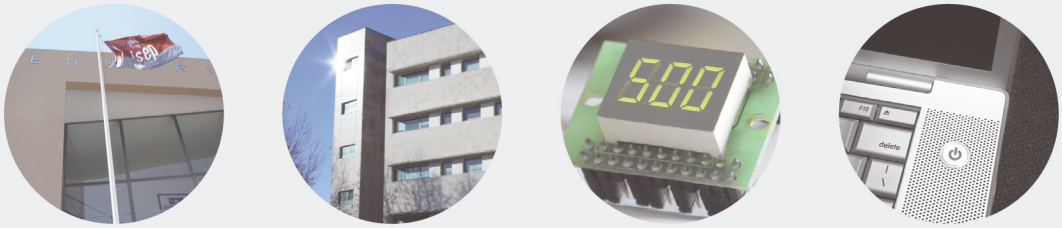
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Infraestrutura de Sensorização Partilhada para Cidades Inteligentes

UBIRAJARA FABIANO MACENA BARBOSA DE ALMEIDA
novembro de 2018

POLITÉCNICO
DO PORTO

Infraestrutura de Sensorização Partilhada para Cidades Inteligentes

Ubirajara Fabiano Macena Barbosa de Almeida

Departamento de Engenharia Electrotecnica

Instituto Superior de Engenharia do Porto

2018

Este relatório cumpre os requisitos contidos na cadeira de TEDI,
do 2º ano do Mestrado em Engenharia Elétrica e,
Computadores, Sistemas Autonomos

E-mail:

1120769@isep.ipp.pt

Orientador ISEP: Dr. Lino Figueiredo, lbf@isep.ipp.pt

Empresa: CEiiA

Orientador CEiiA: Eng. Emanuel Costa, emanuel.costa@ceiia.com

Departamento de Engenharia Electrotecnica

Instituto Superior de Engenharia do Porto

13 de Novembro de 2018

Agradecimentos

Este projeto não seria possível sem o apoio de várias pessoas, que merecem um agradecimento especial pela ajuda e tempo disponibilizado.

Desde já, quero agradecer à minha família, em especial aos meus pais por todo o esforço e dedicação e por me proporcionarem todas as condições necessárias para continuar a lutar todos os dias para atingir os meus objetivos, à minha namorada e à minha irmã que sempre acreditaram em mim. Obrigado a todos pelo constante apoio e paciência em todos os momentos.

Ao Centro para Excelência e Inovação na Industria Automóvel (CEiiA), por me ter dado a oportunidade e ferramentas necessárias para desenvolver a tese num meio empresarial. Ao grupo de trabalho do Laboratório de Eletrónica, que sempre esteve pronto e dedicado a ajudar-me, com um espírito de equipa notável.

Por fim, gostaria agradecer aos meu orientadores, no CEiiA e no Instituto Superior de Engenharia do Porto (ISEP), Engº Emanuel Costa e Engº Lino Figueiredo.

This page intentionally left blank.

Resumo

O objetivo desta tese consistiu em criar um sistema para ser integrado na atual infraestrutura da rede Mobi.me, que é uma plataforma que permite uma gestão centrada no Utilizador de serviços de mobilidade, que seja capaz de monitorizar parâmetros relevantes de cada cidade para os seus cidadãos e gestores, obtendo um mapa de dados de uma cidade ou região. Este sistema conecta dados e pessoas alavancando uma plataforma relevante em termos de indicadores de uma cidade de forma a permitir uma melhor gestão da mesma e um aumento da qualidade de serviços disponíveis.

Com a ajuda da evolução da micro eletrotónica e das tecnologias de comunicação desenvolveu-se uma Caixa de Sensores *low power* e modular a ser instalada em pontos fixos da cidade ou em veículos de forma a monitorizar uma região de interesse.

Os dados recolhidos são enviados para um Servidor que analisa e demonstra toda informação através de um interface web. Para fazer com que estes dados cheguem ao Servidor potencializou-se os produtos existentes na empresa e os utilizadores como *gateways* para servirem de ponte entre a Caixa de Sensores e o Servidor.

Os consumos de energia da Caixa de Sensores, bem como, os resultados obtidos em termos de comunicação entre a mesma e os *gateways* foram bastante satisfatórios.

Com a criação deste sistema foi possível construir um mapa de dados demonstrado através de um interface web, como requerido para primeiro protótipo.

Palavras-chave

Cidades Inteligentes, Evolução tecnológica, Partilha de dados, Caixa de Sensores.

Abstract

The objective of this thesis was to create a system to be integrated into the current infrastructure of the Mobi.me network, which is a user-centered platform that allows management of mobility services, being able to monitor relevant parameters of each city for its citizens and managers, getting a data map of a city or region. This system connects data and people by leveraging a relevant platform in terms of indicators of a city in order to allow a better management of the city and an increase in the quality of services available.
With the evolution of microelectronics and communication technologies, it was developed a modular low-power Sensor Box to be installed at fixed points in the city or in the vehicles in order to monitor a region of interest.
The data collected must be sent to a server that analyzes and demonstrates all information through a web interface. In order to make this data reach the server, it is intended to use and leverage the existing company products and the users as gateways to serve as a bridge between the Sensor Box and the Server.
The energy consumption of the Sensor Box, as well as, the results obtained in terms of communication between it and the gateways, were quite satisfactory.
With the creation of this system it was possible to build a data map through a web interface, as required for the first prototype.

Keywords

Smart Cities, Technological Evolution, Data Sharing, Sensor Box.

This page intentionally left blank.

Conteúdo

This page intentionally left blank.

Lista de Figuras

This page intentionally left blank.

Lista de Tabelas

This page intentionally left blank.

Lista de Acrónimos

3GPP *3rd Generation Partnership Project*

AC *Alternating Current*

ADC *Analog to Digital Converter*

AES *Advanced Encryption Standard*

AoT *Array of Things*

API *Application Programming Interface*

ARM *Advanced RISC Machine*

ATT *Attribute Protocol*

BLE *Bluetooth Low Energy*

CAN *Controller Area Network*

CBOR *Concise Binary Object Representation*

CCCD *Client Characteristic Configuration Descriptor*

CEiiA Centro para Excelência e Inovação na Industria Automóvel

COV Compostos Orgânicos Voláteis

CPU *Central Processing Unit*

CSMA-CA *Carrier sense multiple access with collision avoidance*

DC *Direct Current*

DPPM *Dynamic Power-Path Management*

ECDH *Elliptic-curve Diffie–Hellman*

eDRX *Adjustable Extended Discontinuous Reception*

FET *Field Effect Transistor*

GAP *Generic Access Profile*

GATT *Generic Attribute Profile*

GSM *Global System for Mobile Communications*

GPIO *General Purpose Input/Output*

GPS Sistema de Posicionamento Global

GPRS *General Packet Radio Service*

HAN *Home Area Network*

HCI *Host Controller Interface*

HTTP *Hypertext Transfer Protocol*

I2C *Inter-Integrated Circuit*

IC *Integrated Circuit*

ID Identificador

IEEE *Institute of Electric and Electronic Engineers*

IoT *Internet of Things*

IOS Sistema operacional Móvel da Apple

IP *Internet Protocol*

ISEP Instituto Superior de Engenharia do Porto

ISI *Institute for Scientific Interchange*

ISM *Industrial, Scientific and Medical*

JSON *JavaScript Object Notation*

L2CAP *Logical Link Control and Adaptation Protocol*

LED *Light-Emitting Diode*

LoRa *Long Range*

LoRaWAN *Long Range Wide Area Network*

LPWAN *Low Power Wide Area Network*

LTE *Long Term Evolution*

MAC *Media Access Control*

MCU *Microcontroller*

MDC *Mobility Device Controller*

MEEC Mestrado em Engenharia Eletrotécnica e de Computadores

MIC *Message Integrity Code*

MOSFET *Metal Oxide Semiconductor Field Effect Transistor*

MQTT *Message Queuing Telemetry Transport*

NB-CIoT *Narrowband Cellular IoT*

NB-IoT *Narrowband IoT*

NB-LTE *Narrow-Band Long-Term Evolution*

NTP *Network Time Protocol*

OMS Organização Mundial de Saude

PCB *Printed Circuit Board*

PDU *Protocol Data Unit*

PM2.5 Partículas inaláveis, de diâmetro inferior a 2,5 micrómetros

PM10 Partículas inaláveis, de diâmetro inferior a 10 micrómetros

PSM *Power Saving Mode*

QoS *Quality of Service*

RAM *Random Access Memory*

RF Rádio Frequência

ROM *Read Only Memory*

SDA *Serial Data*

SDC *Smart Device Cloud*

SDK *Software Development Kit*

SoC *System on a chip*

SPI *Serial Peripheral Interface*

SSC *Smart Service Cloud*

SSL *Secure Socket Layer*

SVC *SuperVisor Calls*

TCP *Transmission Control Protocol*

UART *Universal Asynchronous Receiver/Transmitter*

UDP *User Datagram Protocol*

uMDC *Micro Mobility Device Controller*

USB *Universal Serial Bus*

UUID *Universally Unique Identifier*

This page intentionally left blank.

Capítulo 1

Introdução

Este documento relata todo o desenvolvimento do projeto e explica todas as decisões tomadas durante o mesmo, no âmbito da unidade curricular de TEDI, do 2º ano do MEEC realizado na empresa CEiiA.

1.1 Contexto

Este projeto surge da necessidade do CEiiA de desenvolver uma nova funcionalidade do já existente sistema Mobi.me, a sensorização de cidades. O sistema Mobi.me, descrito no capítulo 4, consiste numa plataforma inteligente preparada para gerir e obter informações sobre sistemas de partilha e *tracking* de veículos elétricos, tendo como grande intuito construir uma cidade inteligente em termos de mobilidade. Este sistema pretende alavancar a rede Mobi.me de forma a permitir uma melhor gestão da região de interesse e um aumento da qualidade de serviços disponíveis, como por exemplo, se esta plataforma começar a monitorizar as concentrações de Dióxido de Carbono (CO2) numa região de interesse e em horas de maior tráfego detetar maior concentração do gás, provavelmente, é um local com maior fluxo de veículos

a combustão o que seria uma boa região para a implementação de um serviço de aluguer de veículos elétricos em termos qualidade de vida, ambientais e de negócio.

Visto que atualmente o maior fornecedor de dados desta rede são os próprios veículos através de equipamentos de *sharing* ou *tracking* neles instalados, é importante que o dispositivo de sensorização a desenvolver comunique com os equipamentos já existentes de modo a aumentar a qualidade de informação de todo o sistema e a possibilidade de construir um mapa de uma região ou cidade. Também se pretende potenciar o utilizador como *gateway* de informação do sistema, atualmente apesar desse canal estar estabelecido não é aproveitado para recolher informação, como acontece por exemplo nas aplicações de trânsito em que os Utilizadores têm a opção de reportar acidentes em tempo real.

O tipo de sensores a utilizar neste sistema pode variar consoante o que se pretende avaliar em cada cidade, como por exemplo a qualidade do ar, a temperatura, contagem de veículos em entradas e saídas da cidade, entre outros, adequando-se assim às necessidades de cada região/cidade. O equipamento a desenvolver será instalado nos veículos e em diversos pontos fixos de uma cidade, como por exemplo, semáforos, postes de iluminação, paragens de autocarro, lugares de estacionamento. É essencial que seja um produto *low-power*, pois alguns locais podem ter restrições de alimentação.

Os dados recolhidos devem ser centralizados e demonstrados de maneira simples e compreensível ao utilizador.

