa atualmente para a compressão das páginas que dão suporte ao sistema. Na secção 2.3 e 2.4 irá ser introduzido o plano inicial dos projetos a desenvolver e a base já existente tal como as tecnologias que estes irão utilizar. Na secção 2.6 será abordado as soluções e tecnologias existentes na comunidade científica e alguns produtos similares, já existentes para os projetos anteriormente referidos.

2.2 Coletor de dados - Nidus

O sistema Nidus, apesar das suas diversas versões de *hardware* partilha entre todas as versões o mesmo centro de processamento o módulo RCM6760 da Rabbit. O sistema Nidus é composto por dois módulos principais, o *Back-end* que gere toda a parte de leitura de sensores, de atuação e envio de alertas, log entre as demais funcionalidades e o *Front-end*, duas páginas WEB *Single-Application* de modo a não sobrecarregar o módulo com a interface e mover o processamento da interface para o browser do cliente. Na primeira página é possível visualizar os valores obtidos pelo *Back-end* com atualização em tempo real. Na segunda página é possível carregar as configurações para realizar alterações nas mesmas. A comunicação entre os dois componentes é feita através de XML. Para consultar os valores na primeira página o *Front-end* acede ao ficheiro "values.xml" gerado pelo *Back-end* onde contém todos os valores necessários. Na página de configurações à semelhança da primeira página os valores são carregados por um ficheiro XML o ficheiro "setup.xml", incluindo a particularidade de aceitar pedidos POST de modo a alterar as configurações do equipamento.

A Nidus dispõe de base para o utilizador variadas funcionalidades tais como, leitura

de sensores TH3 e Airo, INPUTS digitais, OUTPUTS digitais e analógico, leitura de sensores SNMP e MODBUS, envio de alertas via GSM e E-mail, programação de eventos, envio automático para um portal *Cloud* e Log Interno. Outras funcionalidades estão disponíveis mediante o pedido do cliente tais como sensores específicos, leitura de sensores por RS232 ou outros protocolos de comunicação específicos. Na tabela 2.1 são apresentadas as principais características do módulo RCM6760 da Rabbit utilizado nos equipamentos Nidus.

Tabela 2.1: Especificações do Módulo RCM6760

Microprocessor	Rabbit 6000
Frequencia do Microprocessor	200 MHz
Memória Flash	4 MB (Código e Sistema de Ficheiros)
SRAM	1 MB
Consumos	260 mA 3.3V - Ethernet ON

2.2.1 Páginas do coletor de dados Nidus

O código desenvolvido de modo a chegar á fase de produção é comprimido e compilado de modo a que ocupe o mínimo espaço e possa ser armazenado na memória do módulo e coabitar com o *Firmware* de *Back-end*, segue os seguintes passos de desenvolvimento:

1. Desenvolvimento/ alteração do código JavaScript necessário;
2. Compressão das Imagens necessárias com recurso a ferramentas online tais como o TinyPNG[10] e posterior conversão em base64 para incluir no JavaScript a imagem e o mesmo poder fazer a gestão da apresentação
3. Compilação/compressão do JavaScript num ficheiro único com recurso ao Google Clousure Platform, nesta etapa para cada versão de hardware é compilado consoante os ficheiros a incluir, poupando o espaço não necessário como o código referente aos Inputs e Outputs na Nidus C, C+ e W, ou o código referente ao módulo *Wireless* nas versões não *Wireless*.
4. Geração do minificado do código HTML
5. Compressão de cada ficheiro para o seu respetivo GZIP

Após estes passos fica disponível uma nova versão da página pronta a ser carregada na Nidus. Na imagem 2.1 é apresentado o estado e layout de uma página da Nidus IT no momento do início do estágio.

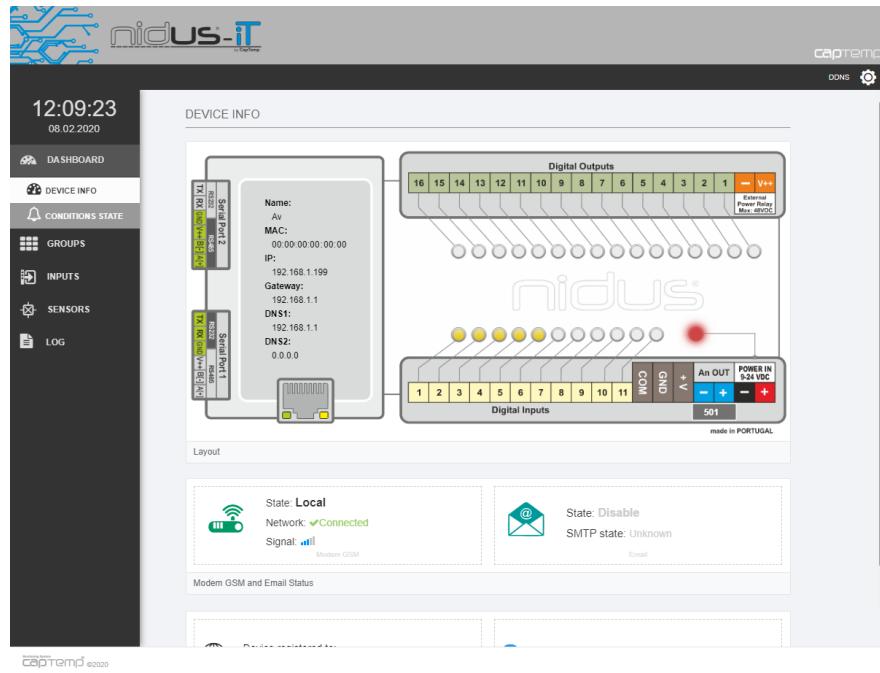


Figura 2.1: Layout página da Nidus IT no início do estágio

2.3 NB-Iot & Digi Xbee 3

Os módulos Xbee 3 representado na figura 2.2 da DIGI dispõe recentemente de uma versão NB-Iot/ LTE. Ideal para projetos com baixo volume de transmissão de dados e com baixo consumo de energia. O módulo inclui também um compilador de MicroPython, porém a versão MicroPython desenvolvida pela DIGI e incluída no módulo XBee, não inclui todas as funcionalidades do MicroPython tais como por exemplo a biblioteca de gestão de Arrays e o módulo de "_thread" pois o mesmo não tem suporte para *multithread*. Na tabela 3.2 são apresentadas as principais características do módulo XBee 3 da Digi[11].

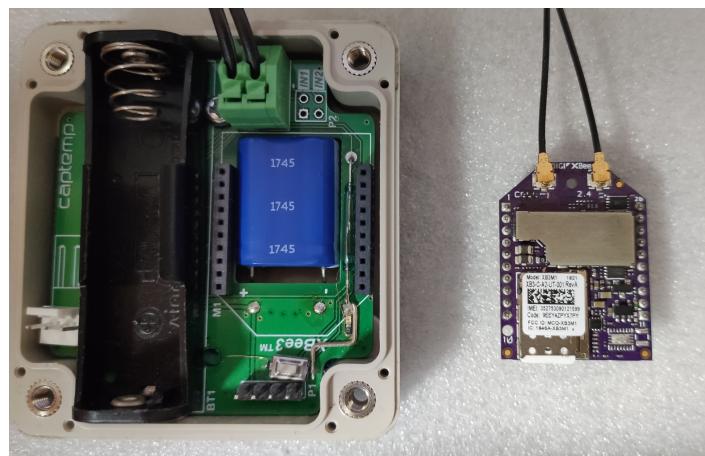


Figura 2.2: Módulo Xbee 3 e placa de expansão desenvolvida pela Captemp

Tabela 2.2: Especificações do Módulo Xbee 3

Chipset	U-blox SARA-R410M-02B
Dimensões	24.38 mm x 32.94 mm
Temperatura de Funcionamento	-40º C to +85º C
Tipo de SIM	4FF Nano
Interfaces	UART, SPI, USB
Programação MicroPython	32 KB Flash / 32 KB RAM
I/O	4 ADC (10-bit), 13 I/O digitais, USB, I2C
Bluetooth	BLE Ready
Potencia de Transmissão	Até 23 dBm
Sensibilidade de Recepção (LTE-M)	-105 dBm
Sensibilidade de Recepção (NB-IoT)	-113 dBm
Velocidade Downlink/Uplink(LTE-M)	Até 375 kb/s
Velocidade Downlink/Uplink(NB-IoT)	Até 27.2 kb/s Downlink, 62.5kb/s Uplink
Alimentação	3.3-4.3VDC
Pico corrente na transmissão	550mA - Bluetooth OFF 610mA - Bluetooth ON
Corrente média de transmissão (LTE-M)	235mA
Corrente média de transmissão (NB-IoT)	190mA
Modo Power Save	20uA
Modo Deep Sleep	10uA

A Captemp pretende, através da utilização deste módulo e de uma placa de expansão desenvolvida pela própria, apresentada anteriormente na figura 2.2, desenvolver uma versão do seu outro equipamento de Nb-Iot, mais simples representando numa opção de menor custo para o cliente. Será necessário desenvolver todo o código referente à gestão interna de Logs para guardar informação quando não existe cobertura para envio, o agendamento do envio e leituras, otimização da memória e bateria e implementação de comunicação bidirecional com encriptação com o portal Senslive. Sempre com recurso á programação em MicroPython. A placa de expansão inclui um módulo de RTC, um conversor 1Wire para possibilitar a leitura de sondas já desenvolvidas pela Captemp, um sistema de alimentação para possibilitar a alimentação por pilha ou por alimentação externa. Ao desenvolver todo o equipamento a empresa tem o controlo total sobre o Firmware e sobre a estrutura de envio e a vantagem de tornar o equipamento compatível com todos os sensores que já possui.

2.3.1 MicroPython

O MicroPython[12], lançado em 2014, é um compilador e interpretador que implementa a linguagem Python3 e otimiza o seu funcionamento em microcontroladores. Escrito em C e disponibilizado em *open-source* é possível adaptar o mesmo para os diversos equipamentos.

É suportado por diversas arquiteturas de processadores tais como:

- x86
- x86-64
- ARM
- ARM Thumb
- Xtensa

Em microcontroladores que suportem *multi-thread*, não sendo o caso do módulo usado está disponível ao programador o módulo de "`_thread`" para criar processamento paralelo. Disponibiliza a programação de interrupções físicas, úteis em microcontroladores, tem disponível um *Garbage collector* para gerir a memória do microcontrolador e bibliotecas tais como "`usocket`" para criação e gestão de sockets, "`network`" para gerir a comunicação com o módulo específico de cada microcontrolador, ou a biblioteca para gerir o módulo de bluetooth denominada por "`ubluetooth`". As bibliotecas disponíveis encontram-se no site oficial da documentação[13].

2.3.2 NB-Iot/ LTE-M

O NB-Iot ou Narrowband Iot e o LTE-M são tecnologias de *Low Power Wide Area*. São indicadas para sistemas *smart* em diversas áreas como a monitorização, a agricultura, localizadores entre outras áreas. Similar ao funcionamento da rede móvel, onde cada equipamento possui um cartão SIM e se liga á rede fornecida pelo operador, mas utilizado em equipamentos com menor transmissão de dados e que não tem acesso a fontes de alimentação fixas e requerem de baterias, o NB-Iot promete autonomias das baterias a rondar os 10 anos[14]. Devido ao baixo volume de dados o plano de dados é possível apenas com pequeno investimento obter anos e até décadas de transmissões de dados.

De entre as vantagens podem-se destacar:

- Baixo consumo

- Longo alcance e boa penetração
- Baixo custo de desenvolvimento na implementação da cobertura
- Custo reduzido pelas transmissões
- Sem necessidade de Roaming

A cobertura da rede está a ser implementada pelas operadoras de telecomunicações que já possuem cobertura da rede GSM e infraestrutura de ligação á rede Internet desenvolvida e apenas necessitam de disponibilizar cobertura nas antenas de rede móvel, normalmente já existe compatibilidade de *Hardware* e basta atualizações de *Firmware*. É aconselhado pelas operadoras que se utilize o Nb-Iot para equipamentos fixos e o LTE-M para equipamentos em movimento.

2.3.2.1 Low Power Wide Area

As redes *Low Power Wide Area*, ou simplesmente denominadas por LPWAN são redes usadas frequentemente no IOT quando é necessário enviar dados a distâncias longas. Combinam a largura de banda e o consumo de bateria presente em redes como BLE e Zigbee, com alcance igual ou superior às redes de comunicação GSM. São caracterizadas por ter longo alcance, um baixo custo de transmissão e baixo consumo, onde simples baterias podem fornecer alimentação na ordem das décadas. Este alcance pode ser conseguido por exemplo por redes *multihop* ou modulações específicas que privilegiam o consumo energético e o alcance. A comunicação 2G e 3G pode ser usada em comunicação M2M mas as mesmas tem uma largura de banda superior ao necessário o que resulta em consumo de bateria excessivo onde não é tirado proveito da largura de banda disponível. Alguns exemplos de redes LPWAN, são o DASH7, o SigFox, LoRa, Ingenu, Telensa ou o NarrowBand Iot.[15]

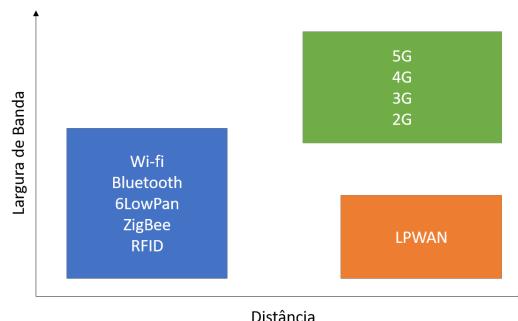


Figura 2.3: Gráfico com relação Distancia vs Largura de Banda[1]

2.4 Kea Tracker

O Projeto Kea Tracker utiliza *beacons* da Ruuvi, *beacons open-source*[16], que disponibiliza de forma *open-source* tanto o *Firmware* para alterações, como as aplicações para Android e IOS. Será desenvolvida uma aplicação baseada na aplicação fornecida e o *Firmware* para disponibilizar a funcionalidade de data-logger.

2.4.1 Beacons BLE

O *Bluetooth Low Energy* ou simplesmente BLE foi desenvolvido a pensar nos novos equipamentos IOT, onde os utilizadores querem vários equipamentos ligado ao mesmo tempo. Para tal foi desenvolvido o BLE que permite mais ligações ao mesmo tempo comparando com o *Bluetooth* clássico. Como é indicado no nome, o principal fator diferenciador nesta versão, utilizada muitas vezes em equipamentos IOT, é o baixo consumo de aproximadamente metade relativamente ao *Bluetooth* normal. Outras características melhoradas a visar os equipamentos de IOT no BLE são a baixa largura de banda e o baixo tempo de transmissão.

Com o desenvolver do BLE foram criados, novos tipos de equipamentos, nomeadamente os beacons, equipamentos quase sempre alimentados por pilhas, que comunicam através de BLE, tornando o equipamento portátil. Os beacons são caracterizadas por transmitir pequenas quantidades de informação em *broadcast*. Existem dois tipos de beacons, os beacons não conectáveis e as conectáveis[2]. Como indicado no nome os beacons conectáveis permitem que um equipamento (como um *smartphone*) se conecte ao *beacons* e esta fica preparada para receber dados. As não conectáveis apenas permitem o *broadcast* dos dados, poupando energia pois apenas é necessário ter o módulo acordado para fazer o *broadcast* e o restante do tempo podem estar num estado *sleep*. Na figura 2.4 é apresentado o pacote que é transmitido em *broadcast* para os outros equipamentos ao alcance.

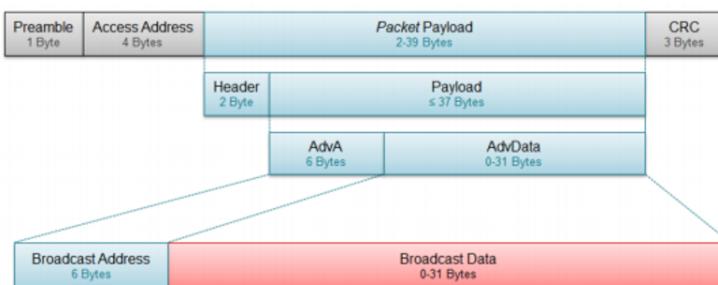


Figura 2.4: *BLE broadcast packet*[2]

2.4.2 Ruuvi *beacons*

Neste projeto o *Firmware* dos *beacons* necessita de uma alteração, tornar o *beacon* num *beacon* conectável e esta armazenar internamente as últimas leituras num *buffer* circular e criar um data-logger e caso o cliente pretenda poderá conectar mais tarde para fazer o *download* para aplicação e posterior envio para o Senslive, não necessitando a proximidade do *smartphone* ao *beacon* durante todo o tempo. A Ruuvi dispõe de dois modos de desenvolvimento de *Firmware* do *beacon* em C ou usando o Espruino, à semelhança do MicroPython um interpretador de JavaScript para microcontroladores lançado em 2012, totalmente compatível com os *beacons* da Ruuvi.

2.4.3 Apps smartphones

Na fase inicial será adaptada a versão disponibilizada para Android para agilizar a integração com o portal Senslive. Na segunda fase deste projeto será desenvolvida uma versão de raiz apenas com as funcionalidades necessárias. A aplicação base para android disponibilizada pela Ruuvi foi desenvolvida em Kotlin[17], uma linguagem desenvolvida pela JetBrains multiplataforma e que inclui o Android nessas plataformas compatíveis. De seguida estão apresentadas as alterações necessárias na aplicação já existente:

- Alteração das imagens e logotipo da app;
- Alteração do nome da App;
- Remoção de conteúdo não necessário;
- Bloqueio do URL de envio para usar exclusivamente o portal Senslive;
- Melhoramento da precisão da posição GPS;
- Possibilidade da alteração dos intervalos de registo
- Leitura dos registos dos equipamentos

2.5 dot.Tracker

O Projeto dot.Tracker é um desenvolvimento de um projeto a pedido do cliente que pretende através de uma plataforma WEB monitorar a posição de objetos e de pessoas em tempo real. Neste projeto serão utilizados *beacons* BLE e *Gateways* responsáveis por receber os pacotes dos *beacons* e os transmitir para um servidor. O servidor será capaz de processar os pacotes e calcular a posição do *beacon* no mapa e informar

os utilizadores sobre alertas e estatísticas. Visto este projeto ter sido solicitado por um cliente que pretendia uma solução à medida e este não se encontrar no espaço Europeu não há problemas de registo e armazenamento do *Tracking* das pessoas devido á legislação em vigor na Europa para proteção de dados, o RGDP.