1.2 Requisitos do Sistema

Numa fase inicial foram definidos os requisitos que o sistema a implementar deve reunir, tais como:

- A Caixa de Sensores deve:
  - Ter uma unidade de processamento;
  - Disponibilizar várias interfaces de comunicação para conectar os sensores, tais como, *Analog to Digital Converter* (ADC), *Inter-Integrated Circuit* (I2C), *Serial Peripheral Interface* (SPI) e *Universal Asynchronous Receiver/Transmitter* (UART);
  - Não estar limitada ao nível de alimentação da unidade de processamento;
  - O *software* deve ser flexível de modo a que a introdução de um novo sensor tenha um impacto reduzido;
  - Ser capaz de comunicar com o *Micro Mobility Device Controller* (uMDC) e com o *Smartphone*;
  - Ser energeticamente autossustentável, ou seja, não ser necessário questões de manutenção devido à falta de carga na bateria;
  - Ser compacta e de fácil instalação.
- Recolher dados de diferentes tipos de sensores periodicamente;
- Permitir configurar os intervalos de leitura de cada um dos sensores, a periodicidade do envio dos dados e a janela de *Advertising*;
- Evitar monopolização do serviço por utilizadores desconhecidos;
- Potencializar o utilizador como fornecedor de informação do sistema;
- Construir métricas dos dados recolhidos;
- Ter uma interface gráfico intuitivo que contenha os dados recolhidos pelo sistema.

1.3 Casos de Uso

Para poder recolher e demonstrar os dados de uma cidade ou região é importante definir o principal caso de uso com o qual o sistema mais vezes vai ser confrontado.

A Caixa de Sensores deve ser instalada num veículo e alimentada através da bateria do mesmo ou num ponto fixo de interesse com um painel solar de forma a ser energeticamente autossustentável.

Os dados serão recolhidos periodicamente, o ciclo de leitura/*report* dos dados e o número de leituras de cada sensor devem ser configuráveis. Estes dados devem ser reportados para o uMDC ou para os utilizadores através do seu *Smartphone*, se possível.

Quando o uMDC ou o utilizador através do *Smartphone* obtiverem os dados reportados pelas Caixas de Sensores, estes devem ser encaminhados para o servidor da Caixa de Sensores para poderem ser analisados e demonstrados.

1.4 A Empresa

O CEiiA já existe no mercado há mais 10 anos, e é uma instituição que promove a inovação através da engenharia, com o objetivo de aumentar os meios de mobilidade de forma sustentável e competitiva sem prejudicar o ambiente.

Pretende ser, uma referência local e mundial a nível de investigação, desenvolvimento, produção e testes de produtos/serviços para a indústria da mobilidade em todos os seus ramos, a nível rodoviário com o desenvolvimento do sistema Mobi.me, aeronáutico com a colaboração no projeto KC-390 [1], e marítimo com o desenvolvimento do Medusa Deep Sea [2], entre outros projetos que alavancam a mobilidade em todas as suas vertentes.

Relativamente à equipa onde este projeto foi realizado, é a equipa de siste-

mas embebidos que trabalha no desenvolvimento de projetos/serviços para que o CEiiA cumpra os seus objetivos na indústria da mobilidade rodoviária.

1.5 Objetivos

Com o desenvolvimento desta dissertação pretende-se desenvolver um sistema que ao ser integrado no Mobi.me, permita sensorizar parâmetros relevantes de cada cidade para os seus cidadãos e gestores, obtendo um mapa de dados de uma cidade ou região. Para atingir o principal objetivo deve-se passar por um conjunto de passos intermédios, tais como:

- Análise de produtos e soluções existentes no mercado;
- Definição da Arquitetura do sistema;
- Desenho de uma Caixa de Sensores;
- Implementação da recolha de dados através dos sensores utilizados na Caixa de Sensores;
- Implementação do envio dos dados recolhidos;
- Potencializar o utilizador final e os produtos já existentes na empresa para a recolha dos dados disponibilizados pela caixa de sensores;
- Demonstração e análise dos dados.

1.6 Planeamento

De modo a cumprir com os objetivos propostos na secção 1.5 numa fase inicial do projeto foi delineado um plano para atingir os mesmos demonstrado no Anexo A.

Numa primeira fase é importante fazer uma pesquisa sobre o atual estado da arte e produtos existentes no mercado. Posteriormente é necessário definir a arquitetura do sistema como um todo e da Caixa de Sensores. Após a definição da arquitetura segue-se a fase de implementação da Caixa de Sensores, recolha dos dados e envio dos mesmos para o restante sistema através de um protocolo de comunicação definido na arquitetura do sistema, potencializando os utilizados e produtos existentes como *gateways*. Os dados recolhidos devem ser centralizados, analisados e demonstrados através de um interface web para o utilizador final. Na fase final devem ser realizados testes que provem o princípio de funcionamento de todo o sistema com a implementação de Caixas de Sensores Fixas e Móveis. A escrita deste relatório deve ser feita em paralelo com todo este desenvolvimento.

1.7 Estrutura

Este relatório foi estruturado de maneira a ser o mais explícito possível, espelhando todas as decisões e descrições do que foi estudado e desenvolvido. Após este capítulo é feita uma análise ao estado da arte de modo a identificar a importância das cidades inteligentes e projetos existentes nesta vertente. De seguida apresenta-se um capítulo sobre tecnologias de comunicação que podem ser utilizadas neste tipo de sistemas.

O quarto capítulo pretende apresentar a arquitetura geral do sistema existente na empresa. Por sua vez, o quinto capítulo demonstra como este projeto vai ser integrado na arquitetura existente, apresenta todas opções tomadas no decorrer da tese. O sexto capítulo explica como este sistema foi implementado em termos de *hardware* e *software* com detalhes e descrições técnicas da implementação.

No sétimo capítulo pretende-se validar o funcionamento do que foi implementado no capítulo anterior, descrevendo possíveis cenários de testes e re-

sultados. Por fim, no último capítulo é apresentada uma análise ao que foi desenvolvido e quais os eventuais passos futuros.

This page intentionally left blank.

Capítulo 2

Estado da Arte

Este capítulo apresenta uma pesquisa de mercado incidindo sobre sistemas e equipamentos que possam ter características ou funcionalidades semelhantes ao projeto a desenvolver.

2.1 Cidades Inteligentes

As cidades são os principais centros da atividade humana e económica. Permitem grandes oportunidades de desenvolvimento aos seus habitantes como por exemplo, melhores empregos, educação, cuidados médicos, acesso a cultura, entre outros. No entanto, também geram uma ampla gama de problemas que podem ser difíceis de enfrentar à medida que crescem em tamanho e complexidade como por exemplo, excesso de população, aumento do consumo de energia, poluição entre outros.
Devido ao desenvolvimento de novas tecnologias tanto no campo da comunicação como da sensorização o conceito de cidades inteligentes surge como um meio para alcançar cidades mais eficientes e sustentáveis. O cenário atual exige que as cidades encontrem modos mais eficazes de gerir os novos desafios.

Cidades em todo o mundo começaram a procurar soluções que possibilitem em tempo real conexões de transportes e serviços urbanos de alta qualidade, leitura e transmissão dos consumos dos recursos naturais (e.g. água, luz e gás), monitorização da qualidade do ar, entre outros, com o objetivo de obter uma melhor qualidade de vida com efeitos positivos a longo prazo sobre a economia. Diferentes métodos e índices de medição foram desenvolvidos de acordo com os vários significados do conceito de cidade inteligente. Os sistemas de classificação através de indicadores quantitativos sintéticos estão a requerer mais atenção entre os decisores políticos, para decidir onde concentrar o tempo e os recursos, bem como para comunicar o desempenho da cidade aos seus cidadãos, visitantes e investidores. [3], [4], [5]
Outro ponto importante na nova demanda das cidades inteligentes é a interação do utilizador com os sistemas de monitorização quer na vertente de visualização do sistema, como por exemplo a possibilidade de analisar o trânsito em tempo real, quer na vertente de fornecedor de informações ao sistema, como por exemplo, dar indicação ao sistema que em determinado ponto houve um acidente.

O tipo de sensorização utilizada em cidades inteligentes permite monitorizar:

- Tráfego pedonal e rodoviário;
- Lugares de estacionamento disponíveis;
- Vibrações em pontes e monumentos;
- Ruído em zonas urbanas;
- Poluição atmosférica;
- Possibilidade de incêndios;
- Poluição dos rios;

- Consumo e gestão de energia pública;
- Auto-estradas (aviso de condições climatéricas ou acidentes aos utilizadores). [6], [7]

Tendo em conta os tópicos analisados na monitorização de cidades inteligentes, a poluição atmosférica é um tema muito visado na atualidade principalmente devido às consequências causadas pelo decréscimo da sua qualidade. O ar pode apresentar-se mais ou menos poluído por substâncias gasosas, líquidas ou sólidas, de origem natural ou antropogénica (resultantes da atividade humana), sendo que as emissões de origem antropogénica representam grande relevância na degradação da qualidade do ar. Com o crescimento das cidades bem como a expansão demográfica, a qualidade do ar que respiramos tem sido significativamente alterada, uma vez que, tal crescimento leva à redução dos espaços verdes, aumento das indústrias, maior utilização dos veículos a combustão, aumento de alguns processos agrícolas nocivos, entre outros.

Ao alterar a qualidade do ar, o ecossistema e os fatores que o envolvem como o clima, o solo e a água também são alterados, provocando assim, diversos fenómenos como por exemplo, o efeito estufa, a chuva ácida, a inversão térmica e a destruição da camada de ozono. Além dos fenómenos naturais, segundo Organização Mundial de Saude (OMS) [8], a poluição atmosférica pode provocar no ser humano ardor nos olhos, irritação na garganta, náuseas, vómitos ou até desmaios. A exposição contínua e prolongada à poluição atmosférica pode levar o indivíduo a desenvolver doenças crónicas como doenças pulmonares e cardiovasculares ou o enfraquecimento do sistema imunológico. Estas consequências são agravadas em indivíduos com doenças respiratórias como a asma ou alergias. Na figura 2.1, podemos ver um exemplo extremo de poluição urbana em Pequim. [9], [10]

Figura 2.1: Poluição Urbana [11]

Alguns dos principais gases poluentes provenientes da acção humana são:

- Óxidos de Nitrogénio (NOx) - Os mais relevantes como poluentes atmosféricos são o Óxido Nítrico (NO) e o Dióxido de Nitrogénio (NO2) embora apenas este último seja alvo de regulamentação. Em áreas urbanas a principal fonte de NOx são os veículos automóveis, pelo que as concentrações deste poluente acompanham geralmente as variações do tráfego automóvel. Nos veículos automóveis a emissão de NOx ocorre maioritariamente sob a forma de NO.

- Ozono (O3) - Os poluentes primários NOx e COV dão origem à formação do O3 são essencialmente resultantes das emissões dos veículos automóveis e de determinadas atividades industriais. Nas áreas urbanas, na proximidade das fontes emissoras, o NO emitido pelos veículos automóveis pode reagir com o O3, reduzindo-se assim localmente as concentrações deste poluente.

- Partículas inaláveis, de diâmetro inferior a 2,5 micrómetros (PM2.5) e Partículas inaláveis, de diâmetro inferior a 10 micrómetros (PM10) - As partículas são um conjunto complexo de substâncias, que se encontram em suspensão na atmosfera, sob a forma líquida ou sólida. Estas partículas são emitidas para a atmosfera a partir de uma gama variada de fontes antropogénicas sendo as mais importantes a queima de

combustíveis fósseis, o tráfego rodoviário (desgaste de travões e pneus) e determinados processos industriais. Em geral, os veículos a gasóleo emitem uma quantidade maior de partículas finas, por veículo, do que os veículos a gasolina. Quanto mais pequenas as partículas, maior o risco de induzirem efeitos negativos.

- Monóxido de Carbono (CO) e Dióxido de Carbono (CO2) - O CO e o CO2 são muitas vezes confundidos um com o outro. A diferença é que o CO2 é um gás comum, natural, necessário para toda a vida vegetal e animal. Por sua vez, o CO não é comum e é geralmente um subproduto da combustão de combustível com pouco oxigénio (combustão incompleta). O processo de combustão num veículo emite CO e CO2. Em meio urbano, o tráfego automóvel é a principal fonte de CO e CO2 sendo as zonas de tráfego intenso as que apresentam concentrações mais elevadas destes poluentes. As condições de circulação, tráfego mais ou menos fluido, também influenciam as concentrações, dado que as emissões de CO e CO2 são inversamente proporcionais à velocidade de circulação. Sendo o CO2 o gás mais poluente da atmosfera [12], [13].

- Dióxido de Enxofre (SO2) - é emitido no momento da queima de combustíveis fósseis. As principais fontes são as centrais térmicas, as grandes instalações de combustão industriais e as unidades de aquecimento domésticas. As emissões provenientes dos veículos têm vindo a baixar com a diminuição progressiva do enxofre nos combustíveis.

- Compostos Orgânicos Voláteis (COV) - enquadram n compostos sendo o benzeno objeto de regulamentação. Entram na composição dos combustíveis mas também na de diversos produtos de uso corrente como as tintas, colas, cosméticos, solventes, detergentes de limpeza de uso doméstico, profissional ou industrial. Estes compostos são emitidos

durante a sua combustão (nomeadamente nos gases de escape dos veículos rodoviários), ou por evaporação no momento da sua produção, armazenamento e utilização. [14], [15], [16]

Cada vez mais a monitorização e a previsão da poluição atmosférica são tarefas fundamentais. Atualmente existem dezenas de soluções no mercado para a monitorização da qualidade do ar entre outros fatores preponderantes na qualidade do ecossistema. De seguida, são apresentados vários projetos relacionados com cidades inteligentes alguns dos quais direcionados para a vertente da qualidade do ar [17] [18].

2.2 Programa Sharing Cities

O programa *Sharing Cities* (programa da União Europeia, até 31 de dezembro de 2020) visa alterar irreversivelmente a forma como pensamos sobre o papel da tecnologia digital, *Big Data* e *Internet of Things (IoT)* são ideias-chave do conceito, nas cidades e esclarecer como todos nós podemos beneficiar e contribuir para esse processo de transformação. Em três locais estratégicos - Londres, Lisboa e Milão - vai ser demonstrada a eficácia das novas tecnologias na melhoria da mobilidade urbana, aumento da eficiência energética dos edifícios e redução das emissões de carbono.
O uso das plataformas digitais e as decisões assentes em grandes massas de dados permitirão tomar decisões estratégicas da cidade quase em tempo real e aproximar o município, cidadãos e empresas. O programa articula-se em torno de três grandes eixos: pessoas, plataformas digitais e lugares/infraestruturas como demonstra a figura 2.2.

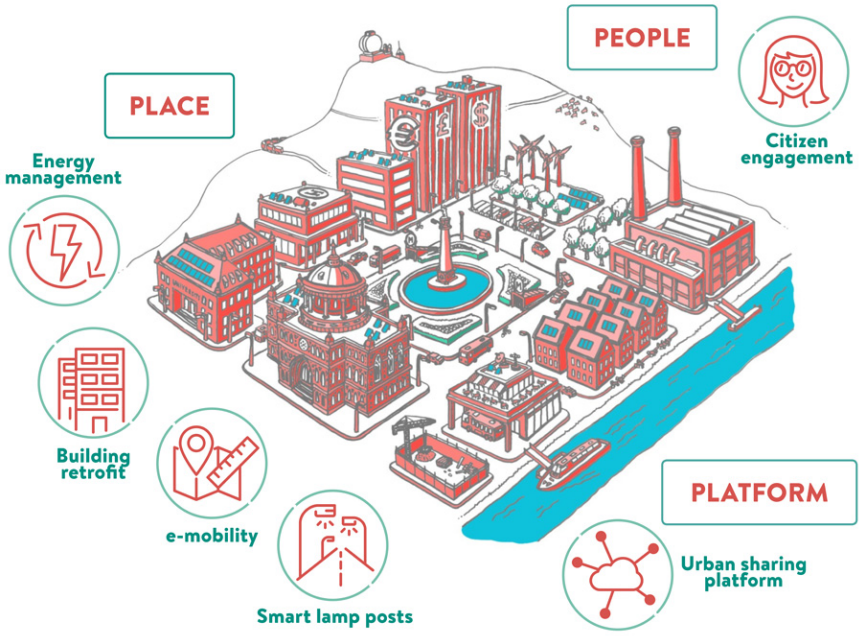


Figura 2.2: Diagrama do projecto Sharing Cities [19]

No âmbito deste programa, Lisboa elaborou uma estratégia de desenvolvimento urbano para as próximas décadas (Programa Operacional Regional de Lisboa 2020). Os principais objetivos da estratégia são atrair mais habitantes, melhorar a qualidade de vida na cidade através de medidas de eficiência energética, mobilidade e coesão social, dinamizar a economia e aumentar o emprego, atraindo mais empresários e ampliando o acesso ao ensino superior. Para atingir estes objetivos o programa foca alguns pontos essenciais:

- Reabilitar edifícios, que além de os modernizar fisicamente, dotá-los-á ainda de fontes de energia renovável, o que permite sistemas de gestão integrada e otimizada de energia;

- Soluções de mobilidade elétrica partilhada, quer de veículos automóveis, quer de bicicletas;

- Postes de iluminação inteligentes;
- Ferramentas para a aquisição de dados relativos à cidade.

As pessoas são decisivas em todo este processo. A participação ativa dos cidadãos não é só uma exigência, como também é uma necessidade, para isso, prevê-se também que sistemas e aplicações possam funcionar em regime de dados e código disponíveis para todos. Isto permite agregar contributos de todos e fortalecer a participação. O consórcio das Sharing Cities integra 35 parceiros, oriundos de todos os países envolvidos incluindo o CEiiA. [19], [20], [21], [22]

2.3 Sistemas de Sensorização Fixos para Cidades Inteligentes

Após uma introdução ao que se pretende nesta nova demanda sobre cidades inteligentes, segue-se a apresentação de vários projetos em que cada um utiliza uma abordagem proprietária para contribuir para uma cidade ou região mais inteligente. As plataformas que são apresentadas de seguida foram projetadas de modo a serem fixadas em pontos estratégicos da cidade ou região. Deste modo algumas destas soluções não têm restrições energéticas.

2.3.1 *Array of Things*

O *Array of Things (AoT)* é um projeto de sensorização urbana, onde uma rede de caixas de sensores interativa será instalada em Chicago para recolher dados em tempo real sobre ambiente, infraestrutura e atividades da cidade para pesquisas e uso público. O AoT servirá essencialmente como um *smartwatch fitness* para a cidade, mede fatores que afetam a habitabilidade

em Chicago tais como, o clima, a luminosidade, o ruído e a qualidade do ar, este sistema também contempla uma câmara para processamento de imagens. Todos estes dados são processados localmente e podem ser enviados via *WiFi*. Na figura 2.3 estão representados os módulos para aquisição e processamento de dados, bem como um exemplar de um protótipo da caixa de sensores. Estes dispositivos foram projetados para serem instalados em postes de iluminação ou no exterior de edifícios de modo a reduzir as restrições energéticas.

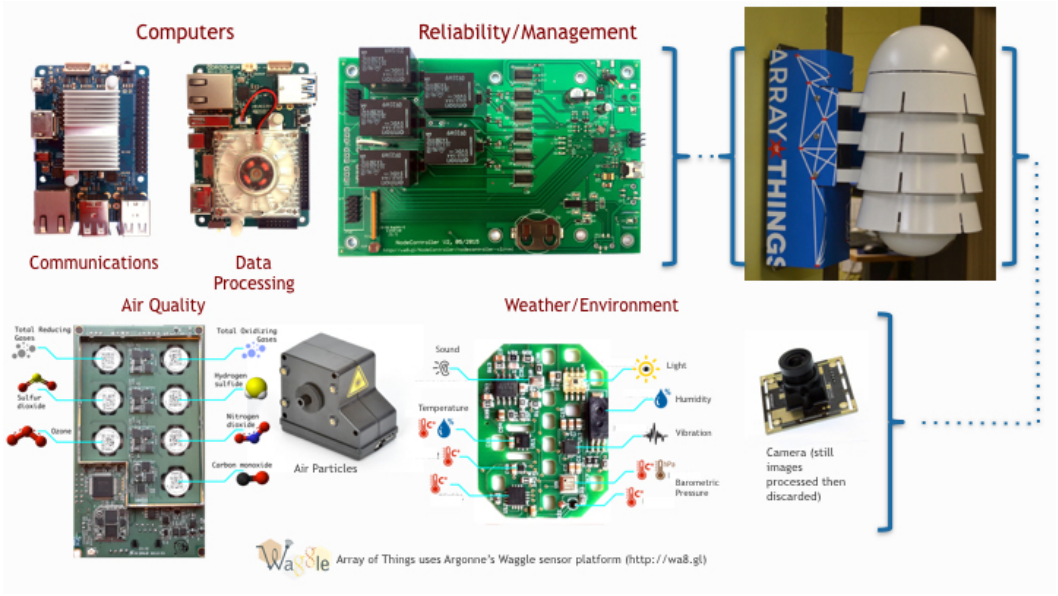
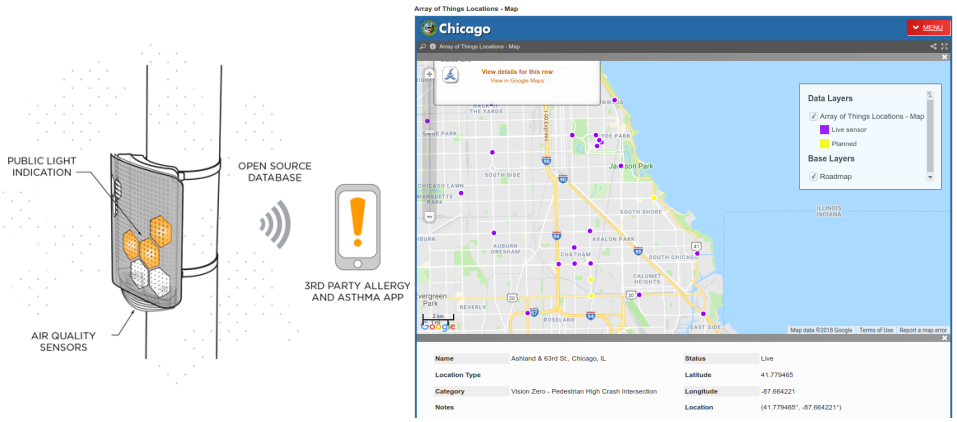


Figura 2.3: AoT: módulos [23]

O *AoT* fornece dados em tempo real via *Wifi*, tanto para o Servidor central para ser demonstrado num interface gráfico (cf. 2.4b) e para os cidadãos através do *Smartphone* com o recurso a aplicações *Android* ou *Sistema operacional Móvel da Apple (IOS)*, este sistema também prevê uma luz de *status* de acordo com a qualidade do ar demonstrado na figura 2.4a.



(a) Transmissão de dados [23] (b) Interface gráfico [23]

Figura 2.4: AoT

Esta iniciativa tem o potencial de permitir que investigadores, políticos, programadores e residentes trabalhem juntos e adotem ações específicas que tornem Chicago e outras cidades mais saudáveis, mais eficientes e mais habitáveis. Uma vez que os dados serão publicados abertamente, irá apoiar o desenvolvimento de aplicações inovadoras, por exemplo, que permitam um residente acompanhar a sua exposição à má qualidade do ar ou ruído e congestionamentos excessivos.