À semelhança do projeto Kea Tracker o projeto dot.Tracker usa igualmente *beacons* BLE para enviar a informação necessária para o respetivo portal. É necessário recolher os pacotes recebidos dos *beacons* enviá-los para o servidor e calcular a distância entre o *beacon* e o recetor e com o auxilio de múltiplos recetores realizar a triangulação do *beacon* num mapa. No decorrer do projeto será necessário desenvolver uma plataforma WEB para receber e visualizar as localizações provenientes dos *beacons* e respetivas configurações, adotar o método de algoritmo para a triangulação do *beacon* relativamente a vários recetores e realizar testes ao funcionamento e precisão do sistema.

2.5.1 Beacons e Gateway

Para este projeto irá ser utilizado durante o desenvolvimento a solução da Beacon Line[18] e outras a pesquisar de modo a procurar as melhores soluções e posteriormente desenvolvido recetores proprietários da Captemp. A solução apresentada pela Beacon Line, é composta por um *Gateway* e vários nós. Cada nó possui um recetor BLE e quando o mesmo recebe um *broadcast* proveniente do *beacon* o transmite para o *Gateway*. Caso exista alguma divergência da potência de transmissão desde o último pacote enviado por esse mesmo *beacon* o gateway com conectividade Ethernet realiza o *Publish* num broker onde é possível o servidor obter os pacotes dos *beacon*. As restantes soluções baseiam-se em equipamentos com uma interface BLE e outra Wi-Fi e comunicam diretamente com o servidor ao invés de comunicarem com um ponto central como acontece no caso da Beacon Line.

2.6 Soluções e tecnologias disponíveis

Neste subcapítulo é abordado algumas das abordagens possíveis para atingir os objetivos nas secções anteriores (Secção 2.6.1), assim como a aplicação dos mesmos em produtos existentes (Secção 2.6.2).

2.6.1 Tecnologias disponíveis

Existem diversas abordagens possíveis para a compressão de ficheiros de código, de imagens e localização *indoor*. Algumas destas são descritas de seguida.

2.6.1.1 Compressão de ficheiros

Atualmente a vida *online* do Homem passou a ter um grande impacto na sua vida. Para tal as páginas WEB e seus conteúdos foram aumentado em quantidade e tamanho e com menores tempos de resposta. Isso é aplicável tanto aos ficheiros que contem o layout da página, quer das imagens. Para poupar dados de transmissão e reduzir tempos de envios, ou simplesmente suportar larguras de banda inferiores, os *browsers* integraram a possibilidade de receber os ficheiros comprimidos e fazer a descompressão para mostrar ao cliente quase em tempo real. Atualmente os *browsers* recentes suportam a compressão por GZIP(já utilizado na página do equipamento Nidus) e compressão utilizado a codificação Brotli [19] [20]. Cada método de compressão possui as suas vantagens e desvantagens, o brotli por sua vez á semelhança de outros métodos em comparação com o GZIP, tem uma taxa de compressão superior[21], isto significa que consegue reduzir o mesmo ficheiro no seu respetivo ficheiro comprimido ocupando menos espaço em relação ao GZIP, mas como desvantagem o tempo de compressão do mesmo é superior. Ao contrário da compressão, na descompressão o Brotli tem melhores resultados do que nas restantes alternativas apresentando velocidades superiores de descompressão.

O GZIP e o brotli usam na sua compressão para reduzir o tamanho do ficheiro o algoritmo de compressão LZ77, que procura sequências repetidas utilizando o método de janela deslizante e substitui essas sequências por referências para a primeira ocorrência que não foi substituída indicando a distância a que a primeira ocorrência ocorre e o tamanho a substituir.

O sistema de janela deslizante define um tamanho da janela e ao deslocar a janela do tamanho definido define um dicionário. Após definir o dicionário com vários tamanhos de janelas, percorrer novamente o ficheiro através do método de janela deslizante novamente a procurar repetições das entradas que existem no dicionário. Quando uma sequência é encontrada esta é substituída por uma referência da posição da primeira ocorrência da mesma. Na figura 2.5 é apresentado um exemplo do funcionamento da janela deslizante para a obtenção do dicionário com o tamanho da janela a variar de 2 a 7.

Na figura 2.7 e 2.8 é apresentado dois exemplos visuais e simples utilizando frases de como o LZ77, usado pelo GZIP e Brotl através do sistema de janela deslizante procura as repetições e comprime os ficheiros. Na Figura 2.6 é apresentado o ficheiro base, representado por um pequeno texto. No exemplo apresentado pela figura 2.7 apenas foi utilizado a substituição de palavras inteiras, na figura 2.8 procura sequências de caracteres sejam elas palavras ou não. Nos exemplos apresentados a redução foi de 20% $[1 - \frac{48}{60} \times 100\%]$ no primeiro exemplo e de aproximadamente 32% $[1 - \frac{41}{60} \times 100\%]$



Figura 2.5: Funcionamento da Janela Deslizante

no segundo.

A	M	A	R	I	A	F	O	I	A	P	R	A	I	A .
A	P	R	A	I	A	E	S	T	A	V	A	V	A Z	I A
S	O	L	A	E	S	T	A	V	A	A	M	A R	I A	.

Figura 2.6: Sequência não comprimida

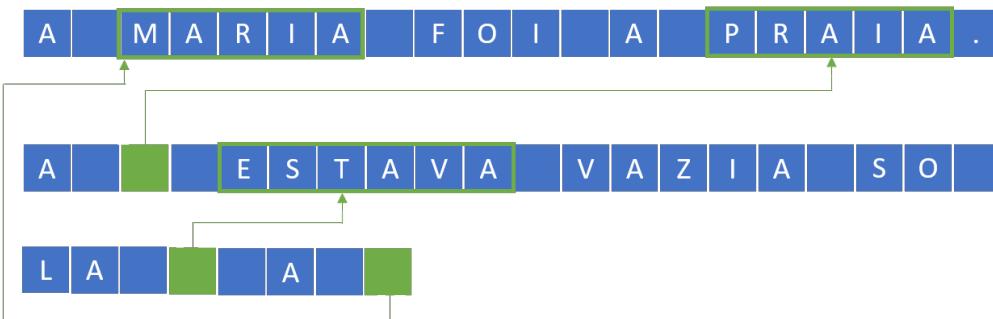


Figura 2.7: Sequência comprimida com LZ77 (apenas palavras)

2.6.1.2 Compressão de imagens

O utilizador pretende igualmente ver as imagens com a máxima qualidade, mas qualidade significa um maior detalhe e por sequencia um ficheiro de maior tamanho. Existem atualmente vários *softwares online* e locais que reduzem o tamanho das imagens. Na conceção da página da Nidus é utilizado o website TinyPNG.com que analisa a imagem

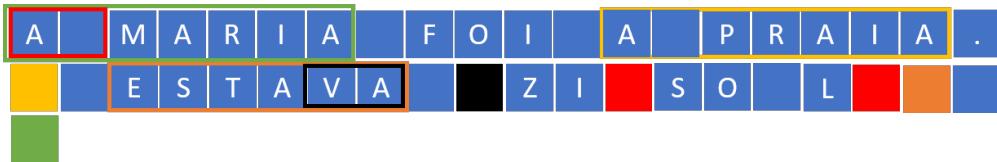


Figura 2.8: Sequência comprimida com LZ77(palavras e sequências)

original e converte as cores em cores mais simples de o sistema armazenar, como por exemplo uma imagem com 24 bits de profundidade de cor, onde cada cor é representada por 24 bits pode ser convertido em uma similar com apenas 8 bits reduzindo o tamanho do ficheiro e imperceptível para o olho humano num ecrã digital[22]. Alternativamente ao Tiny Png existem *softwares*, similares alguns de licença GNU/GPL, para compressir imagens. Com o "Mass Image Compressor"[23](apenas um exemplo), é possível compressir as imagens com a possibilidade de indicar a quantidade de compressão.

Com a enorme quantidade e diversidade de monitores existentes, as páginas web necessitam de ser responsivas e apresentar a melhor imagem para o monitor em questão, isso normalmente traduz-se em várias versões similares da imagem alojadas no servidor. No caso dos microcontroladores e sistemas embedidos o espaço encontra-se limitado e deve-se arranjar uma solução. Uma solução possível é ao invés da utilização de imagens PNG, JPG ou outras, é a utilização de imagens em SVG, onde a imagem é representada por um ficheiro XML que descreve uma imagem bidimensional e utiliza na sua constituição modelos matemáticos para o cálculo das posições dos elementos. Com isto é possível manipular o XML em tempo real para alterar elementos ou remover, alterar cores, criar animações entre outras. Inclui a vantagem de como a imagem é representada por formulas matemáticas, é possível escalar a imagem sem perder qualidade, pois a função matemática é ajustável. Num sistema embedido como o caso da Nidus é vantajoso a utilização de imagens em SVG para criação das animações. Atualmente as animações da página da Nidus são criadas com várias imagens PNG comprimidas e convertidas em base64 e são alternadas no HTML pelo JavaScript. Com a utilização de imagens SVG é possível ter apenas uma imagem alojada e manipular a imagem em tempo real através do JavaScript de uma forma mais suave para o utilizador, pois apenas a zona a alterar é alterada na imagem. À semelhança dos JPG e PNG o SVG também pode ser comprimido, para tal basta no XML da imagem remover os meta-dados e utilização de funções matemáticas mais simples, não necessários para o browser apresentar a representação gráfica do mesmo, mas os *softwares* de edição adicionam para funcionalidades exclusivas do editor. À semelhança dos ficheiros HTML após a remoção dos meta-dados o ficheiro pode ser minificado.

2.6.1.3 Localização *indoor*

É possível encontrar na comunidade científica vários estudos sobre a utilização de redes Wi-Fi e Bluetooth para sistemas de localização. Estes mesmos focam-se no cálculo das distâncias do equipamento para vários receptores no mesmo intervalo temporal, algumas destas soluções baseiam-se nos valores de RSSI da transmissão e o valor definido como constante da potência de transmissão à distância de 1 metro, e estimar a sua distância aproximada de cada receptor, com essas aproximações é possível através do algoritmo escolhido[24], obter a estimativa da localização do equipamento e a sua colocação num mapa. A distância de um receptor para o emissor baseada no valor de RSSI é expressa pela seguinte fórmula, onde dbm é a constante da potência de transmissão da *beacon* a 1 metro, n a constante do ambiente e o RSSI corresponde ao RSSI da transmissão:

$$d = 10^{\left(\frac{dbm - RSSI}{10 \times n}\right)}$$

Após a obtenção da distância para cada receptor é possível aplicar um algoritmo para estimar a localização. Os mais referenciados e adotados são o *centroid* baseado no centro geométrico do polígono formado pelas intersecções das circunferências criadas com o raio da distância calculada pela fórmula anteriormente apresentada, o método *Three-border Positioning* e o *Least Square Estimation*. Como é possível observar na figura 2.9 utilizando o método *centroid*, o centro geométrico corresponde à localização do equipamento com base nos receptores. A fórmula que representa o centro utilizando o *centroid* é expressa pela seguinte equação onde n representa o número de receptores utilizados no cálculo.

$$(x, y) = \left(\frac{x_1 + x_2 + x_3 + \dots + x_n}{n}, \frac{y_1 + y_2 + y_3 + \dots + y_n}{n} \right)$$

Ao invés da utilização do método do *centroid* se for adotado o método *Three-border Positioning*, é criada a função definida por ramos composta pelas três equações da circunferência A, B e C com os respetivos centros em cada receptor e com o raio igual à distância calculada para esse mesmo receptor. Para calcular a posição estimada é calculado o resultado dessa mesma função de modo a encontrar o ponto x,y que representa a posição do equipamento.

Utilizando o método *Least Square Estimation* ou simplesmente LSE e à semelhança do Three-border Position[25] é criada a função de ramos das equações das circunferências dos vários receptores com o raio da distância calculada, mas pode igualmente como o *centroid* utilizar mais do que três receptores aumentando a precisão.

Hua, Z., Hang, L., Yue, L., Hang, L., & Kan, Z. (2014). Geometrical constrained least squares estimation in wireless location systems. 2014 4th IEEE International

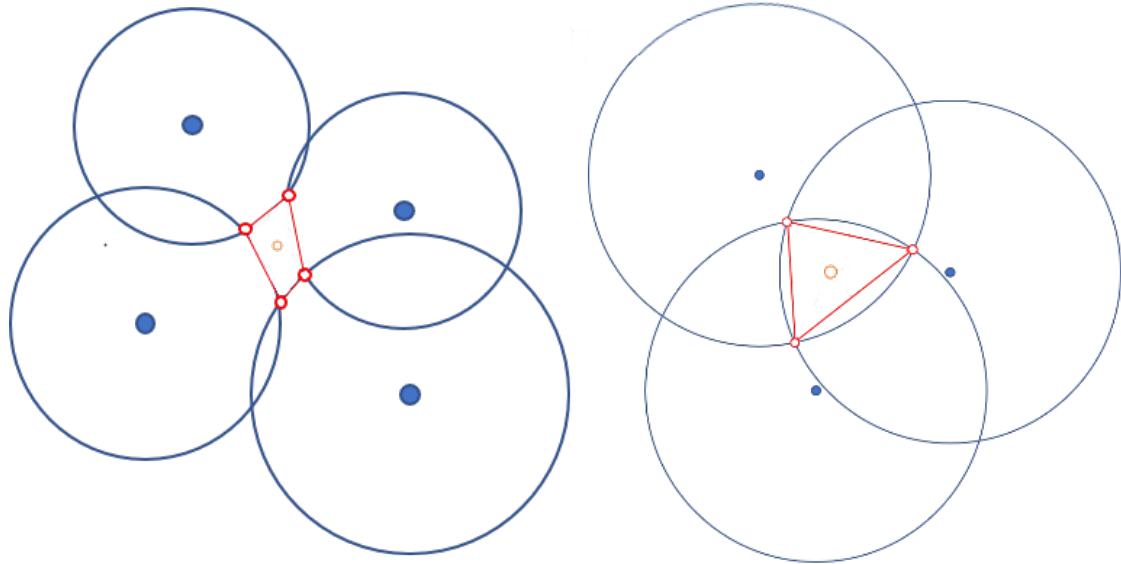


Figura 2.9: Posição utilizando o método *Centroid* com 3 e 4 receptores

Conference on Network Infrastructure and Digital Content. apresenta os passos necessários calcular a posição X do equipamento através do método LSE. Em primeiro são criadas a função de ramos composta pelas equações das circunferências com centro nos receptores ($[x_1, y_1], [x_2, y_2], [x_3, y_3], [x_4, y_4]$) e o raio igual á distância calculada(d_1, d_2, d_3, d_4).

$$\begin{cases} (x_1 - x)^2 + (y_1 - y)^2 = d_1^2 \\ (x_2 - x)^2 + (y_2 - y)^2 = d_2^2 \\ (x_3 - x)^2 + (y_3 - y)^2 = d_3^2 \\ (x_4 - x)^2 + (y_4 - y)^2 = d_4^2 \end{cases}$$

Após a criação da função é subtraído o primeiro ramo aos restantes ramos e a função reduz o numero de ramos para $n-1$ onde n representa o numero de receptores a usar na função.

$$\begin{cases} 2(x_2 - x_1)x + 2(y_2 - y_1)y = x_2^2 - x_1^2 + y_2^2 - y_1^2 + d_2^2 + d_1^2 \\ 2(x_3 - x_1)x + 2(y_3 - y_1)y = x_3^2 - x_1^2 + y_3^2 - y_1^2 + d_3^2 + d_1^2 \\ 2(x_4 - x_1)x + 2(y_4 - y_1)y = x_4^2 - x_1^2 + y_4^2 - y_1^2 + d_4^2 + d_1^2 \end{cases}$$

A função pode ser representada pelo seu equivalente numa representação de matrizes por $2AX = b$ onde.

$$A = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ x_4 - x_1 & y_4 - y_1 \end{bmatrix}$$

$$B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} x_2^2 - x_1^2 + y_2^2 - y_1^2 - d_2^2 + d_1^2 \\ x_3^2 - x_1^2 + y_3^2 - y_1^2 - d_3^2 + d_1^2 \\ x_4^2 - x_1^2 + y_4^2 - y_1^2 - d_4^2 + d_1^2 \end{bmatrix}$$

A posição estimada do equipamento representada no exemplo por X é definida por:

$$X = \frac{1}{2}(A^T A)^{-1} A^T b$$

Os testes analisados demonstram[24] , que o método LSE é o método que obtém os melhores resultados com os valores mais próximos do real. No teste apresentado em segundo lugar está o *Three-border Position* e por último o *centroid*. Com algumas discrepâncias em algumas das amostragens.

2.6.2 Produtos similares

Neste subcapítulo é abordado alguns produtos já existentes no mercado para os projetos NB-IOT e Kea Tracker

2.6.2.1 NB-Iot

Atualmente no mercado começam a surgir alguns produtos similares ao que se pretende desenvolver como é o caso dos sensores da Efento[26], que disponibiliza vários tipos de sensores que comunicam por NB-Iot. A Efento é uma empresa fundada em 2014 e é focada em desenvolvimento de equipamentos IOT. Atualmente desenvolveram versões com suporte para NB-Iot. Estes equipamentos tem a desvantagem de não ser compatível com o pacote de envio desenvolvido no portal Senslive e apenas permite o envio para o portal da Efento e não existe a possibilidade da utilização das sondas já comercializadas pela Captemp. Como vantagem á semelhança do equipamento a desenvolver é a utilização de um sistema com Log para quando não existe possibilidade de comunicação. Devido ao desenvolvimento da tecnologia ainda existem poucas soluções em comercialização, estando as mesmas em desenvolvimento. A Captemp possui igualmente outro equipamento, completamente desenvolvido pela empresa, em desenvolvimento que tira partido do NB-Iot com o acréscimo em relação ao que se pretende desenvolver durante o estágio, a possibilidade de ter mais sensores, maior capacidade de Log interno, configuração por Bluetooth, GPS e um Display integrado como extras.

2.6.2.2 Kea Tracker

Após pesquisas *online* é possível encontrar algumas soluções de *beacons* que permitem o armazenamento interno de leituras para desenvolver um sistema de data-logger tais

2.6. SOLUÇÕES E TECNOLOGIAS DISPONÍVEIS

como o *beacon* da Fujitsu, a FWM8BLZ02A-109069[27] , á semelhança do *beacon* da Ruuvi usa o mesmo chip o nRF52832 da Nordic Semiconductor, mas apresenta como vantagens a inclusão de um sistema de Logs interno com capacidade para aproximadamente 4080 leituras e a diversidade de sensores já incluídos. Como desvantagem em relação á *beacon* da Ruuvi tem a inclusão de um sensor de temperatura ao invés de temperatura e humidade, não possui sensor de pressão atmosférica e não é open-source possuindo um firmware fechado. A vantagem de se desenvolver um produto desde a sua raiz é a possibilidade de ter o controlo total sobre a solução para posteriores melhoramentos e ter a solução a desempenhar apenas o que pretendemos.

Outra solução existente no mercado é igualmente a solução da Blue Maestro que possui variadas versões de *beacons*. Á semelhança do *beacon* da Fujitsu possuem igualmente sistema de Log. Contrariamente á FWM8BLZ02A-109069 é um *beacon* que tem disponível em *open-source* uma API e um SDK para desenvolver as nossas aplicações. Comparada com o *beacon* da Ruuvi, a Ruuvi é completamente *open-source* e não apenas a API para comunicação.

Na tabela 2.3 são apresentadas as diferenças e semelhanças entre os três modelos analisados

Tabela 2.3: Comparação entre *beacons* [6][7][8]

	Ruuvi Tag	Fujitsu <i>beacon</i>	Blue Maestro
Processador	nRF52832	nRF52832	?
Memória	512kB Flash 64kB RAM	32K Não volátil	?
Protocolos	Bluetooth 5 Wirepass Mira OS QUUPA Others (2.4GHz)	Bluetooth 4.1	BLE 4.2
Potência de Transmissão	+4 dBm	-16, -12, -8 -4, 0, +4 dBm	-4, 0, +4 dBm
Sensores	Acelerometro Temperatura Humidade Pressão	Acelerómetro Temperatura	Temperatura Humidade Pressão
NFC	✓	-	-
Bateria	CR2477 1000mA·h - Li/MnO ₂	CR2450	CR2032
Autonomia (espetável)	10 Anos	1 Ano em Broadcast	1 Ano em Broadcast 2 Anos com Log
Data Logger	- (a desenvolver)	✓	✓
Open Source	✓	-	✓(API & SDK)
Informações	IP67 2 Botões 2 Leds 52mm Ø	Led 40 x 31 x 12mm	24000 Registos 33mm Ø

Capítulo 3

Trabalho Desenvolvido

3.1 Introdução

Nesta secção é apresentado o trabalho desenvolvido em cada um dos projectos realizados durante o estágio do Mestrado. À semelhança do capítulo anterior, o desenvolvimento de cada projecto é descrito num subcapítulo dedicado.

3.2 Coletor de dados - Nidus

Durante o estágio o projeto referente ao coletor de dados Nidus apresenta 5 subsecções a tratar: 1) o desenvolvimento do sistema de tradução para a página do equipamento no menor espaço de memória possível; 2) a compressão dos ficheiros; 3) compressão de imagens de modo a melhorar/manter a página com layouts e funcionalidades de acordo com a concorrência e atualidade; 4) o desenvolvimento de páginas com layouts específicos; 5) a correção de *Bug's* que possam existir e venham a ser descobertos. Cada um dessas subsecções serão abordadas em cada subsecção seguinte.

3.2.1 Sistema de tradução automática

Os sistemas de internacionalização das páginas web, fornecem ao utilizador final um sistema com tradução automática para a língua pretendida, são cada vez mais utilizados. Isto acarreta um acréscimo da complexidade do sistema e por consequência o acréscimo do espaço ocupado pelo código. O sistema utilizado no caso particular das páginas web com JavaScript é o I18N, uma *Framework* desenvolvida em JavaScript, com várias funcionalidades além da tradução de páginas, tais como por exemplo a plurarização das traduções, carregamento dinâmico dos dicionários por ajax entre outros. Estas funcionalidades extras não necessárias para o projeto apenas acarretam o aumento do peso do plugin no sistema, diminuindo possibilidades futuras de alterações e novos desenvolvimentos. Para tal será desenvolvido de raiz um sistema similar ao I18N apenas com as funcionalidades pretendidas de modo a ser possível comparar a diferença de espaço ocupado.

3.2.1.1 O funcionamento

Os sistemas de tradução são baseados numa função chamada no momento necessário da obtenção de uma tradução onde é passado um parâmetro indicando qual a tradução pretendida. Esta função é responsável por percorrer o conjunto de traduções da língua selecionada e através de um sistema associativo chave-valor retornar caso este exista o valor para a chave fornecida. Caso este não exista é devolvido o valor *default*, por norma a chave do mesmo. Na figura 3.1 é apresentado o esquema do funcionamento do sistema de tradução descrito acima.

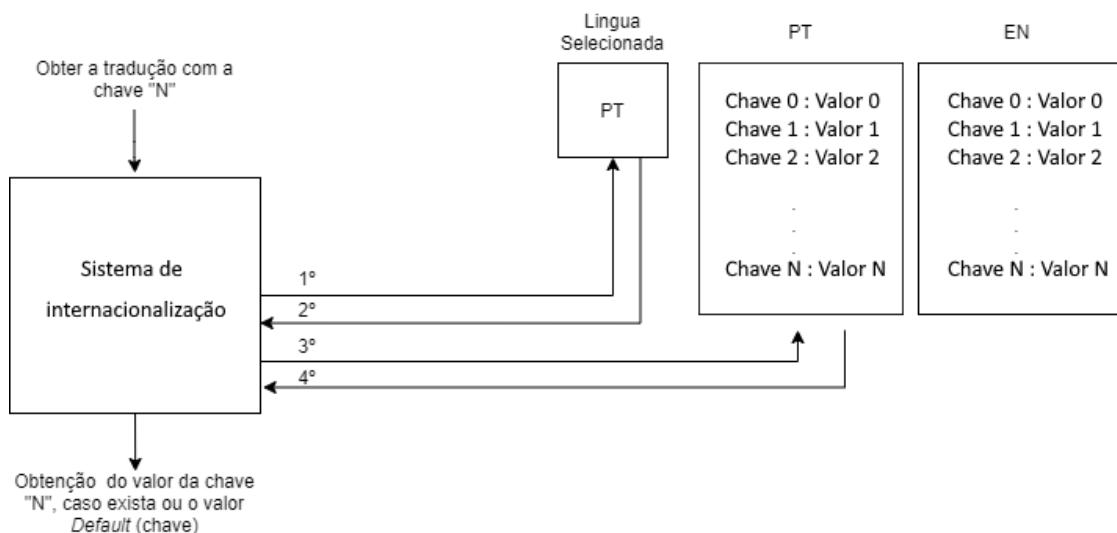


Figura 3.1: Processo da obtenção da tradução de um valor

Após análise as funcionalidades requeridas para o projeto Nidus, são as seguintes:

- Tradução automática da página
- Armazenamento persistente da linguagem escolhida
- Alteração da linguagem pretendida, mediante a lista de opções
- Carregamento das linguagens dinamicamente.

De modo a armazenar a linguagem selecionada pelo utilizador, esta estará alocada no browser sob a forma de uma *cookie*. No momento de uma tradução, o sistema verifica o valor da *cookie* em vigor e procura na associação chave-valor respetiva à linguagem selecionada, se a chave pretendida existe e caso exista o valor desta é devolvida. Caso a *cookie* não exista é devolvida a opção *default*.

O Sistema é composto por 4 funções. A primeira, gLng, responsável por verificar se existe a *cookie* no browser e retornar o seu valor ou o valor da língua *default*, o Inglês ("EN"). A segunda função, sLng, é utilizada quando do guardar da linguagem

para posterior utilização, esta como parâmetros recebe o valor a inserir, esta *cookie* fica disponível no *browser* durante 365 dias após a sua última atualização. Após guardar a *cookie*, a página é recarregada de modo a atualizar todos os campos e não apenas os gerados a partir do momento da seleção da nova língua, garantindo assim que a página tenha várias línguas no mesmo instante de tempo visto que o código HTML da mesma é gerado no momento em que é necessário pelo JavaScript e é incorporado no HTML principal. A terceira função "`_`" é responsável por retornar o valor da chave fornecida no único parâmetro da mesma. Caso a chave não exista na língua selecionada pela *cookie* é retornado o valor de *default*, correspondente à chave fornecida como parâmetro da função. Por último existe a função "`ll`" com dois parâmetros, como primeiro parâmetro temos a *string* correspondente às iniciais da língua a adicionar (Exemplo: "PT", "EN", "ES", "FR") e o segundo parâmetro um objeto com associações chave-valor para a língua fornecida. Esta função é responsável por adicionar ao conjunto de dicionários a língua pretendida ou atualizar o dicionário da língua caso este já exista.

O conjunto de dicionários armazenado segue o esquema apresentado na figura 3.2. No caso de apenas se pretender desenvolver numa língua é possível apenas fazer a definição das funções e não fornecer nenhum dicionário, visto nesse caso ele devolver a chave, bastando para isso a chave corresponder à tradução pretendida na única língua disponível, deixando o produto já preparado para futura implementação do sistema de internacionalização.

```
{
  "EN": {
    "key": "value",
    "key1": "value1"
  },
  "PT": {
    "key": "value"
  }
}
```

Figura 3.2: Estrutura JSON do conjunto de dicionários

No excerto 3.1 é apresentado o código referente às funções criadas para o sistema.

```

1
2 var _d={}; //Dicionarios
3
4 function gLng() {
5   var v = document.cookie.match('(^|;) ?locale=([^;]*)(;|$)');
6   return v ? v[2].toUpperCase(): "EN";
7 }
8
```

```

9  function sLng(v) {
10    var d = new Date;
11    d.setTime(d.getTime() + 31536000000);
12    document.cookie = "locale=" + v + ";path=/;expires=" + d.
13    toGMTString();
14    location.reload();
15
16  function _(k){
17    if(_d==null|| _d==undefined || cd==undefined || cd==null || _d[
18      gLng()]==null || _d[gLng()]==undefined) {return k;}
19    return _d[gLng()][k];
20
21  function ll(a,b){
22    if(_d==null|| _d==undefined){ _d={};}
23    _d[a]=b;
24 }

```

Algoritmo 3.1: Definição do Sistema de Internacionalização

3.2.1.2 Resultados obtidos

Após comparação do sistema implementado em comparação com o I18N[28] é possível observar a redução do tamanho para cerca de 34 % ($1178b - 66\% = 404b$), este valor foi obtido na comparação entre as duas versões minificadas e comprimidas com o GZIP como é possível observar no gráfico apresentado na figura 3.3, no gráfico é possível verificar os tamanhos originais, após a minificação e o tamanho do minificado após a compressão com o GZIP. Ambos valores relativos à comparação não contemplam os dicionários, apenas o código inerente à utilização das funcionalidades.

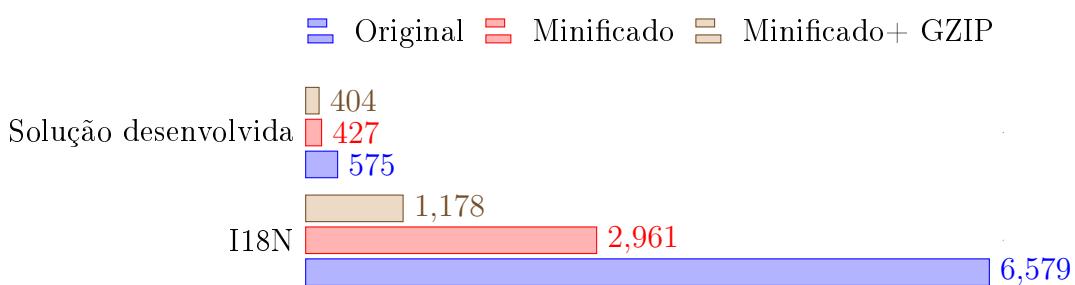


Figura 3.3: Gráfico com comparação do tamanho entre versões (em Bytes)

3.2.2 Compressão de ficheiros

Durante o estágio foi estudado igualmente a comparação entre a utilização do GZIP e do Brotli de modo a analisar as vantagens e desvantagens aplicadas ao projeto em questão.

Segundo estudos online [29], é possível analisar que comparando apenas o Brotli e o GZIP, que este último tem uma taxa de compressão inferior, significa isto que o mesmo ficheiro após compressão é maior no caso do GZIP, tornando o Brotli um candidato a ponderar. Já no campo da velocidade de descompressão o caso inverte, sendo o GZIP a possuir melhores resultados, este parâmetro, não afeta o espaço ocupado na memória do microcontrolador. A velocidade de compressão é inferior no Brotli não o tornando ideal para compressões em tempo real, mas visto o servidor WEB alojado na Nidus já possuir todos os ficheiros comprimidos e estes são sempre estáticos, o tempo de compressão não afeta o equipamento neste projeto.

Visto o método Brotli possuir mais validas ao projeto, de modo a estudar e comparar as vantagens/desvantagens inerentes à migração para o Brotli, serão comprimidos os ficheiros atuais do projeto em ambos os métodos de modo a analisar os ganhos obtidos no projeto em específico antes da sua atualização. Na tabela 3.1 são apresentados os resultados obtidos para cada ficheiro do projeto, os valores apresentados correspondem ao tamanho em bytes. No valor apresentado correspondente ao tamanho original, este no caso do HTML e CSS encontra-se minificado, caso seja JavaScript este encontra-se minificado e comprimido com recurso ao Google Clousure Compiler.

Analizando os ganhos obtidos com a utilização do Brotli é possível libertar cerca de 57 KB alterando o método de compressão dos ficheiros. De modo a realizar a migração entre a utilização do GZIP e Brotli apenas é necessário alterar o *software* desenvolvido responsável por comprimir os ficheiros originais na sua respetiva compressão de forma autónoma e a alteração dos cabeçalhos enviados pelo servidor aquando de uma resposta por parte deste para o cliente. Na figura 3.4 é possível observar a diferença presente nos cabeçalhos da resposta HTTP proveniente do *browser*.

Estas fases não serão realizadas durante o estágio referente deste relatório.

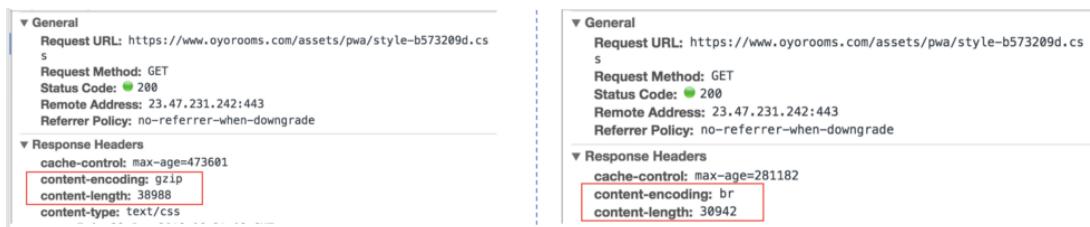


Figura 3.4: Comparação entre os *headers* HTTP GZIP e Brotli[3]

Tabela 3.1: Comparação entre Brotli e GZIP

Tipo do Ficheiro	Tamanho original Minificado	Tamanho GZIP	Tamanho Brotli	Diferença
JavaScript	230 304	58 061	52 453	-5 608
JavaScript	462 316	95 196	75 702	-19 494
JavaScript	627 032	221 754	206 150	-15 604
JavaScript	140 389	46 945	42 687	-4 258
JavaScript	84 249	28 579	26 594	-1 985
HTML	37 302	16 728	15 893	-835
HTML	39 352	17 251	16 348	-903
FONT (.TTF)	111 368	66 831	62 819	-4 012
CSS	53 642	6 815	5 956	-859
CSS	100 495	15 524	12 826	-2 698
CSS	49 145	5 674	4 771	-903
FAVICON	1150	352	348	-4
Total	1 936 744	579 710	522 547	-57 163

3.2.3 Compressão de imagens

As imagens são um ponto importante dos sistemas IOT, onde o utilizador através de esquemas, imagens e gráficos consegue obter o que pretende sem muito esforço. O sistema Nidus apenas despõe de imagens em algumas versões a pedido de clientes devido ao sobre carregamento de espaço que acarreta a utilização de imagens.

Para desenvolver uma interface com recurso a sistemas visuais é necessário ter ou uma imagem com uma qualidade estipulada para o ecrã máximo necessário para ser possível reduzir a mesma em ecrãs mais pequenos, ou existirem múltiplas imagens de vários tamanhos para cada tipo de ecrã. A utilização de imagens pequenas que ocupem pouco espaço em disco, aquando da utilização em ecrãs maiores irá deixar visível na imagem cada pixel revelando ao utilizador a fraca qualidade do sistema. Na utilização de imagens superior ao máximo dos ecrãs e fazer o redimensionamento para um tamanho inferior, torna as imagens mais atrativas, pois o efeito não provoca o aparecimento da imagem “Pixelizada”. Isto implica mais espaço de memória para alojar imagens, que não existe na configuração de *hardware* atual do sistema Nidus. Em alternativa às codificações mais comuns nas imagens (PNG, JPEG, entre outras) existe a possibilidade de em imagens que não representem fotografias a utilização de SVG. No SVG ao contrário dos outros formatos indicados não é feita a representação de cada pixel da imagem, mas sim a definição de uma função que representa uma reta,

uma forma, um polígono, ou as coordenadas de pontos, sobre a forma de uma estrutura XML. O HTML já possui suporte para elementos SVG, Na utilização de SVG dentro de páginas HTML é possível a utilização de todas as funcionalidades inerentes ao CSS tais como criar animações.

Outra funcionalidade possível com a utilização de SVG é a alteração do texto existente na imagem apenas fazendo a alteração do valor do elemento XML, à semelhança da alteração do texto numa página HTML.

Os editores de SVG possuem variadas opções que não representam nada para o utilizador final, são apenas informações para o próprio editor. Para remover essa informação são utilizadas ferramentas para a remoção de tal informação. A ferramenta escolhida durante o estágio foi a SVGO[30], uma ferramenta desenvolvida em node.js que permite otimizar os SVG. Com esta ferramenta é possível remover toda a informação desnecessária e otimizar e simplificar algumas funções não afetando a qualidade da imagem. De modo a ser possível em tempo real fazer alterações é necessário, à semelhança das páginas HTML, cada elemento possuir um ID para fazer a sua procura na árvore XML e deste modo ser possível alterar a cor, o conteúdo do texto adicionar animações. O próprio SVG possibilita a inserção de imagens PNG, JPEG dentro do SVG. Estas imagens são adicionadas ao XML sob a forma de Base64. No apêndice A é apresentado o exemplo do SVG gerado pelo editor. No Apêndice B a respetiva compressão utilizando a ferramenta SVGO. Uma das considerações na utilização do SVGO é a necessidade de preservar os ID's e não os remover durante a compressão. Para tal na utilização da ferramenta é necessário indicar que pretendemos manter os ID's, para tal basta durante a utilização utilizar a opção '`--disablecleanupIDs`'.