As caixas de sensores medirão numa fase inicial, a temperatura, a pressão barométrica, a luz, a vibração, o CO, o NO2, o SO2, o O3, a intensidade do som ambiente e fará análise de imagens através de uma câmara. A pesquisa e o desenvolvimento contínuo têm como objetivo ajudar a criar sensores para monitorizar outros fatores urbanos de interesse, como inundações, precipitação, vento e outros poluentes.

Este conjunto de sensores servirá para vários fins, entre os quais:

- Monitorizar o tráfego de veículos pesados;
- Sugerir rotas de caminhada mais saudáveis ou povoadas (devido a questões de segurança);

- Estudar a relação entre doenças (alergias ou asma) e o ambiente urbano;
- Deteção em tempo real de inundações urbanas de maneira a melhorar os serviços e a infraestrutura da cidade;
- Medições de microclima em diferentes áreas da cidade.[23]

2.3.2 Smart Citizen

O *Smart Citizen* é uma plataforma para gerar processos participativos das pessoas nas cidades ao conectar dados, pessoas, recursos, tecnologia e conhecimento. O objetivo da plataforma é servir como um nó para a construção de indicadores relevantes e ferramentas distribuídas para posteriormente, permitir a construção coletiva em termos de sensorização da cidade para os seus próprios habitantes. Este projeto é baseado em geolocalização, comunicação e *hardware/software* com o código-fonte aberto para a recolha e partilha de dados, tudo conectado através de uma plataforma online que pode ser acedida através de um *Smartphone*, *tablet* ou computador como demonstra a figura 2.5.

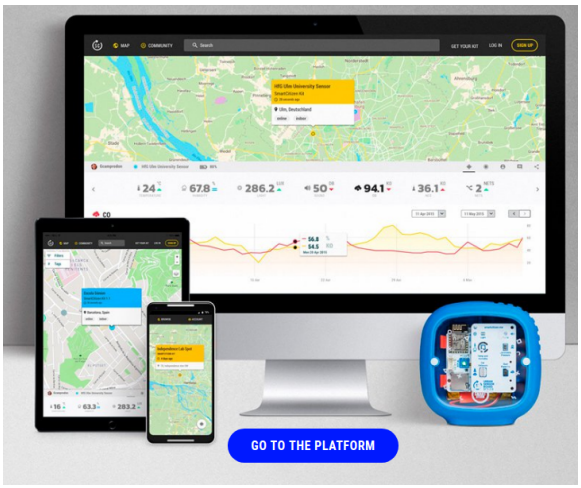


Figura 2.5: Plataforma Smart Citizen [24]

O *Smart Citizen Kit* é uma peça de *hardware* composta por duas *Printed Circuit Board (PCB)s* para a aquisição, processamento e transmissão de dados, uma bateria e uma caixa. A *PCB* principal contém um ATMega para o processamento de dados, um módulo *WiFi* para comunicação, um cartão de memória para guardar os dados em situações que não é possível a sua transmissão, um interface USB para a configuração do dispositivo, um interface para ligar a placa de sensores, um conector para a bateria e a possibilidade de acrescentar um painel solar. Por sua vez, a PCB dos sensores pode medir a composição do ar (CO e NO2), temperatura, humidade, intensidade da luz e níveis sonoros. Estas duas *PCBs* estão representadas na figura 2.6.

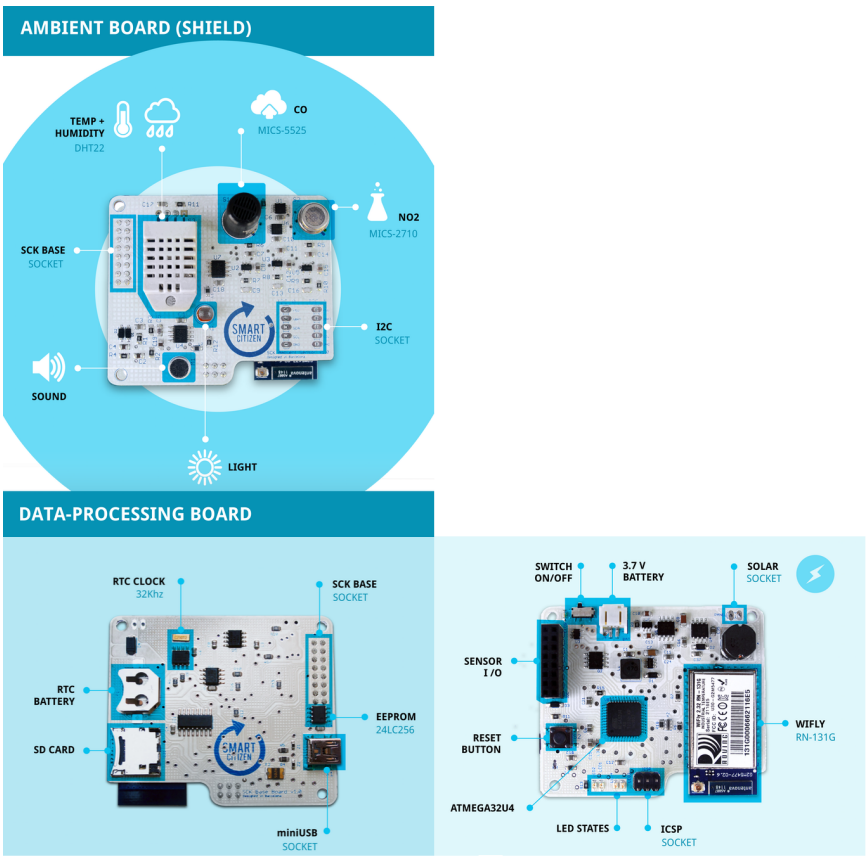


Figura 2.6: *Smart Citizen Hardware* [25]

Este dispositivo tem de ser configurado para se conectar a uma rede *WiFi*

com ligação à internet ao seu alcance para ser possível transmitir os dados recolhidos. O baixo consumo de energia do dispositivo permite colocá-lo em varandas, janelas e no exterior de edifícios.[24], [25]

Na cidade piloto deste projeto, Barcelona, demonstrou-se um caso de uso real deste tipo de sistemas. Esta plataforma foi utilizada pelos residentes de um local movimentado do centro da cidade para demonstrar que viviam com níveis de ruído inaceitáveis principalmente à noite. Para isso, os moradores instalaram nas suas varandas e janelas *kits smart citizen* e recolheram os dados de ruído durante a noite que revelaram picos de 100dB, o que não é recomendado pela OMS. Com esta informação dirigiram-se à câmara municipal pressionando-os para repensarem o uso daquele local. Atualmente, já foram tomadas medidas para melhorar a qualidade de vida destes moradores, como por exemplo, a polícia passou a ter um cuidado especial com aglomerado de pessoas naquela zona e a recolha do lixo passou a ser feita de manhã [26].

2.3.3 Telensa Planet: Iluminação inteligente

A *Telensa* pretende implementar um controlo remoto sem fios para a iluminação de ruas, estradas e áreas de lazer beneficiando da atual mudança do tipo de iluminação para *Light-Emitting Diode* (LED). Com controlo à distância, o utilizador pode economizar energia (utilizar apenas a quantidade precisa de luz) e medir com precisão cada *Watt* gasto. Este sistema utiliza para comunicação uma tecnologia sem fios de largura de banda ultra-estreita e longo alcance o que a diferencia de outras soluções no mercado, também oferece uma implementação compatível com os principais fabricantes de iluminação e possível integração com outros sistemas. Além disso, permite adicionar outro tipo de sensores aos postes de iluminação na vertente das

cidades inteligentes como por exemplo, sensores de qualidade do ar ou câmaras para análise de tráfego de veículos.

O objetivo desta plataforma é por localidade (2-3km no meio urbano e 5-8km no meio rural) ter vários postes de iluminação inteligentes que recolhem os dados pretendidos, ligados entre si através de uma tecnologia *wireless* de largura de banda ultra-estreita, a uma estação base que permite o interface com cada um destes dispositivos. Por sua vez, a estação base está conectada via *internet* ao sistema central que permite a gestão e visualização de todos os dados recolhidos pelo sistema como podemos visualizar na figura 2.7.[27]

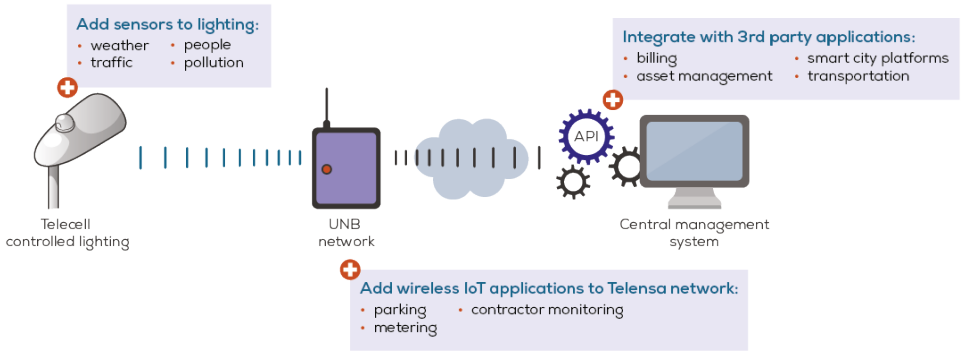


Figura 2.7: Telensa: Arquitectura do sistema [27]

2.3.4 Bigbelly: Gestão inteligente de resíduos

A *Bigbelly* propõe uma solução inteligente para o depósito de resíduos e reciclagem em espaços públicos. Implementam estações de reciclagem e lixo inteligentes, alimentadas a energia solar e equipadas com sensores que transmitem o seu estado em tempo real para as equipas de coleta de forma a permitir que as mesmas sejam eficientes.

As estações podem ser configuradas como estações únicas independentes ou como estações duplas ou triplas (*multi-stream*) para acomodar as recolhas de resíduos de cada comunidade ou instalação. Cada estação é personalizada com base na capacidade (fluxo de resíduos), podendo incorporar sensores

consoante a necessidade, como por exemplo, qualidade do ar e ruído), também tem capacidade para efetuar processamento de dados, um Sistema de Posicionamento Global (GPS) para georreferenciação, pode servir como um *hotspot WiFi*, contém um *beacon* para enviar mensagens e notificações para a vizinhança como por exemplo fazer publicidade, todas estas características estão representadas na figura 2.8).

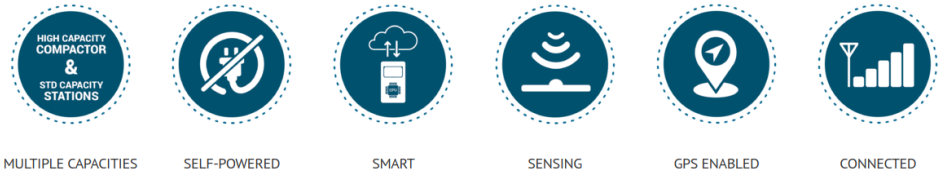


Figura 2.8: Bigbelly: Características da solução [28]

Estes contentores comunicam todos os dados recolhidos via 3G ou 4G (rede móvel) para um Servidor central que pode ser acedido através de um *smartphone* ou computador como demonstra a figura 2.9.[28]

Figura 2.9: Bigbelly: Arquitectura do sistema [28]

2.4 Sistemas de Sensorização Móveis para Cidades Inteligentes

Além dos sistemas fixos apresentados na secção 2.3 existem abordagens móveis para a sensorização e monitorização das cidades o que obriga a outro tipo de arquitetura de sistema, tendo em conta restrições de consumo e tamanho.

2.4.1 Projeto Ekobus - Libelium

O sistema *EkoBus* está enquadrado no projeto *SmartSantander* [29], foi desenvolvido em colaboração com a *Ericsson* e implementado nas cidades de Belgrado e Pancevo na Sérvia. O sistema utiliza os veículos de transporte público para monitorizar um conjunto de parâmetros ambientais numa grande área, bem como fornecer informações adicionais para o utilizador final, como a localização dos autocarros e os horários de chegada estimados.

Cada autocarro integrante no projeto tem um *kit* desenvolvido pela Libelium designado de *Waspmotes*. Este *kit* é completamente modular, pois podem ser adicionados ou removidos sensores (e.g. CO, CO2, NO2, COV), interfaces *wireless* (e.g ZigBee, SigFox, LoRa, WiFi), protocolos de comunicação (e.g RS-232, RS-485, CAN) e permite programação *over-the-air*. Dependendo do tipo de *hardware* e *software* adicionado o preço pode atingir mais de mil euros [30].

Para este projeto em foram desenvolvidos 65 *kits* distribuídos em duas localizações diferentes com a possibilidade de medição de cinco parâmetros:

- Temperatura
- Humidade
- CO

- CO2
- NO2

Para saber onde cada *kit* está localizado e consequentemente cada autocarro, os *Waspmotes* podem integrar um *GPS*, que fornece informações precisas de posição e tempo. Normalmente estes *kits* transmitem os dados recolhidos via *Zigbee* e têm um módulo de rádio secundário para transmitir os dados via *General Packet Radio Service* (GPRS), garantindo assim uma maior disponibilidade de rede e redundância em situações em que é fundamental garantir a recepção da mensagem. O módulo GPRS pode operar em 4 bandas diferentes, por isso suporta qualquer fornecedor de rede móvel utilizado pelos telemóveis, tornando-o capaz de funcionar em todo o mundo. Neste projeto os dados são enviados via GPRS, pois os autocarros estão em constante movimento por toda a cidade, na figura 2.10 está representada a arquitetura acima descrita que contém os 65 *kits* distribuídos pelas duas cidades e que comunicam os dados recolhidos para o Servidor central via GPRS.

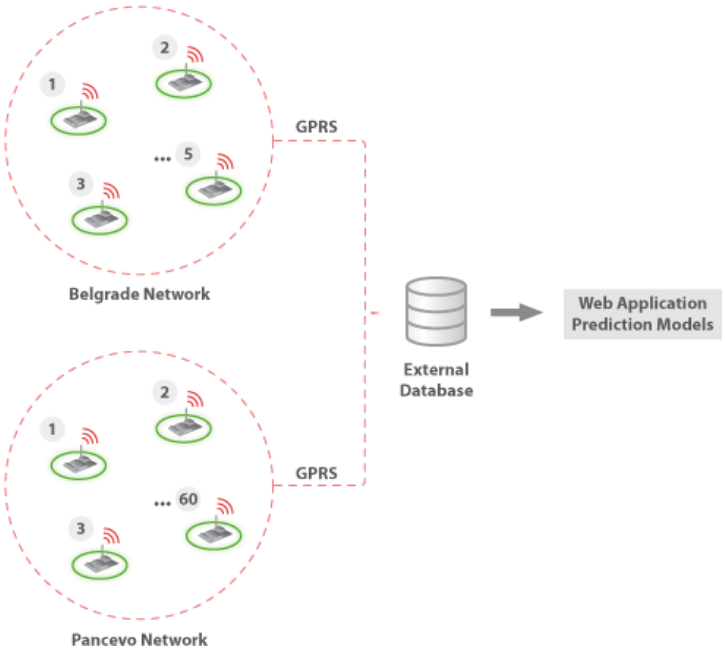
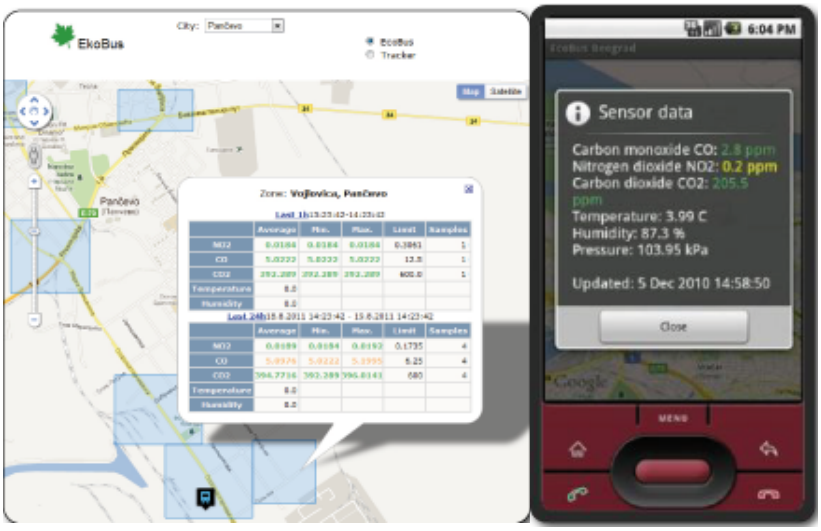


Figura 2.10: Projecto *Ekobus*: Arquitetura do sistema [31]

A nível energético, o *Waspmote* tem um modo *low-power*, no qual periodicamente o sistema acorda, recolhe os dados dos diferentes sensores, envia os mesmos para a estação a central e volta a dormir. Cada dispositivo pode ser alimentado através de baterias recarregáveis ou um painel solar.

Normalmente, a calibração dos sensores de qualidade do ar é realizada em laboratórios certificados e é um processo complexo e caro. Como ponto de partida a calibração de um pequeno conjunto de sensores é feita em laboratório, e todos os outros restantes sensores são calibrados tendo como base esses sensores de referência.

Recorrendo à comunicação de dados via GPRS, todas as leituras são entregues ao Servidor onde as mesmas são processadas e armazenadas. Por sua vez, os utilizadores através de uma aplicação *Android* ou interface *Web*, podem aceder a todos esses dados como está ilustrado nas figuras 2.11a e 2.11b. Como se pode ver na figura 2.12, os sensores foram colocados dentro de uma caixa possibilitando a sua instalação nos tejadilhos dos autocarros. [31]



(a) Web (b) Android

Figura 2.11: Projeto Ekobus: Interface Gráfico [31]

Figura 2.12: Projecto Ekobus: Instalação dos dispositivos [31]

2.4.2 *EveryAware*: Monitorizar a qualidade do Ar

A Fundação *Institute for Scientific Interchange (ISI)* [32] é uma instituição de pesquisa privada localizada em Turim, Itália. Um dos casos de estudo desta instituição é a sensorização da qualidade do ar baseando-se em três componentes principais:

- Caixa de sensores;
- Aplicação *Android* para *smartphones*;
- Aplicação *web*.

A caixa de sensores é um dispositivo portátil que mede as concentrações de poluentes no ar e é localizada através de um GPS. Este processo é feito uma vez por segundo, a fim de ter um mapa detalhado da poluição enquanto a caixa é transportada. Os dados recolhidos dos sensores são lidos por um microcontrolador e armazenados num cartão de memória. Esses dados são guardados em conjunto com a sua posição. Um módulo Bluetooth permite a conexão da caixa de sensores com um *smartphone* de modo a transferir todos os dados recolhidos para serem visualizados em tempo real através de uma aplicação *Android* ou encaminhar para o Servidor principal via 3G (rede móvel) para uma análise global de toda a comunidade através de um

interface web como representa a figura 2.13.

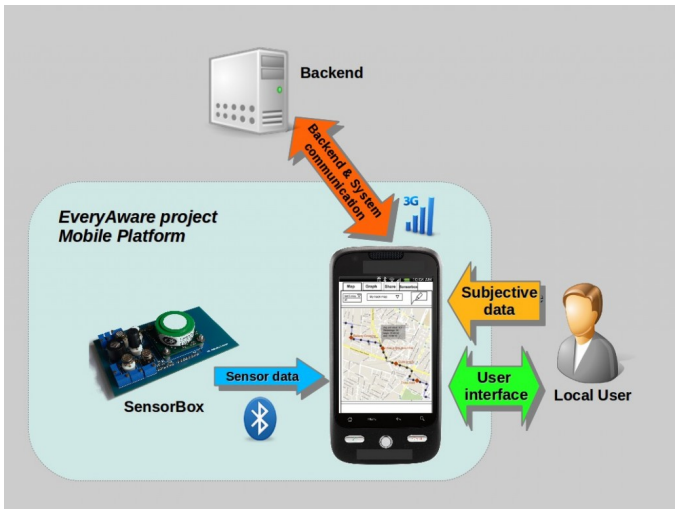


Figura 2.13: Everyaware: Arquitectura do sistema[33]

A caixa de sensores é uma pequena PCB que utiliza diferentes sensores de qualidade do ar de baixo custo para avaliar a poluição atmosférica. É pequena o suficiente para ser carregada em uma mochila e é alimentada com uma bateria externa. A caixa de sensores tem um pequeno ventilador que empurra o ar externo para uma câmara onde existem os diferentes sensores que medem os poluentes do ar como é possível visualizar no exemplar representado na figura 2.14.

Figura 2.14: Everyaware: Caixa de sensores [33]

Este tipo de sensores normalmente funcionam bem em ambientes altamente controlados, como um laboratório, enquanto em condições reais o desempenho e a sua precisão são afetados por uma série de fatores como o vento, condições climatéricas ou gases interferentes. Um dos objetivos é verificar se o acoplamento de diferentes sensores de baixo custo ajuda na redução de distúrbios e fornece uma medição suficientemente precisa. Esta caixa de sensores foi calibrada com a ajuda de instrumentos certificados que detetam partículas ultra finas, tendo o objetivo de fornecer medidas confiáveis. Os tipos de sensores utilizados são:

- Três tipos diferentes de sensores de CO;
- NO2;
- Sensores que reagem à presença de gasolina ou gasóleo;
- O3;
- COV;
- Humidade;
- Temperatura.

Na figura 2.15 podemos ver um exemplo dos dados recolhidos em uma viagem através da aplicação web.[33]

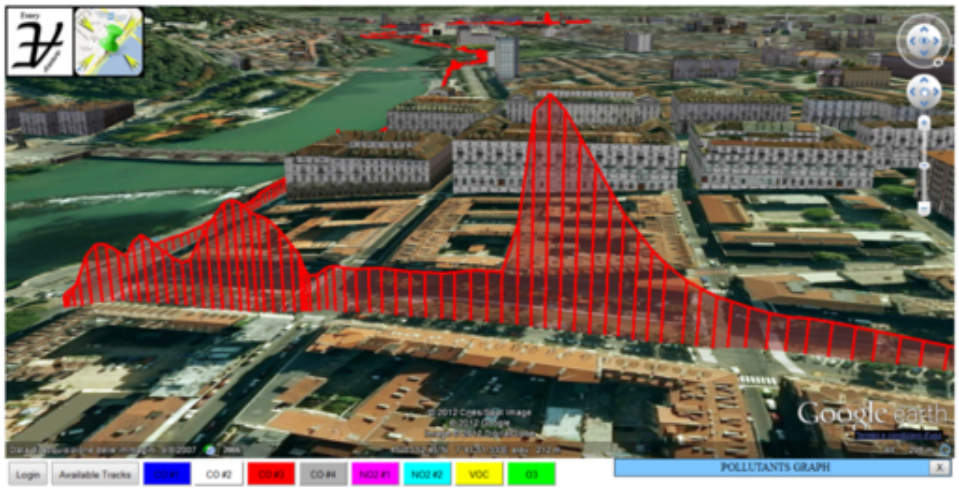


Figura 2.15: Análise de dados na aplicação web da EveryAware [33]

Capítulo 3

Tecnologias de Comunicação

Foi desenvolvido um estudo sobre algumas tecnologias de comunicação que podem ser úteis durante o desenvolvimento deste projeto. Neste capítulo são apresentadas as tecnologias mais relevantes com destaque para o *Bluetooth Low Energy (BLE)*.