Para criar animações à semelhança do HTML é possível atribuir classes a cada elemento do SVG e tirar partido das animações possíveis no CSS.

No gráfico da figura 3.5 é apresentado o tamanho ganho na compressão do SVG do Apêndice A. O Tamanho do SVG após a otimização com a ferramenta SVGO é de apenas cerca de 20% do tamanho original ($2\,485\text{b} - 80\% = 559\text{b}$).

■ SVG- Apêndice A

Otimizado ■ 559

Original ■ 2,485

Figura 3.5: Gráfico com comparação do tamanho entre original e otimizado(em Bytes)

3.2.4 Desenvolvimento a pedido de cliente

Durante o estágio existiu apenas um desenvolvimento pedido pelo cliente. O cliente pretende utilizar uma Nidus IT para controlar o seu sistema de alimentação dos animais de forma automática. Além de todas as questões a tratar no Back-end da Nidus, relativas a funcionamentos específicos da solução, o cliente pretende aceder na página da Nidus uma interface visual para inserir num monitor com o estado da alimentação, dos motores, da água. Este desenvolvimento para a sua realização usou as capacidades referidas no capítulo 3.2.3, de modo para obter uma imagem única e responsiva com animações em tempo real. Após o desenho da interface pretendida no editor SVG, o mesmo foi comprimido com a ferramenta SVGO indicada anteriormente.

De modo a criar uma interface atrativa ao utilizador foram criadas algumas animações nas linhas de alimentação, no silo da alimentação indicando o seu estado atual visualmente e numéricamente, tal como o estado das condutas de água.

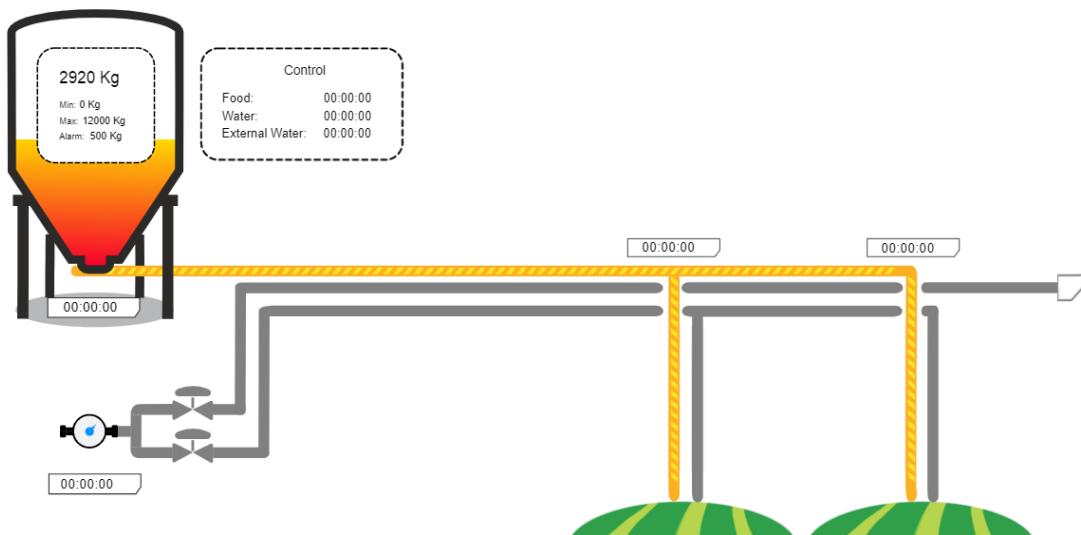


Figura 3.6: Exemplo do SVG utilizado na Solução

3.2.5 Correção de bugs

Durante o estágio apenas foi reportado a existência de um *bug*. Este pode-se com a estrutura criada para comunicação entre o *Front-end* e o *Back-end* ou com sistemas que pretendam integrar os equipamentos Nidus.

No momento do carregamento da página WEB, a mesma solicita um ficheiro XML com todas as informações necessárias. Em alguns dos pontos da estrutura como por exemplo o caso dos Input, Outputs, Sensores, entre outros, os mesmos são agrupados num elemento principal como é possível observar no exemplo 3.2.

```

1 ...
2 <Sensors>
3     <Entry>
4         <ID>0</ID>
5 ...
6     </Entry>
7     <Entry>
8         <ID>1</ID>
9 ...
10    </Entry>
11 ...
12 </Sensors>
```

Algoritmo 3.2: Estrutura parcial do XML da Nidus(Sensores)

Este ID é utilizado na comunicação de modo a indicar a que sensor se refere os dados. O problema surge aquando da eliminação de um sensor. Supondo o seguinte cenário. O Utilizaodr elimina na página o sensor com o ID 0, o sistema Nidus elimina o sensor e a entrada no XML com o ID 0 fica inutilizada passado o XML a conter os ID's 1,2,3,4 ,... de modo ao sistemas que integram a Nidus poderem utilizar o ID como um identificador único. O problema reside no código JavaScript que converte o XML numa variável JavaScript, mais concretamente num *array*. O sistema em alguns dos pontos do código ao invés de utilizar o valor presente no elemento ID, utilizava a sua posição no *array*, fazendo que em sistemas onde tivessem sido feitas alterações (Exemplo: eliminar sensores) os ID's não correspondiam. Supondo o exemplo 3.3 onde existem dois sensores com os ID's 0 e 1. Na conversão para um objeto no JavaScript era obtida a estrutura apresentada na figura 3.7.

```

1 ...
2 <Sensors>
3     <Entry>
4         <ID>0</ID>
5         <Name>0</Name>
6     </Entry>
7     <Entry>
8         <ID>1</ID>
9         <Name>1</Name>
10    </Entry>
11 </Sensors>
```

12 ...

Algoritmo 3.3: Exemplo do XML antes da eliminação de Sensores

```
{
  "0": {
    "id": 0,
    "name": 0
  },
  "1": {
    "id": 1,
    "name": 1
  }
}
```

Figura 3.7: Estrutura obtida no JavaScript - 1

Após a eliminação do Sensor 0, o XML obtido é similar ao apresentado no código 3.4. Neste caso o *array* criado no JavaScript era similar ao apresentado na figura 3.8. O problema reside na utilização da posição do *array* em algumas partes do código nomeadamente na procura do sensor. No exemplo apresentado na figura 3.8 antes da correção do *bug* o código existente ao necessitar dos dados do sensor com o ID 1 aceder á posição 1 do *array*, ao invés de procurar por cada elemento qual o elemento que possui aquele id.

No caso de terem sido eliminados alguns sensores como apresentado exemplo 3.4 os dados selecionados não correspondem ao pretendido pelo utilizador e apresentados na página de configurações. Os valores apresentados na página inicial, onde são mostrados os valores em tempo real e o mínimo e máximo do sensor não possuam este Bug estando limitado á alguma configurações dos sensores e a criação de Eventos.

```

1 ...
2 <Sensors>
3   <Entry>
4     <ID>1</ID>
5     <Name>1</Name>
6   </Entry>
7   <Entry>
8     <ID>2</ID>
9     <Name>2</Name>
10  </Entry>
11 </Sensors>
12 ...

```

Algoritmo 3.4: Exemplo do XML apóis eliminação do Sensor

```
{
    "0": {
        "id": 1,
        "name": 1
    },
    "1": {
        "id": 2,
        "name": 2
    }
}
```

Figura 3.8: Estrutura obtida no JavaScript - 2

De modo a resolver o problema identificado foi criada no projeto uma função `findObjectByKey`, apresentada no exemplo 3.5, com 3 parâmetros. No primeiro parâmetro é indicado o *array* onde deve o sistema procurar. Em seguida é indicado em que propriedade pretendemos compara e por último indicamos o valor que pretendemos encontrar. No código seguinte é apresentado o código utilizado. Caso o elemento não exista é devolvido o valor *null*.

```

1 function findObjectByKey(array, key, value) {
2     for (var i = 0; i < array.length; i++) {
3         if (array[i][key].toString() === value.toString()) {
4             return array[i];
5         }
6     }
7     return null;
8 }
```

Algoritmo 3.5: Função `findObjectByKey`

Para a correção deste *Bug* é necessário analizar todo o código e ao invés do acesso á posição diretamente no *array* é necessário a alteração de todo o código para fazer utilização da função e obter o valor correto.

3.2.6 Melhoramento da página

Após análise prévia da página do sistema Nidus, foi acordado a necessidade de desenvolver um sistema mais simplificado para o processo de criação de eventos. Pretende-se assim estudar a melhor solução para a criação das ações e das reações e uma interface para criar os eventos com recurso às ações e reações previamente criadas. Todas as opções e menus para a criação e manutenção das ações e reações mantém-se inalteradas no momento atual. Já a criação de eventos necessita de ser restruturado, após debate chegou-se á decisão de necessitar de um sistema visual onde o utilizador por blocos é capaz de criar a situação que pretende.

Durante o estágio foram analisadas as opções possíveis para a solução pretendida.

Após análise foi adotado a utilização do plugin Blockly[31], devido á sua semelhança com o pretendido (Scratch) e devido ao baixo espaço ocupado pós compressão e à possibilidade de criar os próprios tipos de blocos. Devido aos restantes projetos realizados durante o estágio este desenvolvimento ainda não se encontra concretizado e apenas foi selecionado a solução a adotar. No gráfico da Figura 3.9 é apresentado o espaço ocupado pelo código fonte do *plugin* escolhido.

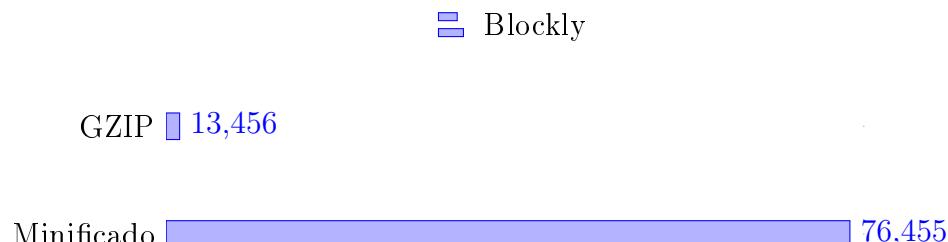


Figura 3.9: Gráfico com comparação do tamanho do plugin Blockly (em Bytes)

3.3 NB-Iot & Digi Xbee 3

O desenvolvimento do projeto NB-Iot & Digi Xbee 3 é composto por 4 fases, 3 das quais desenvolvidas durante este estágio. A fase não desenvolvida durante o estágio refere-se ao desenho e produção do *hardware* e a parte do *software* referente á leitura de sensores (comunicação entre *hardware* desenvolvido e *software*). As fases realizadas durante o estágio são a implementação do envio do pacote definido com os mecanismos de proteção e segurança, sincronismo dos tempos de leitura e envio e testes ao sistema.

3.3.1 Envio de dados para o portal

Como foi indicado no Capítulo 2.3 a estrutura de pacote a enviar é similar ao do outro produto desenvolvido pela Captemp. Este pacote é enviado através de um pacote UDP para o portal que posteriormente confirma a receção na camada de aplicação. O tamanho máximo definido para este pacote é de 1000 bytes. A estrutura criada pela Captemp segue o formato apresentado na figura 3.10.

No primeiro cabeçalho é possível obter os dados do equipamento que fez o envio tais como a data de envio, o IMEI, a versão do mesmo e o CRC do pacote para confirmar a integridade do mesmo. No restante do pacote são adicionados vários sub pacotes seguindo a estrutura apresentada na figura 3.10, o primeiro byte indica o tipo de dados se é o envio do valor de um sensor, ou uma configuração do equipamento(Exemplo:

Operadora), o byte seguinte fornece o número de bytes dos dados e posteriormente segundo o número de bytes, o valor (Exemplo: operadora = "ALTICE").

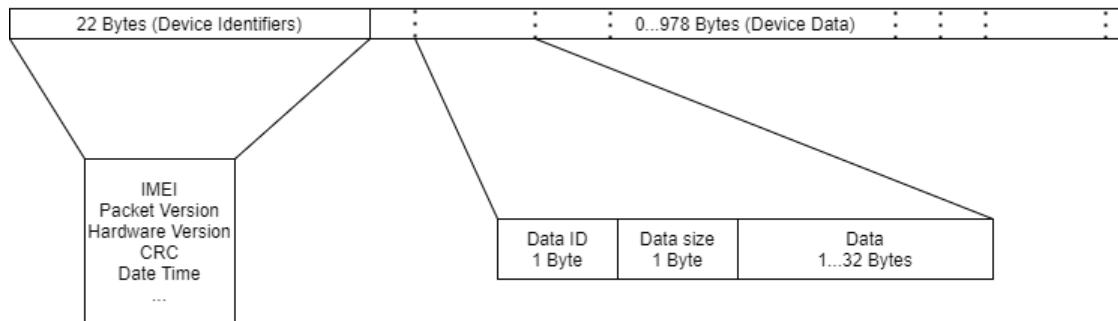


Figura 3.10: Estrutura do Pacote NB-IOT

3.3.2 Gestão de memória

O principal motivo da desistência da utilização deste equipamento como o equipamento principal da Captemp para o NB-Iot prendeu-se com a incorreta gestão de memória do MicroPython. Segundo a documentação, quando uma variável já não é acessível pelo código o espaço ocupado em memória por esta é libertado pelo *Garbage Collector* mas o espaço libertado nem sempre consegue ser reutilizado pelo MicroPython futuramente e este realiza a alocação no final ao invés de procurar o primeiro espaço disponível. No esquema apresentado na figura 3.11 é apresentado o comportamento da memória com a gestão nativa do MicroPython, com o decorrer do tempo a memória fica totalmente alocada não permitindo ao equipamento guardar novas leituras nem enviar para o portal.

3.3.2.1 Diminuição da alocação de memória

A modo de resolver a incorreta gestão de memória é necessário diminuir a alocação de novas variáveis. Para tal todo o código do programa possui todos os dados necessário em constantes e buffers de tamanhos fixos as quais nunca são eliminados ou alocam memória para mudar de tamanho, apenas alteram o seu valor. Os dois pontos críticos identificados são o *array* circular com as leituras dos sensores e a criação do pacote de envio para o portal *Cloud*.

Na criação do pacote todos os cabeçalhos fixos são alocados em constantes globais que não vem o seu valor alterado não afetando a memória e os valores que podem ser alterados são guardados em *buffers* globais criados no inicio e tem o seu tamanho estático. Devido à utilização de um módulo que não possui suporte para *multi thread*, não existe problema de sincronismo entre os fluxos ao utilizar as variáveis e os *buffers*

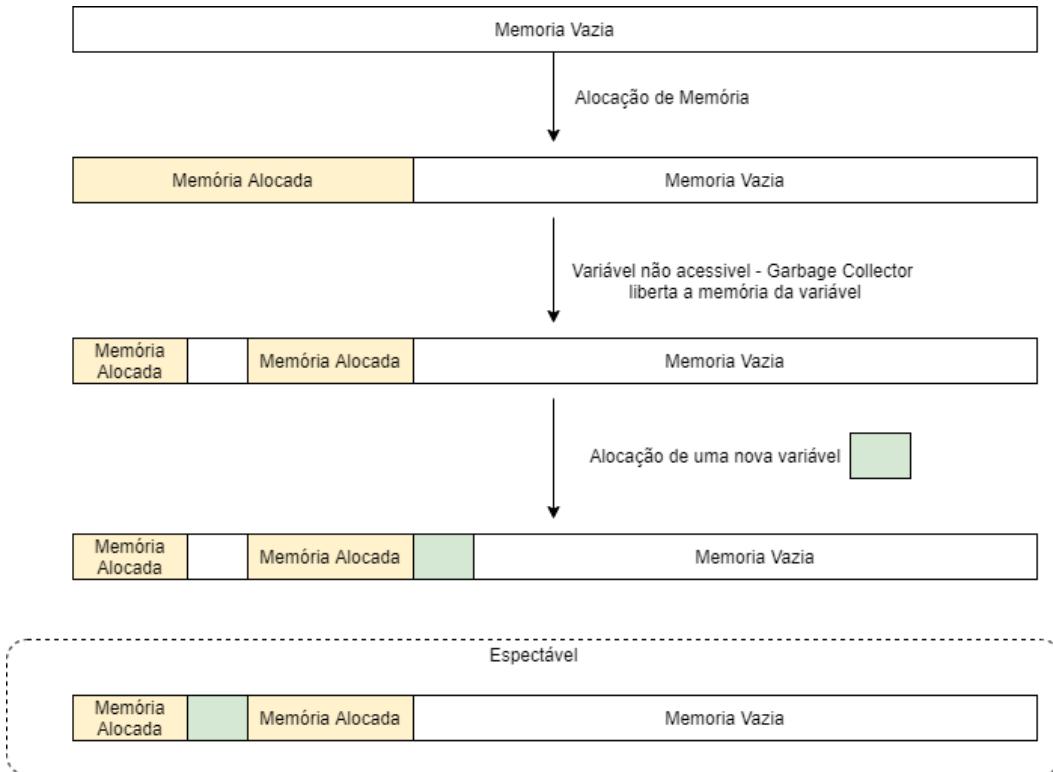


Figura 3.11: Comportamento da gestão de Memória do MicroPython

globais. Ao recolher um dado de modo a enviar para o portal, como por exemplo a operadora, este tem de ser alocado num *buffer* que depois é enviado pela rede para o servidor. Este *buffer* de bytes, com o tamanho fixo do máximo do pacote de envio, é utilizado para fazer a concatenação dos vários campos antes do envio ao invés da abordagem anterior da utilização de um *buffer* de tamanho dinâmico. Caso se pretenda adicionar valores a enviar é consultada o ultimo *byte* ocupado, encontrado numa variável separada e é alterado os bytes das posições seguintes para o bytes do valor não alocando memória para continuar o *array*.

Supondo que o *buffer* tem já preenchidos 300 bytes dos 1000, ao pretender adicionar o pacote da operadora, é copiado para a posição 301 o byte correspondente ao DATA ID do operador, no byte seguinte é colocado o tamanho de bytes que a operadora ocupa ("ALTICE" = 6 Bytes), e nos seguintes 6 bytes é colocada a *string*.