3.1 *Internet of Things*

Um dos pontos mais importantes numa cidade inteligente são as tecnologias da comunicação que interligam todo o sistema. Convém primeiramente, antes de falar de cada uma das tecnologias, introduzir o conceito de *IoT*. O *IoT* é uma rede de objetos físicos, sensores, atuadores, eletrodomésticos, entre outros, que possuem capacidade embebida para estabelecer conexão com uma rede, ou seja, ser capaz de coletar e transmitir dados de maneira a criar uma rede inteligente que seja gerida do modo mais autónomo possível.
O *IoT* emergiu dos avanços de várias áreas como a microeletrónica, protocolos de comunicação e a evolução dos sensores. Este tipo de rede tem recebido bastante atenção tanto a nível académico quanto da indústria, devido ao seu

potencial de uso nas mais diversas áreas das atividades humanas.
O *IoT* é composto por três componentes principais:

- Os dispositivos;
- A rede que os conecta;
- Os servidores para o tratamento de dados.

Como o desenvolvimento desta tese vai incidir sobre a criação e integração de uma Caixa de Sensores numa vertente de cidades inteligentes é importante introduzir algumas tecnologias de comunicação [34], [35].

3.2 ZigBee

Em 2002 a necessidade de uma tecnologia *Home Area Network (HAN)* de baixo custo, baixo consumo, curto alcance e tamanho reduzido do dispositivo levou um grupo de fabricantes e comercializadores a implementar, em conjunto, a norma *Institute of Electric and Electronic Engineers (IEEE)* 802.15.4 [36]. Desta forma a rede tem como principais casos de uso dispositivos que não necessitem de taxas de transmissão de dados elevadas, e que pretendem aproveitar características de baixo consumo. As principais características desta tecnologia são:

- Diferentes frequências de operação e taxas de transmissão de dados: 868 MHz a 20 Kbps, 915 MHz a 40 Kbps, 2.4 GHz a 250 Kbps;
- Cada nó pode ter diferentes funcionalidades na rede (coordenador, *router* e dispositivo final);
- São possíveis diversas topologias de rede (estrela, árvore e *mesh*);
- Suporta comunicações *multihop* (múltiplos saltos entre nós);

- Capacidade de se auto-organizar e auto-reestruturar (entrada e saída de nós na rede);
- Permite um número elevado de dispositivos conectados à rede;
- Alta durabilidade da bateria dos dispositivos.

O coordenador ZigBee tem a responsabilidade da criação e manutenção da rede. Normalmente não tem restrições de alimentação, de forma a reduzir o risco de falha no nó centralizador da rede.
Os nós designados de *routers* possuem tabelas com rotas e permitem encontrar o caminho mais curto para se chegar ao destino pretendido. Caso não possuam o endereço de destino requisitado, este fará o *broadcast* de uma requisição de rota e receberá do destino a rota mais eficaz atualizando a sua tabela. Este mecanismo dá à rede a característica de auto-regeneração caso ocorra uma falha de outros nós roteadores na rede.
Por fim, os dispositivos finais não têm a função de roteamento nem coordenam a rede. Eles comunicam diretamente com um nó roteador e podem ser implementados com maiores restrições em termos de memória e potência, passando quase todo o tempo num estado inativo.
Na figura 3.1 demonstra-se uma das múltiplas possíveis configurações de uma rede ZigBee em que a verde está representado o nó central, a azul os nós roteadores e a cor de laranja os dispositivos finais, também podemos verificar 2 tipos de conexão em estrela e *mesh* (em que todos os nós ao alcance estão ligados entre si).

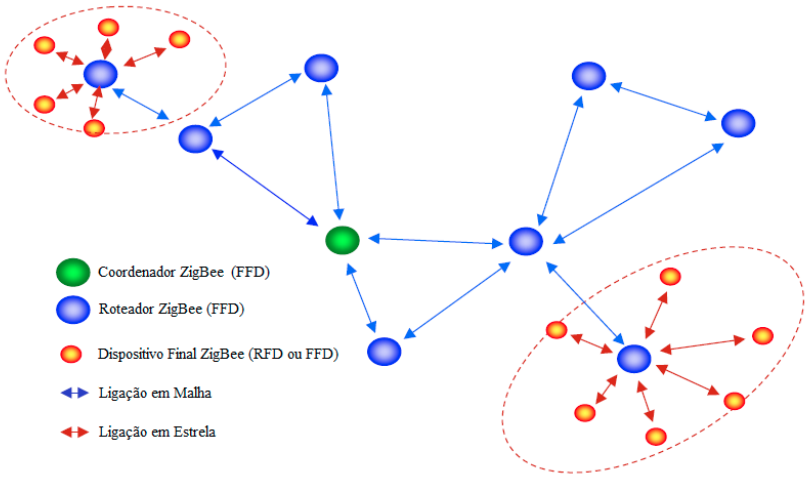


Figura 3.1: Modelo de rede ZigBee [37]

O número de saltos permitido entre nós é limitado pelo tipo de módulo coordenador que normalmente varia entre algumas unidades a dezenas. Por sua vez, o alcance entre 2 nós também é dependente do tipo de módulo e do meio (espaço aberto ou com obstáculos), normalmente varia entre algumas dezenas até centenas de metros.

A maioria dos módulos de séries diferentes ou com *firmwares* diferentes (podem ser atualizados) não são compatíveis.

3.2.1 Arquitetura

O padrão 802.15.4 ficou responsável pela criação das duas camadas mais baixas da tecnologia ZigBee, enquanto que a ZigBee Alliance trabalhava nas camadas superiores. Na figura 3.2 podemos ver como a *stack* está montada.

Figura 3.2: Stack de Zigbee [37]

3.2.1.1 Camada Física

A camada física do ZigBee segue o protocolo 802.15.4 e é responsável por permitir a transmissão dos pacotes de dados (*Protocol Data Unit (PDU)s*), através de ondas de rádio. Também é responsável por indicar a qualidade e potência de conexão e reportar os canais disponíveis para comunicar.

3.2.1.2 *Media Access Control*

É responsável pelo processo do encapsulamento dos dados provenientes das camadas superiores preparando-os para serem transmitidos. O método de acesso ao meio carateriza a rede em dois modos de operação. O modo *beaconing*, que consiste em fazer com que os *routers* transmitam periodicamente *beacon frames*, que são pacotes sinalizadores para confirmar a presença dos dispositivos na rede. Os dispositivos (exceto o coordenador) só precisam acordar quando for o momento de receber um *beacon frame*, interpretar alguma configuração diferente que tenha sido passada, e voltar a dormir até à sua próxima atividade. Neste caso é utilizado como método de acesso ao meio o *Carrier sense multiple access with collision avoidance* (CSMA-CA), o que permite evitar as colisões, de forma a que os dispositivos esperem por um canal livre para transmitir.

No outro modo designado de *non-beaconing* os dispositivos estão sempre li-

gados o que acarreta uma menor poupança de energia bem como uma maior transmissão de dados. Este segundo modo também utiliza um método de controlo de acesso ao meio idêntico ao anterior, com a diferença de que, se o canal em questão estiver ocupado o tempo de espera para retransmitir os dados é aleatório.

3.2.1.3 Camada de Rede / Segurança

Em termos de segurança, esta camada protege os pacotes da camada de *Media Access Control (MAC)* que são transmitidos num único salto na rede. Para saltos múltiplos, a segurança é feita nas camadas superiores (rede e aplicação). É utilizado o algoritmo *Advanced Encryption Standard (AES)* para encriptar e validar os dados que são enviados. A validação ou garantia de integridade dos dados é feita por um *Message Integrity Code (MIC)*. Esta camada também determina o tipo de proteção a utilizar, tem um contador de *frames* que garante a sequência e autenticação dos dados e guarda uma *key* que identifica cada um dos nós para servir de referência.

A camada de rede é responsável pela configuração dos dispositivos, implementação de mecanismos para a descoberta de rotas e encaminhamento da informação. Esta camada tem como funções:

- Encontrar novos dispositivos que possam integrar a rede;
- Monitorização dos dispositivos da rede;
- Armazenar informação relativa aos dispositivos;
- Atribuir endereços (função do coordenador).

3.2.1.4 Camada de Aplicação

A camada de aplicação, como o nome indica é responsável pela comunicação da aplicação com o dispositivo. A separação em camadas permite que a última camada (considerando a camada física como a primeira) utilize todos os recursos analisados anteriormente de forma transparente, e possibilita que outras características sejam adicionadas à estrutura rede, como a segurança que pode ser implementada com a utilização de criptografia e autenticação. As responsabilidades da camada de aplicação incluem a manutenção de tabelas para efetuar o emparelhamento, que é a capacidade de combinar dois dispositivos baseados nos seus serviços e necessidades, e o encaminhamento de mensagens entre os dispositivos ativos. [37], [38]

3.3 Redes *Low Power Wide Area Network*

Após uma breve análise efetuada a uma rede pensada para áreas pequenas, convém ter uma ideia sobre a existência e o funcionamento de redes para áreas de maior dimensão, como por exemplo, uma cidade. Os novos padrões de tecnologia desenvolveram-se para lidar com a crescente demanda do *IoT*. Enquanto os telemóveis utilizam redes próprias para a transmissão de dados (*Global System for Mobile Communications (GSM)/3G*), muitos dispositivos *IoT*, por exemplo, um medidor de caudal de água inteligente só precisa de transferir pequenas quantidades de dados numa área geográfica de dimensões consideráveis. Utilizar a rede *GSM* tal como conhecemos ou via satélite para transmitir os dados destes dispositivos seria caro e energeticamente demasiado exigente, assim surgiu a necessidade da criação de redes projetadas para incorporar dispositivos de baixo consumo que pretendam transmitir pequenas quantidades de dados a médias/longas distâncias.

Este tipo de redes funcionam do seguinte modo, na camada física existem

dispositivos que pretendem comunicar entre si ou com um Servidor central, através de comunicações sem fios de baixo consumo e médio alcance. Como os dispositivos que pretendem comunicar podem não estar ao alcance uns dos outros, esses dados são enviados para *gateways* (espalhados pela região de interesse), que não têm restrições energéticas. Por sua vez, os *gateways* reencaminham a informação para um Servidor central recorrendo a outro tipo de redes, normalmente, 3G/4G ou *ethernet*. Na figura 3.3 demonstra-se uma arquitetura genérica deste tipo de redes.

Algumas tecnologias que representam este tipo de rede são o *Long Range* (LoRa), *Narrowband IoT* (NB-IoT) e SigFox representados nas secções 3.3.1, 3.3.2 e 3.3.3 respetivamente. Em cada uma destas tecnologias a camada de transporte adapta-se consoante o meio de comunicação utilizado, ou seja, podem ser utilizadas diferentes tecnologias para comunicação com os *gateways*. [39], [40], [41].