No exemplo acima elucidado todas as variáveis são estáticas, o DATA ID é definido no inicio do código, o tamanho foi previamente guardado numa variável auxiliar de tamanho fixo, e a operadora é solicitada a funções nativas do MicroPtyhon e guardado num *buffer* de tamanho fixo. Após a definição apenas são efetuadas copias de bytes entre variáveis e *buffers* não afetando a alocação de memória. No código 3.6 é exemplificado a operação anteriormente apresentada. Neste caso de modo a simplificar é apresentado um *buffer* do tamanho da operadora, no projeto foram criados *buffers*

do tamanho 1,2,4,10,16,32 bytes consoante os valores mais comuns, no caso particular de um valor dinâmico que possa ter por exemplo 20 bytes este é guardado no *buffer* de 32 e no momento da gravação apenas são copiados os 20 primeiros bytes.

```

1 c=bytearray(1000) # Packet Array
2 clean=bytearray(1000) # Empty Packet Array
3
4 #BUFFERS
5
6 cmd6=bytes(6)
7 cmdID=bytes(1)
8 cmdLEN=bytes(1)
9
10 #DATA IDs
11
12 c0=bytes([0x01])
13 (...)

14
15 cmdID = c0 # DATA ID
16 cmdLEN = (6).to_bytes(1, 'little') # Data Size
17 cmd6 = bytes(opr, 'ascii') # Data (string to bytes)
18 byteschange(c, 300, 301, cmdID) # Copy to packet array
19 byteschange(c, 301, 302, cmdLEN) # Copy to packet array
20 byteschange(c, 302, 308, cmd6) # Copy to packet array

```

Algoritmo 3.6: Exemplo com a concatenação da operadora

Ao copiar valores de entre *buffers* e não alocando espaço ao *array*, é necessário um maior controlo nos tamanhos das variáveis de modo a não copiar valores de *buffers* vazios ou posições inexistentes. A versão do MicroPython disponibilizada pela Digi não incorpora a biblioteca que faz a gestão de *arrays* limitando a cópia direta de *arrays* para outros *arrays* indicando apenas a posição inicial. Para tal a função desenvolvida, *byteschange* utilizada previamente no código, simula essa operação onde apenas indicamos o *buffer* de destino, a posição inicial, a final e a origem da cópia. Esta função é responsável por verificar se os tamanhos são possíveis de copiar e copiar posição a posição (byte a byte no caso apresentado) para as posições entre os valores indicados. No fim do pacote ser enviado é possível limpar o *buffer* chamando a mesma função indicando como origem da copia o *buffer clean*, um *buffer* constante do mesmo tamanho mas com os bytes todos vazios. Todos os *buffers* utilizados ao longo do projeto tem um *buffer* semelhante em tamanho, mas completamente vazio. Deste modo é possível utilizar sempre os mesmos *buffer*, não existindo a necessidade de alocar novos *buffer* ao longo do programa.

3.3.2.2 Gestão da memória durante leituras

À semelhança do *buffer* onde é gerado o pacote enviado para o portal, as leituras são um dos pontos críticos referente à alocação de memória. Para tal à semelhança do pacote de envio, é inicializado no início do programa, um *array* de tamanho fixo e com cada posição com um *array* do tamanho máximo ocupado por uma leitura com o máximo de sensores do equipamento. Ao adicionar uma nova leitura são colocados nos *buffers* intermédios todos os dados e são copiados para a posição seguinte à última posição ocupada. Caso a última posição ocupada corresponda à última do *array*, é adicionado sobre a primeira posição, criando assim o *array* circular.

Todas as operações para adicionar uma leitura ao *array* ou remover são efetuadas com recurso à função byteschange, copiando o *buffer* temporário com a leitura ou o vazio respetivamente. Na figura 3.12 é apresentado o esquema do *array* circular com as leituras. Uma vez que o *array* é definido quando o sistema inicializa e o utilizador pode alterar o número de sondas enquanto o equipamento está em funcionamento, é sempre alocado para cada leitura o número de bytes necessário para o máximo de sensores. Isto garante que caso o utilizador adicione um sensor, não seja necessário expandir o tamanho da posição no *array* causando os problemas de memoria já identificados. Para eliminar uma leitura é decrementado a posição corrente da última leitura no *array* e por questão de segurança é copiada o *buffer* correspondente a uma leitura vazia para a posição da leitura garantimos que não existem dados sem nexo.

3.3.3 Sincronismo de leitura e envio

À semelhança dos restantes produtos produzidos pela Captemp, neste produto existe sempre sincronismo de leituras entre os equipamentos. Para tal além de cada equipamento possuir um relógio interno (RTC), é necessário fazer o sincronismo dos mesmos na primeira leitura. Isto é feito para evitar que, por exemplo um equipamento programado para fazer leituras a cada 5 minutos, que entre em funcionamento às 00h00, não faça leituras dessincronizadas com outro ligado às 00h12. Neste caso o primeiro equipamento fará leituras ao minuto 5, 10, 15, 20, enquanto que o segundo fará ao minuto 17 e 22.

Para tal o equipamento no momento inicial do arranque verifica qual o intervalo de leitura e de envio, e verifica caso fosse ligado pelas 00:00 qual seria a próxima leitura ao momento atual, neste momento o equipamento não entra em modo de poupança de energia durante os 5 minutos até à leitura e apenas o tempo restante até ao valor caso fosse ligado pelas 00:00, no momento da leitura o equipamento já entra em modo de poupança de energia durante o tempo definido (Ex:5 minutos). Este método é

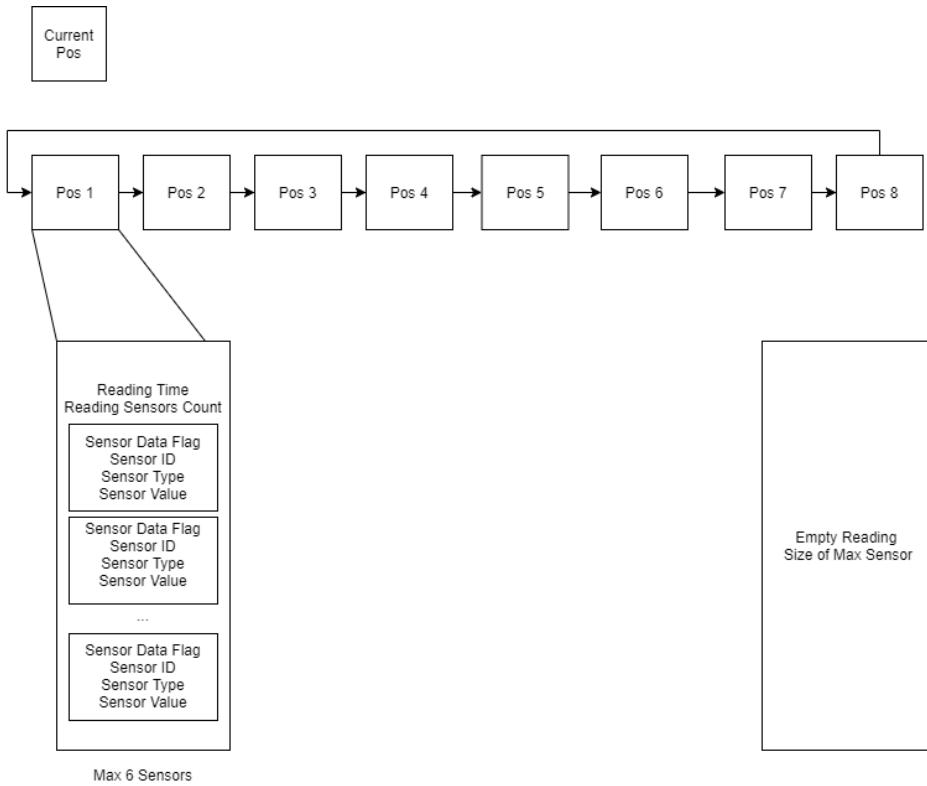


Figura 3.12: Array Circular com leituras

igualmente aplicado ao intervalo de envio. Quando do envio é enviado na resposta proveniente do servidor o *Timestamp* do servidor de modo a sincronizar o tempo de todos os equipamentos e configurar nos novos equipamentos que sejam ligados e ainda possuam o valor *default* do RTC, no caso do escolhido no projeto 1 de Janeiro de 2000.

Caso o equipamento possuir uma data inferior à estipulada de 1 de Janeiro de 2019, o mesmo não faz leituras, esperando pelo intervalo de envio para enviar um pacote apenas com os campos fixos e sem leituras de modo a obter a resposta com o *timestamp* do servidor para iniciar as leituras.

3.3.4 Encriptação dos dados

De modo a proteger os dados na rede todo o pacote é encriptado com recurso á biblioteca MicroPython-AES disponibilizada *online* [32]. Esta biblioteca foi desenvolvida para o MicroPython Base e não para a versão disponibilizada nos módulos DIGI. É necessário reformular a biblioteca do AES para MicroPyhton de modo a não utilizar a biblioteca de gestão de *arrays* indisponível nesta versão. É necessário alterar a biblioteca para não usar as funcções da biblioteca de gestão de *arrays* no momento que os dados são encriptados e desencriptados. No código 3.7 é apresentado o problema inerente a esta biblioteca (linha 4) e a sua respetiva resolução (linhas 13 e 14).

```
1 for offset in range(0, len(data), block_size):
2     block = data[offset:offset + block_size]
3     block_func(block)
4     data[offset:offset + block_size] = block # ERROR
5     #[‘array’ object does not support item assignment]
6 ...
7
8 ...
9 for offset in range(0, len(data), block_size):
10    block = data[offset:offset + block_size]
11    block_func(block)
12    # Solution [‘array’ object does not support item assignment]
13    for i in range(block_size)
14        data[offset+i]= block[i]
15 ...
```

Algoritmo 3.7: Adaptação do Código para o equipamento Digi

3.4 Kea Tracker

De modo a substituir um produto descontinuado e apresentar novas soluções aos clientes, irão ser utilizados *beacons* para armazenar os valores da temperatura, humidade e pressão ambiental para posterior envio para o portal Senslive. No início do estágio não estava disponível nenhuma versão de *Firmware* com armazenamento dos dados e estava proposto o desenvolvimento da solução utilizando o Espruino, uma plataforma que permite fazer a programação de microcontroladores com recurso a JavaScript. Foram realizados pequenos testes e chegou-se à conclusão que a camada interpretadora do Javascript tem um consumo mais elevado comparando com uma versão desenvolvida em C. Além do maior consumo energético, já foi disponibilizado um *Firmware* com armazenamento de leituras para posterior arquivo. Devido a esses dois fatores, não será desenvolvido o *Firmware* como estava inicialmente proposto, mas em alternativa os esforços serão concentrados no melhoramento e desenvolvimento da aplicação responsável por obter os dados e enviar para o portal Senslive.

3.4.1 App - Alterações necessárias

A aplicação móvel na primeira fase irá ser baseada na fornecida pela Ruuvi e serão alteradas as referências ao *website* da RuuviTag para o *website* da Captemp e a alteração das imagens para o novo logótipo da solução. Devido é necessidade de desenvolvimento do *Back-end* no Senslive, para já será adaptada apenas a versão Android da aplicação. Numa segunda fase do projecto, será desenvolvida uma aplicação para Android e iOS

utilizando uma plataforma que permita o desenvolvimento para ambas as plataformas com apenas um código fonte, tais como o NativeScript[33], o Ionic[34] entre outras.

3.4.2 App - Novas funcionalidades

A aplicação fornecida pela Ruuvi não está desenvolvida para a consulta das leituras quando o *beacon* não esteve ao alcance. É necessário então desenvolver o módulo por obter essas leituras e guardar na base de dados para a restante aplicação enviar para o portal Senslive.

O *Firmware* disponibilizado converte a RuuviTag num *beacon* connectável e disponibiliza sobre a forma de um serviço os Logs da mesma[35]. Para tal é necessário adaptar a aplicação de modo a quando esta esteja encontra um novo *beacon* ao alcance descarregue o seu Log. Quando o *beacon* se encontra sempre ao alcance a aplicação não necessita de fazer a conexão para descarregar os Logs visto esta além da conexão faz o *broadcast* dos dados em tempo real. O principal problema a resolver neste projeto é o sincronismo de leituras, já retratado no capítulo 3.3.3 quando a aplicação se encontra a registar através do *broadcast*. Nos Logs não é possível fazer esse sincronismo devido à mesma não possuir essa funcionalidade internamente. No *download* dos Logs é necessário verificar quais as leituras de Log que possam ser referentes a intervalos que existiu a receção de *broadcast* e tiveram leituras sincronizada, não sendo necessário o armazenamento. É esperável com o *Firmware* selecionado obter uma autonomia entre 2 a 3 anos com ligações esporádicas para descarregar Logs ou 6 meses com a conexão sempre ativa. Nos Fluxogramas dos apêndices C, D e E é possível observar o ciclo referente à obtenção das leituras em tempo real para a atualização do dashboard, o descarregar dos Logs, o registo das leituras no intervalo definido e o envio para o portal também com sincronismo.

Sempre que a aplicação recebe uma nova transmissão cria um novo fluxo de processamento de modo a não congestionar a restante aplicação. É analisado a origem da transmissão e caso seja um dos *beacons* registadas na aplicação esta é processada, em caso contrário é descartado. Caso seja uma transmissão de um *beacon* que esteve fora de alcance num intervalo superior a 5 Minutos, correspondente ao intervalo de Log, é feito as tentativas de *download* dos Logs, após esse *download* é feito o processamento dos valores em tempo real presentes no pacote.

Além dos valores em tempo real no dashboard são armazenados na memória RAM os valores referentes à última comunicação registada em cada minuto. Ao receber um pacote, caso este seja o primeiro do minuto da transmissão atual, isto é desde o segundo 0 do minuto esta é a primeira, então é registada como a leitura correspondente à do minuto atual. Caso já exista uma leitura desse minuto é atualizado apenas o *dashboard*

da aplicação. No minuto seguinte esta variável é atualizada para a do minuto seguinte. Este processo está representado no fluxograma do apêndice G.

Paralelamente à receção dos pacotes existem dois fluxos de processamento responsáveis por registar em memória não volátil as leituras a enviar e o envio das mesmas para o Portal. Segundo o funcionamento de sincronismo de leituras já apresentado no capítulo 3.3.3, no intervalo correto é percorrido todos os *beacons* é analizado a leitura sincronizada ao minuto e caso esta seja de uma data superior à ultima leitura é armazenada. Caso o *beacon* tenha estado sempre ao alcance irá ser armazenada a leitura correta do minuto do registo definido nas configurações e criamos sincronismo entre regtos dos vários equipamentos. Caso o *beacon* tenha estado ao alcance mas no momento do registo sincronizado não se encontrava é guardada a última com a data/hora correspondente. Isto previne que, caso o *beacon* não esteja ao alcance no minuto exato da leitura para sincronizar entre todas e caso tivesse estado ao alcance desde o último registo, exista pelos menos um registo a mostrar no gráfico apesar de este não estar sincronizado. De modo a todos os *beacons* ao alcance comunicarem pelo menos uma vez no minuto do registo este processo é executado a cada intervalo de registo constante o configurado mas com um atraso de 50s relativamente ao segundo 0 do minuto. Paralelamente a este fluxo existe o processo referente ao envio para o portal Senslive. Este também segue o mesmo princípio de sincronismo de envio do capítulo 3.3.3. No apêndice C e D é possível analizar o fluxograma referente ao processo de registo de leituras e do envio descrevido anteriormente.

3.4.2.1 *Download manual*

A aplicação desenvolvida permitirá ainda ao utilizador a possibilidade de fazer o *download* dos Logs manualmente ao invés do *download*, configurar o máximo de tentativas do *download* automático, com o valor de *default* definido em 1. Além dessa configuração é implementada uma medida de segurança onde caso o RSSI seja inferior a -90, significando que se encontra longe não são descarregados os Logs devido à possibilidade de perder a ligação no *download*, neste caso é apresentado ao utilizador uma nota para aproximar o equipamento do *beacon*.

3.4.3 Desenvolvimento de uma aplicação multi-plataforma

A *Framework* escolhida foi o Ionic, uma *Framework open-source* que permite o desenvolvimento de aplicações *mobile* para múltiplas plataformas com o mesmo código, diminuindo o esforço necessário para o desenvolvimento e simplificando os processos de atualização visto apenas existir um código fonte a atualizar.

Esta fase do projeto Kea.Tracker é referente à conversão da aplicação existente em Kotlin para o Ionic e migrar todas a funcionalidades existentes atualmente na versão em Kotlin. Ao invés da utilização de *Activities* como no Kotlin o Ionic utiliza um sistema similar a um serviço WEB onde são criadas várias rotas e é possível navegar entre elas. Todo o *design* da plataforma seguiu o mesmo layout da aplicação já existente, onde foram retiradas as opções de alertas e vizualização dos gráficos visto pretendermos centralizar essas opções no portal Senslive e utilizar a aplicação apenas como um *Gateway*. Fica assim disponível na aplicação apenas a opção de adicionar novos equipamentos ao gateway neste caso à App, editar o nome do equipamento para mais fácil distinção entre os vários equipamentos, as configurações da aplicação e um serviço em *Background* para realizar as leituras sem necessidade da aplicação estar aberta.

3.4.3.1 Distinção entre *Beacons* & Leitura de Dados

Para facilitar a utilização da interface da aplicação ao utilizador da mesma, ao adicionar um novo dispositivo apenas são mostrados os dispositivos ao alcance que correspondam a equipamentos da Ruuvi e que estejam a transmitir os dados nos formatos definidos. Todos os equipamentos BLE possuem nos dados transmitidos por *Broadcast* possuem primeiro informações no *header* da transmissão, tais como o fabricante, denominado de "*Company Identifier*". No caso dos *beacons* da Ruuvi é enviado o código correspondente à Ruuvi, "0x0499". Estes códigos podem ser consultados na documentação do Bluetooth [36]. Juntamente com o *Company Identifier* o protocolo define mais alguns parâmetros tais como o tamanho do cabeçalho. No caso dos *beacons* da Ruuvi são enviados os dados referentes aos sensores nos diversos formatos definidos na documentação da Ruuvi [37], precedidos por um byte com o valor correspondente ao formato dos dados.

Segundo a documentação a aplicação deve suportar a leitura em tempo real de dois formato o formato 3 e 5, pois são estes os existentes nos *beacons* atuais em produção.

3.5 dot.Tracker

A pedido de um cliente, foi proposto o desenvolvimento de uma plataforma WEB para fazer a monitorização de pessoas e objetos em tempo real. O projeto passou por várias etapas das quais destacam-se a análise dos requisitos do cliente, análise de tecnologias disponíveis, análise de soluções existentes já em comercialização, o desenvolvimento do portal Web, desenvolvimento do Back-End, e testes ao sistema. Apesar do projeto ser apenas desenvolvido por um elemento, mas devido á maior complexidade e duração do mesmo foi adotada a metodologia SCRUM com entregas/apresentações ao cliente para

obter o *feedback* do trabalho desenvolvido e assim poder alterar alguns dos requisitos solicitados.