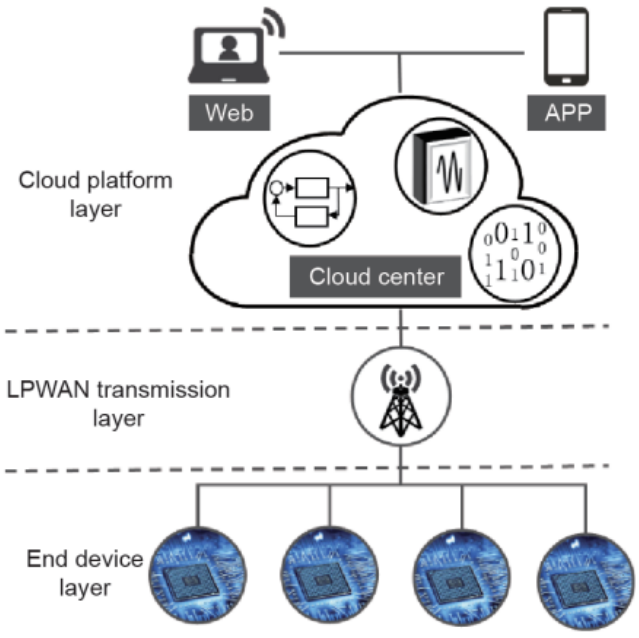


Figura 3.3: Redes LPWAN: Arquitetura genérica [39]

3.3.1 *Long Range*

O termo LoRa e *Long Range Wide Area Network* (LoRaWAN) são frequentemente utilizados com o mesmo significado, mas, por definição, há uma diferença. LoRa define o *standard* para a camada física, o LoRaWAN define além disso, a camada de controlo de acesso ao meio e o interface com as aplicações.O LoRaWAN é um *standard* de comunicação para redes sem fios, este tipo de rede situa-se entre o Bluetooth/Zigbee e o GSM/3G.

As principais características do LoRaWAN são:

- Longo alcance ($>$ 5 quilómetros)
- Módulos com longa duração da bateria ($>$ 10 anos - dependente do nível de utilização)
- Baixa velocidade de transmissão de dados (até 50 Kbps)
- Opera num espetro não licenciado

A comunicação sem fios, fisicamente limitada, é sempre um *trade off* entre distância, velocidade e energia consumida. O LoRa como todas as redes *Low Power Wide Area Network* (LPWAN) foi projetado para casos de uso onde o alcance e o consumo são pontos bastante mais críticos do que a taxa de transmissão dos dados. O LoRa é utilizado em aplicações de estacionamento inteligente, sistemas de controlo de luminosidade, monitorização do ambiente, *tracking* de veículos entre outros.

No LoRa a comunicação é bidireccional, para enviarmos uma mensagem do sensor até à aplicação final a mesma é encriptada e transmitida via rádio pelo módulo LoRa, um ou mais *gateways* recebem a mensagem e encaminham-na

através de outra rede (normalmente 3G ou *ethernet*) para o Servidor central, se necessário o Servidor também replica a mensagem para a aplicação final. Quando a comunicação é feita na direção oposta, ou seja, pretendemos atuar no dispositivo ou enviar alguma configuração, como o LoRa foi concebido para o menor consumo de energia possível, nem todos os dispositivos estão sempre à escuta de mensagens, depende das classes do dispositivo que podem ser:

- A - Só pode receber mensagens no momento em que o dispositivo está a enviar uma mensagem
- B - Pode receber dados no momento do envio e acorda periodicamente para receber os mesmos
- C - Permanentemente à escuta para receber mensagens

Logicamente, os dispositivos de classe A são os energeticamente mais sustentáveis enquanto que os dispositivos de classe C normalmente não são alimentados por baterias.

O padrão LoRaWAN descreve dois tipos diferentes de mensagens, as de controlo de acesso ao meio para controlar as transmissões e a rede, e as mensagens de dados, em que o *payload* é específico do dispositivo. Uma vez que há limitações temporais e no número de mensagens que podemos enviar, as mensagens de controlo podem ser encaminhadas juntamente com mensagens de dados, bem como várias mensagens podem ser enviadas de uma só vez.

Tal como acontece com a maioria dos protocolos de rede, os dispositivos precisam de algum tipo de endereço e identificação para poderem ser únicos e interagirem com a rede. O LoRa usa o seguinte endereçamento:

- Identificador (ID) de hardware exclusivo do dispositivo - endereço de 64 bits. Comparável com um endereço *MAC* para um dispositivo *Transmission Control Protocol* (TCP)/*Internet Protocol* (IP);
- Endereço do dispositivo na rede - endereço de 32 bits atribuído ou escolhido especificamente pela rede. Comparável com um endereço IP para um dispositivo TCP/IP;
- ID do serviço/aplicação final;
- ID da porta do serviço/aplicação final. Comparável com um número de porta para um dispositivo TCP/IP.

Como o LoRa utiliza sinais de rádio frequência para comunicar e opera num espectro não licenciado, significa que qualquer utilizador pode usar livremente esta banda sem pagar ou obter uma licença, seguindo alguns regulamentos. A banda de funcionamento do LoRa é conhecida por *Industrial, Scientific and Medical (ISM) band*. Esta banda varia de frequência de funcionamento entre a Europa, Estados Unidos e Austrália, ou seja, um *gateway* fabricado nos Estados Unidos não pode ser utilizado na Europa.

Esta tecnologia prevê a utilização de algumas técnicas para melhorar a robustez dos dados transmitidos:

- Dispersar os dados pelo espectro: é uma técnica originalmente projetada para aplicações militares onde a informação (por exemplo, um bit) é dispersa por uma larga escala de frequências. Ao fazer isso, a relação sinal-ruído é maior e o sinal (o bit enviado) é menos resistente ao ruído e interferências.
- Adaptação da taxa de dados: altera dinamicamente a taxa de dados e a potência de transmissão em função da qualidade do sinal e da distância

para o *gateway*. A transmissão mais lenta permite uma distância maior e comunicação mais fiável.

- Correção de erros: adiciona informações redundantes (por exemplo, bits de paridade) a cada mensagem para corrigir eventuais erros de sincronização.

Como a LoRa é uma tecnologia *half-duplex*, é importante evitar colisões nos dois sentidos. Um *gateway* LoRa deve responder a uma solicitação de um nó para *acknowledge* ou *downlink* dentro de um período de tempo fixo. Qualquer mensagem enviada durante este período não será recebida pelo *gateway*.

Em relação aos *gateways* utilizados no LoRa também conhecidos por pontos de acesso, recebem todas as comunicações LoRa enviadas por dispositivos ao seu alcance. Não existe obrigatoriamente uma associação entre um dispositivo e um *gateway* LoRa específico. Cada *gateway* na gama de frequências do sinal enviado por um dispositivo irá processar essa mensagem, mesmo que este *gateway* faça parte de uma rede que não conhece o dispositivo que enviou a mensagem. [42], [43], [44]

3.3.2 *Narrowband IoT*

O *3rd Generation Partnership Project* (3GPP) formou um grupo de trabalho para fundir a tecnologia *IoT* da Huawei em parceria com a Vodafone *(Narrowband Cellular IoT (NB-CIoT)*) e a tecnologia *Narrow-Band Long-Term Evolution (NB-LTE)* apresentada pela Nokia, Ericsson e Intel do qual nasceu o NB-IoT. O NB-IoT surge como o LoRa no âmbito das redes LPWAN devido à necessidade do uso de dispositivos que enviam pequenas quantidades de dados, promovem a longevidade da bateria e tendem

a ser colocados em locais remotos. É uma tecnologia que pretende criar uma alternativa sustentável para a conexão de milhões de dispositivos.

O *standard* do NB-IoT começou em 2014 com um estudo global das comunicações sem fio para tecnologias via rádio para perceber o desenvolvimento atual e o que é necessário para cumprir com os objetivos desta tecnologia, tais como:

- Excelente cobertura de rede mesmo dentro de edifícios e no subsolo;
- Facilmente integrável na rede de comunicação dos telemóveis;
- Tecnologia otimizada para baixo consumo e longa duração da bateria;
- Rede segura e fiável.

O NB-IoT utiliza como *standard* para as camadas de alto nível o *Long Term Evolution* (LTE), que é um *standard* de comunicações via rádio 4G. O NB-IoT pretende desenvolver o próprio controlo de acesso ao meio e dos canais utilizados para comunicar. Resumindo a especificação do NB-IoT permite que muitos dispositivos enviem pequenas quantidades de dados em paralelo. Esta tecnologia pode ser implementada, em termos de largura de banda, em três zonas representadas na figura 3.4):

- Independentemente, na própria largura de banda;
- Em larguras de banda de 200 Khz que já não são utilizadas pelo GSM;
- Na banda de guarda ou em blocos de operação do LTE.

Figura 3.4: NB-IoT: Exemplos da largura de banda [45]

O NB-IoT pretende aproveitar a infraestrutura e cobertura já existente para as redes de telemóvel, como nas outras tecnologias, esta também prevê medidas para reduzir o consumo da mesma podendo os dispositivos receber informação de maneira descontinua, sendo capaz de entrar em modo de poupança de energia. Embora sempre que um dispositivo desligue o módulo de rádio para economizar energia, esse mesmo dispositivo normalmente teria que se conectar novamente à rede quando o módulo de rádio fosse ligado para comunicar. Este procedimento de reconexão à rede consome uma pequena quantidade de energia, mas o seu consumo acumulativo pode se tornar significativo ao longo da vida útil de um dispositivo. Esta tecnologia prevê o uso de uma técnica designada de *Power Saving Mode* (PSM), que consiste num tempo negociado entre o dispositivo e rede para o qual a rede mantém o dispositivo configurado e registado na mesma, ou seja, se um dispositivo acordar e enviar dados antes do término do intervalo de tempo que concordou com a rede, um procedimento de re-conexão não é necessário. O NB-IoT usa também uma técnica designada de *Adjustable Extended Discontinuous Reception (eDRX)*, que em vez de desligar o módulo de comunicação, desliga e liga periodicamente a sua capacidade de receção. Para um dispositivo IoT, pode ser bastante aceitável que o mesmo não esteja sempre acessível, como por exemplo, um telemóvel. [45] [46]

3.3.3 SigFox

O SigFox surge igualmente no âmbito das duas tecnologias apresentadas anteriormente, com o intuito de alavancar o avanço do IoT.

O SigFox utiliza uma arquitetura muito parecida com o LoRa no sentido em que os dispositivos comunicam tramas SigFox para a estação base (*gateway*), na estação base essas tramas são processadas e enviadas para um Servidor central, normalmente via *ethernet*.

O sistema de suporte Sigfox constitui a rede central encarregada de processar as mensagens e enviá-las através de *callbacks* para o sistema de Clientes. Esta camada fornece também o ponto de entrada para os diferentes utilizadores do sistema (operadores, *gateways* e Clientes finais) para interagir com o mesmo através de interfaces web ou *Application Programming Interface* (API)s. Esta camada também inclui módulos e recursos que são essenciais para garantir o desenvolvimento, a operação e o monitoramento da rede. Além disso, esta camada inclui o repositório e ferramentas para analisar os dados recolhidos ou gerados pela rede.

O acesso aleatório à rede é uma característica fundamental para alcançar uma alta qualidade de serviço, assim sendo, a transmissão não está sincronizada entre a rede e o dispositivo. O dispositivo emite uma mensagem numa frequência aleatória e, em seguida, envia 2 réplicas em frequências e tempo diferentes, fazendo com que existam potencialmente, muitas repetições da mesma mensagem que chega à rede central, mas apenas uma delas deve ser armazenada.

A infraestrutura da rede está preparada para armazenar os dados em dois locais diferentes separando as mensagens dos Clientes das mensagens relativas à gestão da rede.

Esta tecnologia utiliza uma modulação ultra-estreita em que cada mensagem necessita de 100 Hz de largura de banda e é transferida com uma taxa de dados de 100 a 600 bits por segundo, dependendo da região. Na figura 3.5 temos a demonstração do espectro da largura de banda desta tecnologia.