O Cliente indicou que devia ter atualizações dos mapas em tempo real, alertas enviados para o cliente WEB caso este esteja *online* e por email. É assim possível definir a tabela, apresentada na tabela 3.2 com os requisitos da solução e a sua importância no desenvolvimento.

Tabela 3.2: Requisitos da solução

Requisito	Descrição	Importância (1-10)
Portal Cloud (Front-End)	Portal Cloud com mapas em tempo Real	7
Portal Cloud (Back-End)	API REST para integração com o Front-End e recolha dos dados para a localização	9
Histórico de posições	Possibilidade de re-visualizar no mapa o percurso entre datas	5
Alertas Email	Alertas de Email (Exemplo: Entrada e Saída de zonas criadas no mapa)	6
Alertas Web	Alertas Informativos no Mapa (Exemplo: Entrada e Saída de zonas criadas no mapa)	2

De seguida são apresentados as funcionalidades e objetivos para cada módulo do projeto *Front-end* e *Back-end*.

3.5.1 Portal Cloud - *Front-End*

O *Front-end* da solução é desenvolvido com recurso á *Framework Vue*, tornando a solução numa solução *Single-Page Application*. A adoção da *framework* é baseada na necessidade de possuir fluidez na navegação entre páginas e igualmente nos mapas em tempo real minimizando o atraso.

A plataforma é capaz igualmente de suportar várias *Companies*, significa isto que é possível criar várias organizações ou empresas distintas (*Companies*) e temos os administradores e os utilizadores normais de cada *Company* que apenas tem acesso ás suas definições e equipamentos. Possibilitando fornecer o projeto como uma solução *Cloud* a variados clientes no mesmo servidor, onde cada um apenas possui o acesso ao que pertence à sua *Company*. No final o utilizador da plataforma deve ser capaz de realizar as seguintes operações:

- Login na plataforma para visualizar os dados
- Visualizar mapas com atualizações em tempo real
- Editar o seu perfil
- Visualizar a página numa língua á sua escolha
- Utilizar a plataforma em vários equipamentos PC, Tablet, Smartphone,...
- Gerir utilizadores (Administradores)
- Gerir equipamentos (Administradores)
- Gerir mapas e zonas (Administradores)
- Gerir alertas (Administradores)
- Iniciar/ finalizar missões (Administradores)

3.5.1.1 *Login*

Para aceder á plataforma é necessário aos utilizadores procederem ao login na mesma, uma vez que possuímos uma API o *login* é realizado através de Json WebTokens ou simplesmente, JWT. No momento do *login* são enviadas as credenciais para o servidor, caso este as valide crie um *Token* que é devolvido ao cliente e este em futuras requisições à API inclui o *token* identificando-se e autenticando-se perante o servidor. Este método de login é regularmente utilizado devido á necessidade de possuir métodos *stateless* ao invés da utilização de variáveis de sessão. No exemplo apresentado na figura 3.13 é apresentado o conteúdo de um *Token* JWT.

3.5.1.2 *Rotas e proteções*

Para proteger as páginas apenas referentes a administradores e as gerais ao público não autenticado através da *framework Vue* e da funcionalidade de rotas foram criadas as rotas necessárias para o funcionamento da plataforma e os restantes *end-points* são tratados por uma vista de erro indicando ao utilizador que não existe aquele *end-point*. Em cada rota é aplicado um *middleware* para verificar as permissões. Os *middlewares* criados verificam se o utilizador está autenticado e no caso de serem necessárias permissões se este as possui ou não. Caso o *middleware* indique que não tem acesso a *framework Vue*, redireciona para a rota de tratamento de erro apresentado a mensagem de erro correspondente.

Encoded	Decoded						
<pre>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiawF0IjoxNTE2MjM5MDIyfQ.cThIIoDvwdueQB468K5xDc5633seEFoqxwxF_xSjyQQ</pre>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">HEADER: ALGORITHM & TOKEN TYPE</td></tr> <tr> <td style="padding: 5px;">{ "alg": "HS256", "typ": "JWT" }</td></tr> <tr> <td style="padding: 5px;">PAYOUT: DATA</td></tr> <tr> <td style="padding: 5px;">{ "sub": "1234567890", "name": "John Doe", "iat": 1516239822 }</td></tr> <tr> <td style="padding: 5px;">VERIFY SIGNATURE</td></tr> <tr> <td style="padding: 5px;">HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret) <input checked="" type="checkbox"/> secret base64 encoded</td></tr> </table>	HEADER: ALGORITHM & TOKEN TYPE	{ "alg": "HS256", "typ": "JWT" }	PAYOUT: DATA	{ "sub": "1234567890", "name": "John Doe", "iat": 1516239822 }	VERIFY SIGNATURE	HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret) <input checked="" type="checkbox"/> secret base64 encoded
HEADER: ALGORITHM & TOKEN TYPE							
{ "alg": "HS256", "typ": "JWT" }							
PAYOUT: DATA							
{ "sub": "1234567890", "name": "John Doe", "iat": 1516239822 }							
VERIFY SIGNATURE							
HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), your-256-bit-secret) <input checked="" type="checkbox"/> secret base64 encoded							

Figura 3.13: Token JWT codificado e descodificado [4]

3.5.2 Portal Cloud - *Back-end*

O *Back-End* é responsável por fornecer uma API REST ao *Front-end* para o mesmo obter os dados da base de dados. É igualmente responsável por obter os dados provenientes dos *Gateways* dos *beacons* e calcular as suas posições para apresentar no mapa.

O primeiro problema identificado no *Back-end* é a necessidade de possuir atualizações em tempo real das posições. O *Front-end* não é capaz de calcular quando um *beacon* comunica com o *Back-end* apesar do mesmo enviar o *broadcast* em tempos regulares, mas tanto o *beacon* pode não estar ao alcance do *Gateway*, como o mesmo pode apenas estar ao alcance de menos de X(dependendo do algoritmo) *Gateways* impossibilitando a utilização do algoritmo. Deste modo não é eficaz o *browser* solicitar ao servidor num intervalo regular qual a posição do elemento no mapa.

Para tal além do serviço WEB é disponibilizado um servidor de WEB-Sockets para comunicações em tempo real entre o *Back-end* e o *Front-End*. Os WEB-Sockets são uma tecnologia que permite aos *browsers* mais recentes, ter um canal em tempo real com o *Back-end* sem a necessidade de fazer vários pedidos HTTP, o que acarreta todo o processo do protocolo, como o *3-Way Handshake*. No momento da criação do Web-Socket é criado uma ligação TCP a qual não é finalizada até ao fechar do WEB-Socket, o que elimina o sobre carregamento da criação de pedidos e soluciona o problema da comunicação em tempo real para as atualizações dos mapas.

Outro problema a solucionar, identificado na análise das tecnologias e soluções existentes, é a falta de sincronismo da comunicação dos *Gateways*. Supondo um cenário com 5 *Gateways* e 1 *beacon*. O *beacon* ao intervalo de tempo X1 comunica o pacote e apenas 4 *Gateways* recebem o pacote e o enviam para o servidor através de um

pedido HTTP. O servidor apesar de ter configurado 5 *Gateways* não é capaz de prever se o 5º *Gateway* irá comunicar a transmissão do *beacon*, o mesmo pode não estar ao alcance, pode não ter comunicação ao servidor, pode estar desligado ou pode haver algum problema na rede que atrasa a chegada do pacote. Igualmente por variados motivos os 4 *Gateways* que enviaram o pacote ao servidor não irão chegar todos em simultaneamente, criando o problema "Já chegaram todos os pacotes? Cálculo com os que tenho, ou espero que chegue mais algum pacote?",

A primeira solução adotada para resolver o problema acima citado foi, ao contrário da utilização de servidores WEB como o APACHE2 ou NGINX comuns nas soluções tradicionais a utilização de Node.js (módulo express).

As soluções tradicionais não possuem nenhum sincronismo entre pedidos, isto é, não é possível aceder diretamente ao valor existente no fluxo de um outro pedido e seria necessário armazenar momentaneamente todos os pacotes na base de dados e ter a tabela em constante escrita e leitura, não sendo o mais eficaz no cenário deste projeto. Na solução desenvolvida, é possível armazenar variáveis entre pedidos distintos e irá ter além do serviço WEB o servidor de WEB-Sockets no mesmo processo, centralizando assim os dois serviços e possibilitando igualmente durante o processamento do pedido HTTP o envio de mensagens através do WEB-Socket.

O módulo express é uma *Framework* desenvolvida para Node.js para aplicações que necessitam de rapidez de resposta nos pedidos e à semelhança de outras *frameworks* em outras linguagens permite a utilização de rotas para a criação dos *end-points*.

Na lista apresentada de seguida estão selecionados os pontos principais das funcionalidades do *Back-End*:

- API REST para o *Front-End*(Login+ Dados)
- API REST para o POST dos *Gateways*
- Serviço WEB (express) para disponibilizar o *Front-End*
- Serviço WEB-Sockets
- Algoritmo de posicionamento
- Envio de alertas

3.5.2.1 Sincronismo da receção de pacotes

Os *Gateways* escolhidos enviam dois tipos de pacotes identificados pelos identificadores GPRP e RSPR. Os pacotes GPRP são referentes ao envio de uma transmissão do *beacon*. Os pacotes RSPR são referentes ao restante da transmissão do *beacon* no caso

de esta transmitir no Broadcast uma mensagem superior a 31 bytes. O conteúdo do pacote RSPR apenas contem o restante da mensagem não afetando a posição do *beacon*.

De modo ao servidor possuir todos os pacotes GPRP aquando da utilização no momento da chegada este é armazenado numa variável onde é possível consultar as ultimas transmissões de todos os *Gateways*. Quando é recebida uma transmissão do tipo GPRP caso esta seja enviada do *Gateway X* e a ultima transmissão do *Gateway X* for inferior a 5 segundos (intervalo de envio do *beacon*) este mesmo pacote é descartado, pois o mesmo é referente a um já recebido. Caso seja superior aos 5 segundos do próprio *Gateway* e de todos os restantes, significa que a mesma se trata de uma nova transmissão, neste caso antes de guardar na variável provisória, são selecionados todos os pacotes do intervalo correspondente á ultima transmissão e escolhidos os 4 com o valor mais próximo dos *Gateway*, devido a estes serem os mais fiáveis e é aplicado o algoritmo *Least Squares Estimation* ou simplesmente LSE. Após obtenção da posição estimada esta é guardada na Base de dados para consulta futura e são notificados os clientes de uma nova posição do equipamento. Caso essa posição seja referente a alguma zona de alertas previamente definida é gerado os alertas a enviar. No fluxograma do Apêndice F é apresentado o fluxograma da aplicação para a receção dos pacotes.

3.5.2.2 Diferenças de alturas

Uma questão analisada nos primeiros testes é a divergência nas distâncias observadas quando o *beacon* apenas se movia no eixo do Z (altura). O *Gateway* calcula a potência do sinal(RSSI) recebido do *beacon*, esta receção ocorreu em linha reta entre os dois equipamentos e nos mapas utilizados apenas são utilizadas dimensões 2D significando que o valor da distância obtido pelo RSSI é referente ao mesmo em linha reta e não á distancia numa planta 2D, como é apresentado na figura 3.14.

Para tal na aplicação teve de ser desenvolvido a possibilidade de o utilizador configurar as alturas a que se encontram os *Gateways* e ps *beacons*, de modo a compensar a distância. No exemplo apresentado na figura 3.14 o servidor irá receber a informação que o *Gateway* recebe um pacote com o RSSI correspondente á distância de 5 metros mas no mapa não é a distância a considerar para utilizar o algoritmo que apenas possui suporte a coordenadas 2D. Para tal é necessário conjugar a diferença de alturas e a distância em linha reta para calcular a distância a considerar para o algoritmo, segundo o teorema de Pitágoras. No caso exemplificado ao invés dos 5 metros deve ser considerado os 4 metros.

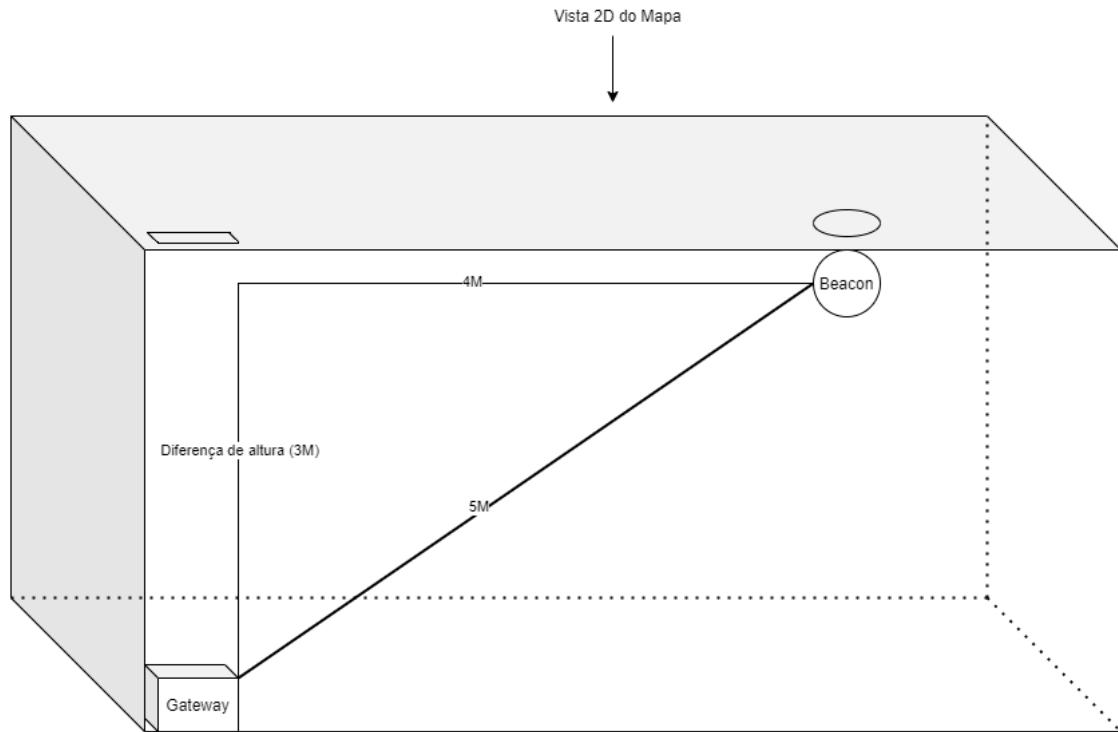


Figura 3.14: Diferença entre distâncias Planta vs realidade

3.5.2.3 Algoritmo de triangulação

Para o projeto foi selecionado o Algoritmo LSE analisado no Capítulo 2.6.1.3, pois o mesmo demonstra os melhores resultados em vários testes. Como parâmetros deste algoritmo é necessário indicar as posições X,Y de cada *Gateway* e a distância calculada previamente segundo o RSSI e o algoritmo de Pitágoras de modo a resolver a diferença referente à altura.

No Caso do mapa ter configurado um número superior ao necessário para a utilização do algoritmo, neste caso 4, e nos casos em que existam mais do 4 *Gateways* que receberam comunicações apenas são utilizadas as 4 que tiverem menores valores de distância, visto que estando mais próximas o algoritmo não tem um erro tão elevado. Previamente à seleção das 4 melhores transmissões caso alguma das transmissões tenha uma distância inferior a 1 metro não é utilizado o algoritmo e é definido a posição do *beacon* igual à posição do *Gateway* no mapa. Caso não exista nenhum valor abaixo do 1 metro é utilizado o algoritmo indicado.

De modo a otimizar o processamento do algoritmo, devido a este envolver cálculos em matrizes e estas não possuírem tamanhos dinâmicos, não foi utilizada nenhuma biblioteca para realizar as operações em matrizes devido ao peso computacional excessivo que estas apresentam. Ao invés, as operações sobre matrizes foram implementadas manualmente conforme as necessidades do projecto, usando as regras de operações. Cada

variável possui o valor de uma posição da matriz correspondente. No apêndice G é fornecida a função responsável por retornar a posição estimada do *beacon* ou -1 em caso de erro ou o mapa definido não possua a escala definida, apesar da plataforma não permitir adicionar *Gateways* a mapas sem escala. No caso específico apresentado nas linhas 47 a 50, de modo a otimizar a eficiência do algoritmo, é efetuada a multiplicação por 0.5 em vez da divisão por 2, uma vez que esta é computacionalmente mais eficiente.

3.5.2.4 Filtros - RSSI

Após o desenvolvimento do algoritmo e de alguns testes à receção de pacotes, foi possível verificar que, numa sala com 6 metros analisar numa sala com 36 metros quadrados e 4 *Gateways* e com 4 Gateways, o sinal dos *beacons*, colocadas em posições fixas (estratégicas), sofre de algumas flutuações, fazendo com que o algoritmo devolva posições incorrectas. De modo a eliminar essas flutuações é necessário filtrar os pacotes com ruido e tentar aproximar o valor do esperado. Para tal irá ser utilizado um modelo matemático denominado por Filtro de *Kalman*[38]. Este modelo matemático é capaz de ao longo do tempo consonante os valores recebidos tentar aproximar o valor recebido do esperável segundo a tendência anterior. Deste modo é necessário alterar o servidor de modo a além de guardar temporariamente os pacotes inserir o valor no filtro correspondente. Por cada *beacon* existente na plataforma existe um conjunto de filtros, um por cada *Gateway* que já recebeu alguma comunicação do *beacon*. Sempre que um *Gateway* comunica a receção de um pacote do *beacon* o valor recebido do RSSI é inserido no filtro onde é retornado o valor estimado segundo os valores anteriormente recebidos. No exemplo da figura 3.15 é possível observar o funcionamento do filtro ao longo do tempo. Para o projeto é utilizado o plugin JavaScript[5] disponível no GitHub onde é implementado o filtro *Kalman* para dados do tipo 1D. Um dos fatores para a utilização deste filtro ao invés de outros, além da sua utilização por parte de outras pessoas na comunidade científica no âmbito das localizações *inndoor*, foi a necessidade de um algoritmo rápido a estabilizar os dados e que não sobrecarregue o servidor. Após análise do filtro foi possível analisar, que por cada filtro apenas são alojados em memória o último valor e 6 variáveis referente aos cálculos necessários na próxima filtragem, não sendo o espaço assim influenciado pelo tempo que tiver em funcionamento.

3.5.2.5 Filtros - Posição no mapa

Após filtragem do ruido gerado pela transmissão dos dados no meio ambiente, é necessário ainda filtrar certas ocorrências que ocorrem em certas posições e suavizar o movimento quando os *beacons* se encontram em movimento. O primeiro caso a filtrar

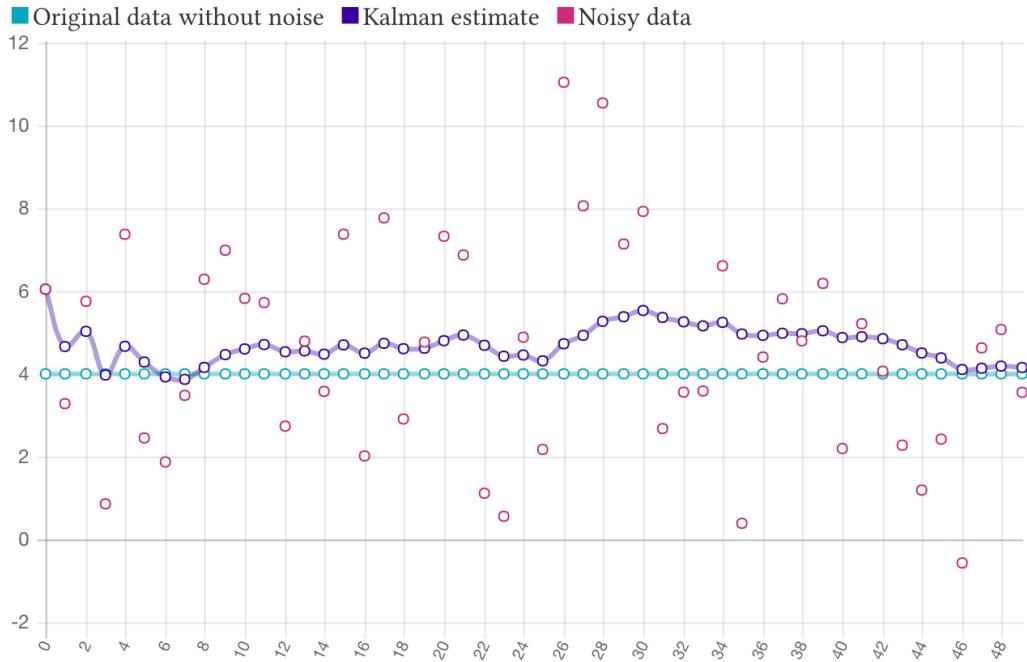


Figura 3.15: Exemplo da aplicação do Filtro Kalman [5]

referente ao caso do *beacon* esteja numa posição X₁,Y₁ no momento t e no momento t+1 em X₂,Y₂ onde a distância entre pontos é elevada. Para tal, quando uma posição é calculada segundo o algoritmo LSE, esta posição é comparada com a última guardada do mapa correspondente e é calculada a distância entre ambos os pontos, de seguida é calculada igualmente a velocidade segundo o tempo entre as leituras e a distância. Caso esta seja superior à definida, 5 m/s (18 Km/h), esta posição não é considerada enquanto o tempo entre leituras não permitir o deslocamento da pessoa/objeto para essa mesma posição. O valor máximo da velocidade é possível de ser adaptado consoante o pretendido medir, mas neste caso o cliente pretende monitorar pessoas e objetos. Os objetos alojados no armazém devem ter um deslocamento maioritariamente de 0 pois encontram-se parados e o valor de 5m/s corresponde á possibilidade de estar em movimento num empilhador ou similar. A velocidade média de uma pessoa a andar é de cerca 1 a 2m/s em caminhada[39] e cerca de 1 a 3.8m/s [40]. em corrida, valores sempre inferiores à já definida no intervalo dos objetos.

Outro filtro a aplicar é no caso particular de um *beacon* se encontrar parada e apesar do filtro *Kalman* filtrar o RSSI a posição possuir pequenas oscilações. Para tal, se for gerada uma nova posição, esta tem de passar no filtro da velocidade pois se esta encontrar-se parada e perto da posição da última comunicação com pequenas oscilações e esta oscilação for inferior a 1 metro a posição é guardada mas não é enviada nenhuma notificação para ser atualizada nos mapas através do Web-Socket. Na próxima receção caso esteja numa posição inferior a 1 metro da posição do mapa dos *browsers*

e a menos de 1 metro da ultima posição guardada existente apenas no servidor, não é atualizada a posição apenas novamente no servidor para futura comparação, caso não aconteça e esteja a menos de 1 metro da ultima guardada no servidor e a mais de 1 metro da presente nos *browsers* esta é guardada e enviada para os *browsers* através do Web-Socket. Caso se encontre a uma distância superior a 1 metro em ambas e abaixo dos 5m/s a posição é atualizada no servidor e nos *browsers*.

3.5.2.6 Zonas de alerta

Um dos requisitos do cliente na solução é a definição de zonas de alerta, onde é possível indicar se a plataforma deve enviar alertas quando os *beacons* saem ou entram de uma zona. Para a criação de uma zona de alerta no *Front-end* é necessário o utilizador definir os vários vértices da zona no mapa selecionado. Após a definição da regra e dos limites, é necessário que, durante o cálculo de uma posição, o servidor verifica se o *beacon* saiu ou entrou numa zona definida como alarme. No caso de polígonos regulares tais como círculos, retângulos, triângulos é facilmente calculado se um ponto se encontra dentro da área. No caso de polígonos irregulares o caso é diferente. Para tal foi utilizado na solução o plugin "point-in-polygon"[41]. Este algoritmo é capaz de indicar se um determinado ponto X,Y se encontra dentro do polígono ou fora, definido pelos conjunto de pontos fornecidos .

3.5.3 Reutilização de *beacons*

De modo a construir uma plataforma modular e reutilizável, cada *beacon* pode ser reutilizada na plataforma. No momento da configuração é adicionada um *beacon* à conta da organização. Para as posições serem guardadas na base de dados é necessário o utilizador aceder à plataforma e iniciar uma missão indicando a data de início, se pretende localizar pessoas ou objetos e qual o utilizador/objeto a localizar. Neste momento sempre que o *beacon* seja localizada nos mapas referentes à organização detentora do *beacon* a posição é guardada com informação sobre a missão do *tracking*. Quando o *beacon* já não é necessária o utilizador pode terminar a missão do *beacon* e a plataforma deixa de registar as posições enquanto não possuir missão ativa novamente. Desta forma o cliente pode adquirir vários *beacons*, mas caso pretenda mudar de objetivo, como por exemplo um empregado deixe de trabalhar na empresa e começar outro funcionário pode ser trespassada o *beacon* e nos históricos estes ficam separados e é possível de forma intuitiva decidir quais os dados que são de cada utilizador, excluindo a necessidade de utilizar um *beacon* com um identificador diferente (MAC) para isso.

Capítulo 4

Testes e Avaliação

Neste capítulo irá ser abordado os testes e a avaliação do trabalho desenvolvido durante o estágio e apresentado no capítulo 3. À semelhança dos capítulos anteriores, o capítulo está dividido em vários um por cada projecto. Todos os projetos obtiveram resultados positivos em relação aos objetivos abordados no início do estágio.

4.1 Nidus

Durante os desenvolvimentos ocorridos durante a realização do estágio foram testados as novas funcionalidades desenvolvidas, nomeadamente em relação à compressão dos ficheiros e das imagens.

A compressão das imagens e utilização dos SVG ao invés dos PNG e JPEG, como foi indicado no subcapítulo 3.2.4, foi implementada com sucesso e abriu novas possibilidades no sistema Nidus e permitiu igualmente para o seu utilizador uma melhor fluidez nas animações e nas transições.

A compressão dos ficheiros com recurso à mudança do método de compressão originalmente o GZIP para Brotli obteve bons resultados, como indicado na tabela 3.1, nos ficheiros atuais da Nidus e permitiu libertar espaço suficiente para alojar as restantes funcionalidades desenvolvidas no estágio tais como o sistema de internacionalização ou a adaptação futura do plugin Blockly na gestão de eventos.

4.2 Nb-Iot

Os testes realizados no *firmware* desenvolvido demonstraram a capacidade de este utilizar toda a memória disponível sem ocorrerem necessidade de alocar memória e não ser possível. Comparativamente com o *firmware* inicial, o qual não permitia algumas funcionalidades tais como o envio para o portal das suas configurações e apenas o envio das leituras e com um limite de 25 leituras de Log, o *firmware* desenvolvido possui todas as necessidades do projeto tais como a comunicação bi-direcional entre o portal e o equipamento teve nos testes realizados um limite de 170 Logs com o valor máximo de sensores estipulados(6) o qual pode ser configurado e caso este seja alterado

a possibilidades de aumentar a capacidade de Log. Para um máximo de 3 sensores o valor máximo de leituras em Log sobe para 340 e sobe para 1020 no caso de apenas ser alocado o espaço para um sensor.

4.3 Kea Tracker

Neste projeto o desenvolvimento limitou-se apenas à aplicação visto durante a rea-lização dos restantes projetos ter sido fornecido uma versão já com suporte às funcionalidades requeridas. Focou-se assim o projeto no desenvolvimento da aplicação multiplataforma a qual integrou todas as funcionalidades já existentes com sucesso e a inclusão das novas funcionalidades referentes ao *download* dos Logs e ao sincronismo de leituras que permite ao utilizador final uma precisão garantida ao minuto.

4.4 dot.tracker

A plataforma dot.Tracker durante os testes demonstrou-se cerca de 5 a 6 vezes mais rápida relativamente ao protótipo desenvolvido com recurso à *Framework* Laravel, além de corrigir os problemas identificados e apresentados ao longo deste relatório. Os filtros desenvolvidos/ aplicados conseguem com sucesso atenuar as oscilações presentes nas comunicações, garantindo um movimento mais estável e mais suave.

A adição da correção de altura demonstrou ser uma importante fase no cálculo da posição estimada do equipamento. Durante estes testes foi possível constatar, um problema na utilização do método atual para cálculo da distância visto que no caso de ser configurada uma altura na plataforma e o equipamento se mover no eixo Z correspondente à altura os resultados obtidos aumentam o erro consoante a diferença de altura real e a calculada. Ainda referente à altura no caso de oscilações muito grandes e em casos específicos o algoritmo descarta pacotes visto não terem sentido. Supondo o teste realizado com um *beacon* e alguns receptores. Configurando na plataforma a altura do *beacon* a 10 M de altura e a altura do receptor a 0 M de altura, caso seja rececionado com um pacote com o RSSI correspondente a uma altura perto de 10 M é possível obter a distância a colocar no mapa como elucidado na figura 3.14. Caso seja movido o *beacon* para uma altura de por exemplo 0.5 M o algoritmo desenvolvido ao compensar a distância com a diferença de alturas não é capaz de calcular esse valor visto que segundo o algoritmo de Pitágoras não é possível obter um triângulo rectângulo com um cateto de 10 M e a hipotenusa de 0.5 M. Nestes casos a plataforma não calcula a sua posição no mapa.

A respeito da precisão do sistema em geral é possível obter uma precisão média

4.4. DOT TRACKER

entre 1 a 2 metros consoantes os ambientes, o seu conteúdo ou simplesmente às interferências existentes. De modo a aumentar a precisão em ambientes mais hostis para localização o aumento do número de receptores e menor distância entre eles permite melhores resultados.

Capítulo 5

Conclusões e Trabalho Futuro

Inserido num contexto empresarial , e tratando-se de projetos em desenvolvimento e de desenvolvimento contínuo, estes não foram concluidos com o concluir deste relatório e estágio. Todos os objetivos definidos no inicio do estágio foram concluidos com sucesso e com bons resultados como apresentado no capítulo 4.

A realização de um estágio com projetos reais em ambiente empresarial permitiu uma melhor compreensão dos vários fatores que não são possíveis replicar em ambiente académico, além da utilização de novas metodologias e tecnologias.

Para trabalho futuro prevê-se a implementação do plugin Blockly na página da Nidus, algumas alterações simples de Design na página e a reformulação da ferramenta de compressão da página para a utilização do Brotli. O projeto do Nb-Iot caso exista atualizações de *Firmware* por parte da Digi que permitam a comunicação por Bluetooth, será implementada a configuração do equipamento através de uma aplicação móvel. No projeto dot.Tracker caso a necessidade de aumentar a precisão se venha a verificar, deve-se ponderar a migração para um sistema diferente de localização indoor sem recurso ao valor de RSSI tal como o *Bluetooth 5.1*, em desenvolvimento, especialmente desenhado para localização em ambientes fechados. O *Bluetooth 5.1* com recurso a várias antenas consegue calcular o ângulo entre o emissor e o receptor e com recurso a vários receptores delimitar a posição do emissor.

Bibliografia

- [1] N. T. . A. A. Hashem, “Low power wide area network (lpwan) technologies for industrial iot applications,” master thesis (in english), DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY, June 2018.
- [2] T. Instruments, “Bluetooth® low energy beacons.” online on 02/03/2020: <https://www.ti.com/lit/an/swra475a/swra475a.pdf>, Oct 2016.
- [3] “Comparação entre header http gzip vs brotli.” online on 10/07/2020:<https://medium.com/oyotech/how-brotli-compression-gave-us-37-latency-improvement-14d41e50fee4/>.
- [4] “Jwt website.” online on 01/08/2020:<https://jwt.io/>.
- [5] “Kalman filter.” online on 01/08/2020:<https://github.com/wouterbulten/kalmanjs>.
- [6] “Beacon fujitsu data sheet.” online on 13/03/2020: <https://www.fujitsu.com/downloads/MICRO/fcai/wireless-modules/fwm8blz02a-109069.pdf>.
- [7] “Beacon blue.” online on 13/03/2020: <https://bluemaestro.com/products/product-details/bluetooth-temperature-humidity-sensor-beacon>.
- [8] “Ruuvi tag data sheet.” online on 13/03/2020: <https://ruuvi.com/files/ruuvitag-tech-spec-2019-7.pdf>.
- [9] “Captemp ast.” online on 21/02/2020: <https://www.captemp.com/tecnologies.php>.
- [10] “Tinypng.” online on 13/03/2020: <https://tinypng.com/>.
- [11] “Digi xbee® 3 cellular lte-m/nb-iot.” online on 01/03/2020: <https://www.digi.com/products/embedded-systems/digi-xbee/cellular-modems/xbee3-cellular-lte-m-nb-iot>.
- [12] “Micropython.” online on 07/03/2020: <https://micropython.org/>.
- [13] “Micropython libraries.” online on 07/03/2020: <http://docs.micropython.org/en/latest/library/index.html#python-standard-libraries-and-micro-libraries>, Mar 2020.

- [14] “Narrowband iot (nb-iot).” online on 06/03/2020: <https://www.u-blox.com/en/narrowband-iot-nb-iot>, Oct 2017.
- [15] R. U. . K. P. . S. Mahesh, “Low power wide area networks: An overview.” online on 06/03/2020: <https://arxiv.org/pdf/1606.07360.pdf>, 2017.
- [16] “Ruuvi beacons.” online on 27/02/2020: <https://ruuvi.com/>.
- [17] “Ruuvi android app github.” online on 15/02/2020: <https://github.com/ruuvi/com.ruuvi.station>.
- [18] B. taskit GmbH, “Beacon line.” online on 28/04/2020: <http://www.beacon-line.com/>.
- [19] J. Alakuijala, A. Farruggia, P. Ferragina, E. Kliuchnikov, R. Obryk, Z. Szabadka, and L. Vandevenne, “Brotli: A general-purpose data compressor,” *ACM Transactions on Information Systems*, 2019.
- [20] “Rfc 7932.” online on 13/03/2020: <https://tools.ietf.org/html/rfc7932>.
- [21] J. Alakuijala, E. Kliuchnikov, Z. Szabadka, and L. Vandevenne, “Comparison of brotli, deflate, zopfli, lzma, lzham and bzip2 compression algorithms,” *Google Inc*, 2015.
- [22] S. M. S. Hilles and A. Abdulsalam, “Image Compression Techniques in Networking : Review Paper,” no. February, pp. 0–5, 2019.
- [23] “Mass image compressor.” online on 13/03/2020: <http://icompressor.blogspot.com/2016/10/introduction-to-hassle-free-image.html>.
- [24] Y. Wang, X. Yang, Y. Zhao, Y. Liu, and L. Cuthbert, “Bluetooth positioning using RSSI and triangulation methods,” *2013 IEEE 10th Consumer Communications and Networking Conference, CCNC 2013*, no. January, pp. 837–842, 2013.
- [25] H. Zhu, H. Li, Y. Li, H. Long, and K. Zheng, “Geometrical constrained least squares estimation in wireless location systems,” *Proceedings of 2014 4th IEEE International Conference on Network Infrastructure and Digital Content, IEEE IC-NIDC 2014*, no. November 2015, pp. 159–163, 2014.
- [26] A. I. Epoka, “Nb-iot sensors efento.” online on 13/03/2020: <https://getefento.com/kategoria-produktu/nb-iot-sensors/>.

BIBLIOGRAFIA

- [27] “Beacon with logs.” online on 13/03/2020: <https://www.edn.com/bluetooth-beacon-performs-data-logging/>.
- [28] “I18n.” online on 10/07/2020:<https://i18njs.com/>.
- [29] “Brotly vs gzip comparasion.” online on 10/07/2020:<https://www.opencpu.org/posts/brotli-benchmarks/>.
- [30] “Ferramenta svgo.” online on 10/07/2020:<https://github.com/svg/svgo>.
- [31] “Blockly.” online on 10/07/2020:<https://developers.google.com/blockly>.
- [32] “Micropython aes library repository.” online on 10/07/2020:<https://github.com/piaca/micropython-aes/>.
- [33] “Native script.” online on 25/07/2020:<https://nativescript.org/>.
- [34] “Ionic.” online on 25/07/2020:<https://ionicframework.com/>.
- [35] “Ruuvi log firmware.” online on 25/07/2020:<https://docs.ruuvi.com/communication/bluetooth-connection/nordic-uart-service-nus/log-read>.
- [36] “Specifications - company identifiers.” online on 16/08/2020:<https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers/>.
- [37] “Official ruuvi broadcast formats.” online on 16/08/2020:<https://github.com/ruuvi/ruuvi-sensor-protocols/blob/master/broadcast-formats.md>.
- [38] “Kalmanfilter2.” online on 16/08/2020:<https://asmedigitalcollection.asme.org/fluidsengineering/article/10/1/011001/10000/New-Approach-to-Linear-Filtering-and-Prediction>.
- [39] “Human walking speed.” online on 01/08/2020:<https://www.healthline.com/health/exercise-fitness/average-walking-speed>.
- [40] L. L. Long and M. Srinivasan, “Walking, running, and resting under time, distance, and average speed constraints: Optimality of walk-run-rest mixtures,” *Journal of the Royal Society Interface*, vol. 10, no. 81, 2013.
- [41] “Point-in-poligon.” online on 01/08/2020:<https://www.npmjs.com/package/point-in-polygon>.

Apêndice

Apêndice A

Exemplo de um SVG

```
1
2 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
3 <!-- Created with Inkscape (http://www.inkscape.org/) -->
4 <svg
5   xmlns:dc="http://purl.org/dc/elements/1.1/"
6   xmlns:cc="http://creativecommons.org/ns#"
7   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
8   xmlns:svg="http://www.w3.org/2000/svg"
9   xmlns="http://www.w3.org/2000/svg"
10  xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/
11      sodipodi-0.dtd"
12  xmlns:inkscape="http://www.inkscape.org/namespaces/
13      inkscape"
14  width="10mm"
15  height="10mm"
16  viewBox="0 0 10 10"
17  version="1.1"
18  id="svg8"
19  inkscape:version="0.92.5 (2060ec1f9f, 2020-04-08)"
20  sodipodi:docname="desenho1.svg">
21 <defs
22   id="defs2" />
23 <sodipodi:namedview
24   id="base"
25   pagecolor="#ffffff"
26   bordercolor="#666666"
27   borderopacity="1.0"
28   inkscape:pageopacity="0.0"
29   inkscape:pagemshadow="2"
```

```
28     inkscape:zoom="15.839192"
29     inkscape:cx="56.676533"
30     inkscape:cy="22.492236"
31     inkscape:document-units="mm"
32     inkscape:current-layer="layer1"
33     showgrid="false"
34     inkscape>window-width="1920"
35     inkscape>window-height="1017"
36     inkscape>window-x="-8"
37     inkscape>window-y="-8"
38     inkscape>window-maximized="1" />
39 <metadata
40     id="metadata5">
41     <rdf:RDF>
42         <cc:Work
43             rdf:about=""
44             <dc:format>image/svg+xml</dc:format>
45             <dc:type
46                 rdf:resource="http://purl.org/dc/dcmitype/
StillImage" />
47             <dc:title></dc:title>
48         </cc:Work>
49     </rdf:RDF>
50 </metadata>
51 <g
52     inkscape:label="Layer 1"
53     inkscape:groupmode="layer"
54     id="layer1"
55     transform="translate(0,-287)">
56 <rect
57     style="fill:#ff0000;stroke-width:0.26458332"
58     id="rect10"
59     width="9.1205721"
60     height="4.660512"
61     x="0.3674956"
62     y="287.3783" />
63 <circle
```

```
64         style="fill:#00ffff;stroke-width:0.26458332"
65         id="path12"
66         cx="3.5747297"
67         cy="293.07449"
68         r="2.3887215" />
69     <text
70         xml:space="preserve"
71         style="font-style:normal;font-weight:normal;font-
size:1.2726717px;line-height:1.25;font-family:sans-serif
;letter-spacing:0px;word-spacing:0px;fill:#000000;fill-
opacity:1;stroke:none;stroke-width:0.03181679"
72         x="8.6893291"
73         y="242.43219"
74         id="text18"
75         transform="scale(0.82155341,1.2172063)">
76     <tspan
77         sodipodi:role="line"
78         id="tspan16"
79         x="8.6893291"
80         y="242.43219"
81         style="stroke-width:0.03181679">SVG </tspan>
82     </text>
83 </g>
84 </svg>
```

Algoritmo A.1: Exemplo de um SVG

Apêndice B

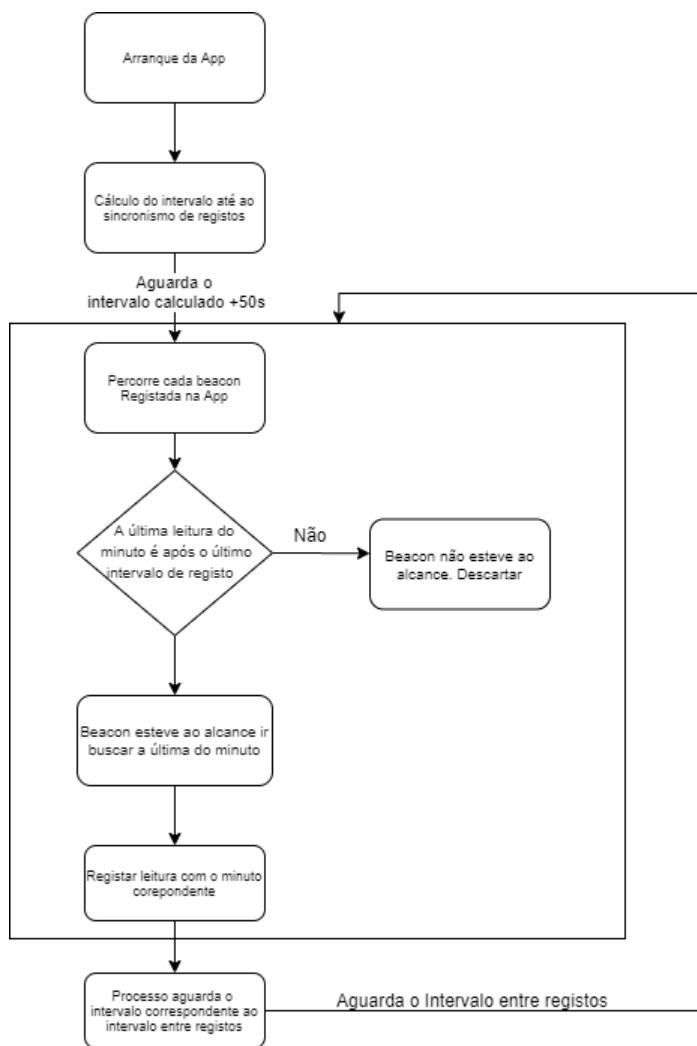
Exemplo de um SVG comprimido

```
1
2 <svg xmlns="http://www.w3.org/2000/svg" width="10mm" height
= "10mm" viewBox="0 0 10 10">
3   <g transform="translate(0 -287)">
4     <path fill="red" d="M.367 287.378h9.121v4.661H.367z
" />
5     <circle cx="3.575" cy="293.074" r="2.389" fill="#0
ff" />
6     <text style="line-height:1.25" x="8.689" y="242.432
" transform="scale(.82155 1.2172)" font-weight="
400" font-size="1.273" font-family="sans-serif"
letter-spacing="0" word-spacing="0" stroke-width
=.032">
7       <tspan x="8.689" y="242.432">SVG</tspan>
8     </text>
9   </g>
10 </svg>
```

Algoritmo B.1: Exemplo de um SVG comprimido

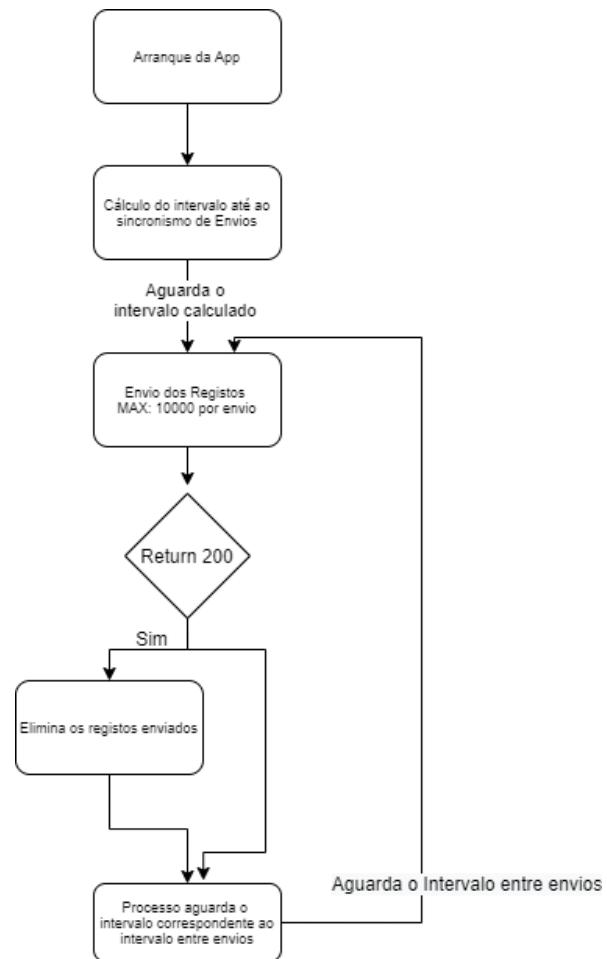
Apêndice C

Fluxograma dos processos de Registo -App



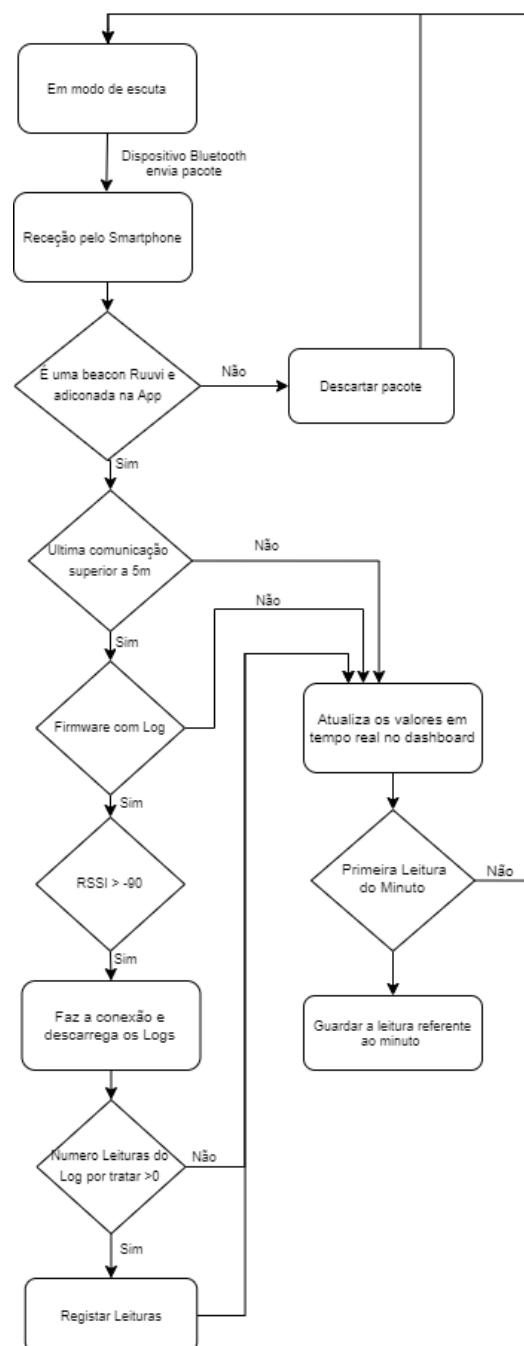
Apêndice D

Fluxograma do processo de Envio - App



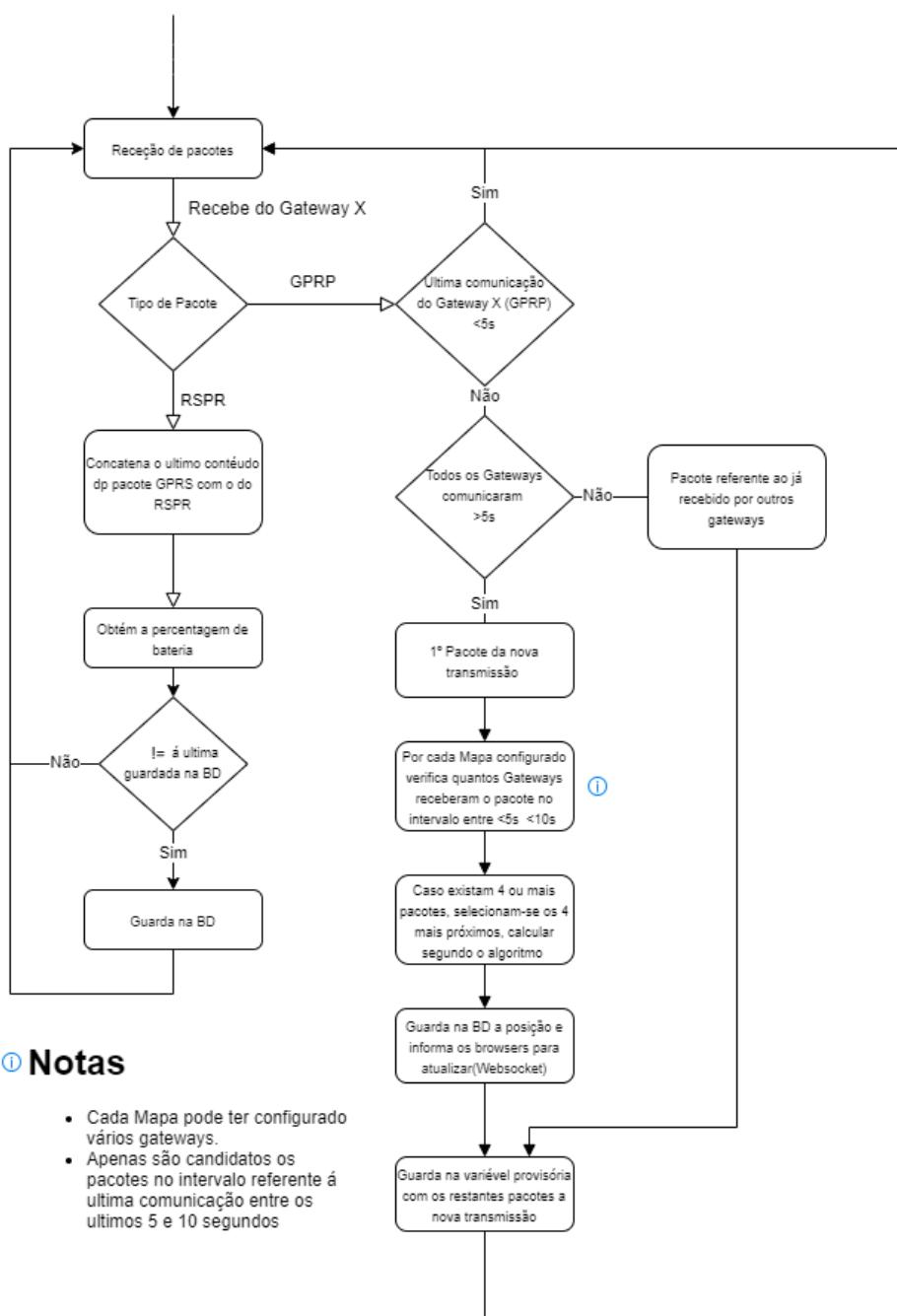
Apêndice E

Fluxograma de receção de Pacotes BLE



Apêndice F

Fluxograma do processo de receção de pacotes



ⓘ Notas

- Cada Mapa pode ter configurado vários gateways.
- Apenas são candidatos os pacotes no intervalo referente à ultima comunicação entre os ultimos 5 e 10 segundos

Apêndice G

Código da obtenção da posição - LSE

```
1 function getPosLSE(d1,d2,d3,d4,n1,n2,n3,n4,scale){  
2     if(scale==null){  
3         return -1;  
4     }  
5  
6     /*Convert values from String to Float*/  
7     scale=parseFloat(scale);  
8     d1=parseFloat(d1);  
9     d2=parseFloat(d2);  
10    d3=parseFloat(d3);  
11    d4=parseFloat(d4);  
12  
13    /*matriz A  
14        | a   d |  
15        | b   e |  
16        | c   f |  
17    */  
18    var a=(n2.x)-(n1.x);  
19    var b=(n3.x)-(n1.x);  
20    var c=(n4.x)-(n1.x);  
21    var d=(n2.y)-(n1.y);  
22    var e=(n3.y)-(n1.y);  
23    var f=(n4.y)-(n1.y);  
24  
25    /*transposta de A  
26        | a   b   c |  
27        | d   e   f |  
28  
29        At * a  
30        | aa  bb  |  
31        | cc  dd  |  
32    */  
33  
34    var aa=(a*a)+(b*b)+(c*c);  
35    var bb=(a*d)+(b*e)+(c*f);
```

```
36     var cc=bb; /*(d*a)+(e*b)+(f*c);*/
37     var dd=(d*d)+(e*e)+(f*f);
38
39     //determinante de At*A
40
41     var det=(aa*dd)-(bb*cc);
42     if(det!=0)
43     {
44         /*pode proceguir ha inversa
45
46         inversa
47             | aaa   bbb |
48             | ccc   ddd |
49
50         formula
51             | dd/det -bb/det |
52             | -cc/det aa/det |
53         */
54     var aaa=dd/det;
55     var bbb=-bb/det;
56     var ccc=-cc/det;
57     var ddd=aa/det;
58
59     // Multiplicar por 1/2
60     var aaaa=aaa*0.5;
61     var bbbb=bbb*0.5;
62     var cccc=ccc*0.5;
63     var dddd=ddd*0.5;
64
65     /* multiplicar pela transposta
66         | aaaa bbbb | * | a b c |
67         | cccc dddd |       | d e f |
68     */
69
70     var a5=(aaaa*a)+(bbbb*d);
71     var b5=(aaaa*b)+(bbbb*e);
72     var c5=(aaaa*c)+(bbbb*f);
73     var d5=(cccc*a)+(ddda*d);
74     var e5=(cccc*b)+(ddda*e);
75     var f5=(cccc*c)+(ddda*f);
76
77     /* definir b
78         | b1 |
79         | b2 |
80         | b3 |
```

```

81      */
82
83      var d12=Math.pow((d1*scale),2);
84      var b1=Math.pow((n2.x),2) - Math.pow((n1.x),2) + Math.pow((n2.
85          y),2) - Math.pow((n1.y),2) - Math.pow((d2*scale),2) + d12;
86      var b2=Math.pow((n3.x),2) - Math.pow((n1.x),2) + Math.pow((n3.
87          y),2) - Math.pow((n1.y),2) - Math.pow((d3*scale),2) + d12;
88      var b3=Math.pow((n4.x),2) - Math.pow((n1.x),2) + Math.pow((n4.
89          y),2) - Math.pow((n1.y),2) - Math.pow((d4*scale),2) + d12;
90
91
92      /* Multiplicar por b
93      | a5 b5 c5 | * | b1 |
94      | d5 e5 f5 |   | b2 |
95      |           |   | b3 |
96
97      */
98      var resX=(a5*b1)+(b5*b2)+(c5*b3);
99      var resY=(d5*b1)+(e5*b2)+(f5*b3);
100     return {"x":resX,"y":resY};
101 }
102
103 return -1;
104 }
```

Algoritmo G.1: Implementação do algoritmo LSE em Javascript