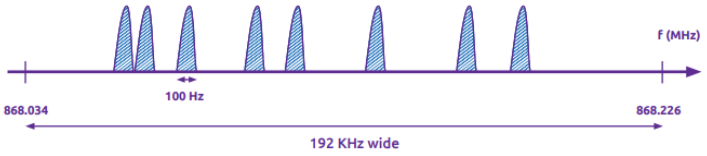


Figura 3.5: Sigfox: largura de banda ultra-estreita [47]

Na Europa a largura de banda utilizada é entre os 868 e 868.2 Mhz, enquanto no resto do mundo é utilizada a banda entre os 902 e 928 Mhz de acordo com as restrições locais.

A capacidade da infraestrutura da rede *Sigfox* é o resultado dos fatores descritos anteriormente: a modulação de banda ultra-estreita tem o benefício de ser eficiente no espectro e resistente a interferências, já que toda a energia é concentrada numa largura de banda muito pequena, bem como a diversidade introduzida pelo acesso ao meio de forma aleatória.

Não é necessário emparelhamento entre cada dispositivo e a rede, o que significa que não são trocadas mensagens de sincronização entre o dispositivo e a estação base antes de transmitir os dados.

Em termos de sistema o Sigfox integra algumas normas de segurança por defeito:

- Autenticação e anti-repetição em mensagens propagadas na rede;
- Encriptação com base em AES para a transmissão de dados;
- Isolamento de cada parte da rede de modo que, no caso de um *hack*, apenas um segmento da rede seja afetado.

3.4 *Bluetooth Low Energy*

Atualmente o Bluetooth lidera as redes *low power wireless* para *stream* de áudio, transferência de informação e *broadcast* de dados. Um estudo da *ABI research* [48], estima que em 2021 existirão 48 biliões de dispositivos com acesso à Internet dos quais um terço terão módulos Bluetooth incorporados. Outro ponto importante é o facto do Bluetooth estar consolidado no mercado sendo um produto maduro, pelo que o risco de deixar de existir suporte a esta tecnologia não se aplica.

Hoje em dia, muitos dos sistemas de comunicação baseados em *wireless* utilizam um nicho de tecnologias que são, maioritariamente, incompatíveis com um computador, telemóvel e outros dispositivos utilizados por clientes e empresas. A ideia de utilizar Bluetooth para comunicações com múltiplos dispositivos tem vindo a crescer, uma vez que o mesmo é robusto no sentido em que num ambiente industrial onde se tem múltiplas ondas, como por exemplo, *WiFi*, rádio, entre outras do espectro eletromagnético, podemos estar confortavelmente no computador a ouvir música, trabalhar com o rato e transferir dados via Bluetooth sem qualquer constrangimento.

O BLE como o nome indica é projetado para um consumo de energia muito baixo e está otimizado para soluções de transferência de dados. Para permitir uma operação confiável na faixa de frequência de 2,4 GHz, usa uma abordagem robusta de frequência adaptativa que transmite dados em 40 canais. Esta tecnologia oferece aos desenvolvedores alguma flexibilidade, incluindo múltiplas topologias de rede e possibilidades de transmissão de dados que suportam velocidades de 125 Kbps até 2 Mbps, bem como vários níveis de energia, de 1 mW a 100 mW.

3.4.1 Arquitectura

O BLE está organizado em 3 principais blocos de construção como demostrado na figura 3.6, esses blocos são:

- Aplicação
- *Host*
- Controlador

O bloco de aplicação é, como o nome indica, a aplicação com que o utilizador interage com a *stack* do protocolo Bluetooth. O *Host* cobre as camadas superiores da *stack*. O controlador, por sua vez, cobre as camadas inferiores. O controlador é quem permite que o *host* "durma"por longos períodos e só "acorde"quando for necessário realizar alguma ação. O *Host* pode comunicar com o módulo *BLE*, para isso utiliza um interface designado de *Host Controller Interface* (HCI). O objetivo do HCI é, obviamente, fazer o interface entre o *Host* e o Controlador, este interface pode ser feito por diferentes protocolos de comunicação (e.g. UART, SPI) permitindo a interoperabilidade entre *Host* e Controladores produzidos por diferentes fabricantes. Também existe a possibilidade destas camadas serem integradas num único circuito integrado.

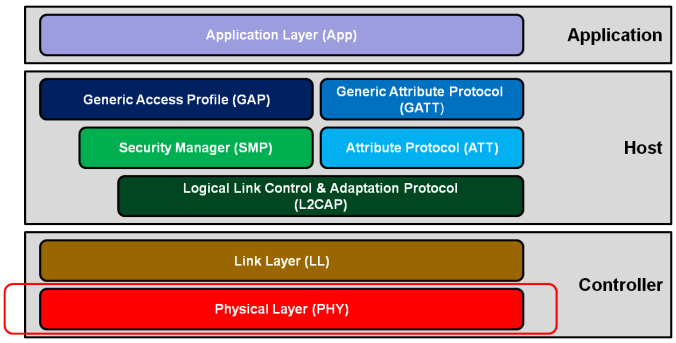


Figura 3.6: BLE: Arquitetura [49]

3.4.1.1 Camada Física

O BLE opera na banda de frequências de 2,4 GHz e define 40 canais de rádio frequência para comunicar. Existem dois tipos de canais Rádio Frequência (RF) no BLE: canais de *advertisement* e canais de dados. Os canais de *advertisement* são 3 e são utilizados para encontrar dispositivos e para transmissões *broadcast* (replica os dados para todos os dispositivos ao seu alcance sem estar conectado a nenhum deles), enquanto os 37 canais restantes de dados são usados para uma comunicação bidireccional entre os dispositivos conectados.

3.4.1.2 Camada Ligação

Nesta camada é implementado um esquema de frequência adaptativa para garantir uma operação robusta. Este mecanismo tem como objetivo remapear um determinado pacote proveniente de um canal com interferências para um canal sem interferências, de forma a que o ruído de outros dispositivos (e.g. *WiFi*) seja reduzido.
Por exemplo, um dispositivo BLE que esteja na mesma área que várias redes *WiFi* que transmitem nos canais de WiFi 1, 6 e 11, marcaria os canais Bluetooth de 0-8, 11-20, 24-32 como canais com interferências. Isso significa que, quando os dois dispositivos estiverem a comunicar, eles percorrerão os canais e serão remapeados num conjunto de canais sem interferências, neste caso os canais 9-10, 21-23 e 33-36 como é exemplificado na figura 3.7.

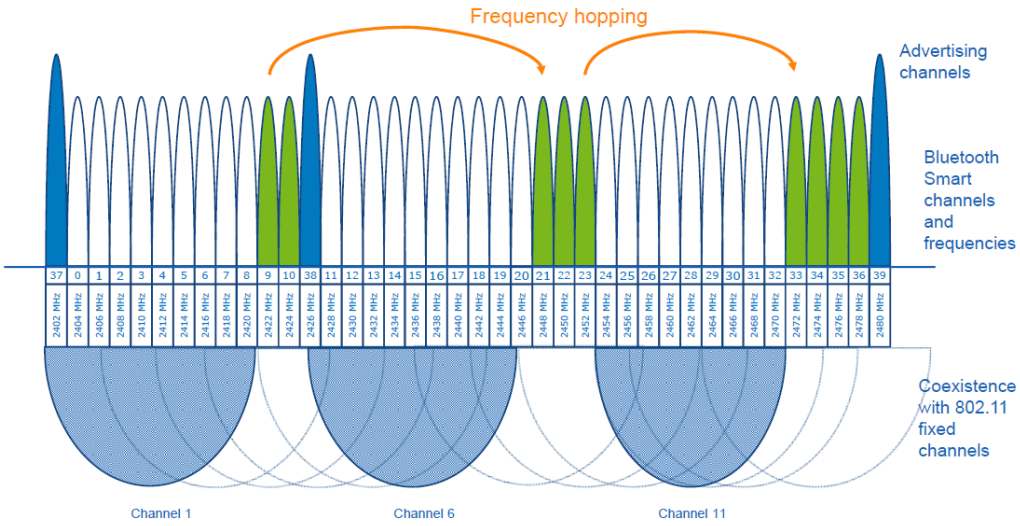


Figura 3.7: Frequência adaptativa [49]

A camada de ligação pode ser descrita em termos de uma máquina de estados que opera em 5 estados distintos, sendo eles: *standby*, *advertising*, *scanning*, *initiating* e *connected* como demontra a figura 3.8.

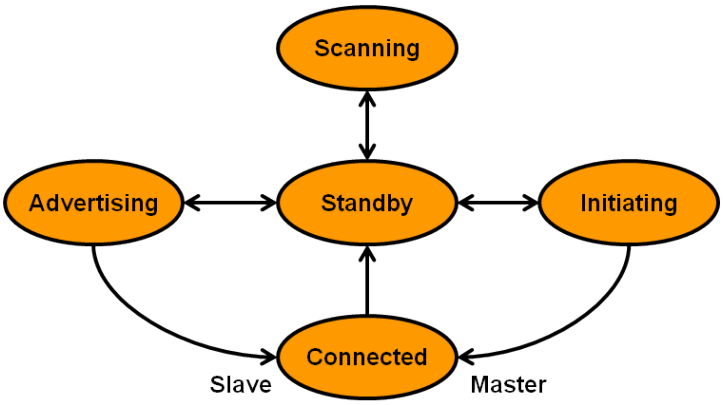


Figura 3.8: Estados da Camada de Ligação [49]

No estado de ***standby*** não são transmitidos ou recebidos quaisquer pacotes, e este estado pode ser acedido a partir de qualquer outro estado.

No estado de ***advertising*** é possível transmitir pacotes de *advertisement* (transmite os dados para todos os dispositivos ao seu alcance sem estar co-

nectado a nenhum deles) para a rede, existindo a possibilidade de receber e responder a pedidos desencadeados por esses mesmos pacotes de *advertisement*. Um dispositivo neste estado é chamado de ***Advertiser***.
Dispositivos que estão à "escuta"de pacotes de *advertisement* nos três canais destinados para o efeito são chamados de ***Scanners*** e como o nome indica este estado é o ***scanning***.
No estado de ***initiating*** ocorre a "escuta"por pacotes de *advertisement* vindos de dispositivos conectáveis, este processo desencadeia no dispositivo que estava à "escuta"uma resposta com o objetivo de iniciar uma conexão com o dispositivo conectável passando para o estado ***connected***. Neste novo estado o dispositivo que estava à "escuta"e iniciou uma conexão é considerado o ***Mestre***, por sua vez, o dispositivo conectável com quem foi estabelecida a conexão é considerado o ***Escravo*** da ligação.

Resumindo, a camada de ligação tem as seguintes funções definidas para a interação entre dispositivos:

- *Advertiser/Scanner*;
- Escravo/Mestre.

As atividades de *advertising* e *scanning* ocorrem em intervalos regulares. No entanto, os *Advertisers* e *Scanners* não estão sincronizados, portanto, estas atividades devem-se sobrepor.
Para evitar que os eventos de *advertising* de vários dispositivos se sobreponham, um pequeno tempo aleatório (0 a 10 ms) é adicionado entre os eventos de *advertising* (exceto pacotes direcionados para um dispositivo em específico).
Na figura 3.9 podemos visualizar um bom exemplo de como o processo de *advertising* e *scanning* devem estar configurados para ocorrer uma sobrepo-

sição de eventos e uma possível conexão se for o caso. Os dois dispositivos estão a executar as suas funções de *advertising* e *scanning* nos 3 canais existentes para o efeito (37-39), o dispositivo *Advertiser* está configurado para enviar os pacotes de *advertisement* nos três canais de 20ms em 20ms mais o tempo aleatório introduzido para evitar sobreposições de pacotes de *advertisement* entre dispositivos. Por sua vez, o dispositivo *Scanner* está à escuta nos mesmos canais, um de cada vez, periodicamente a cada 50ms durante 25ms de forma a ocorrer uma sobreposição de eventos como é demonstrado pelos retângulos tracejados a preto.

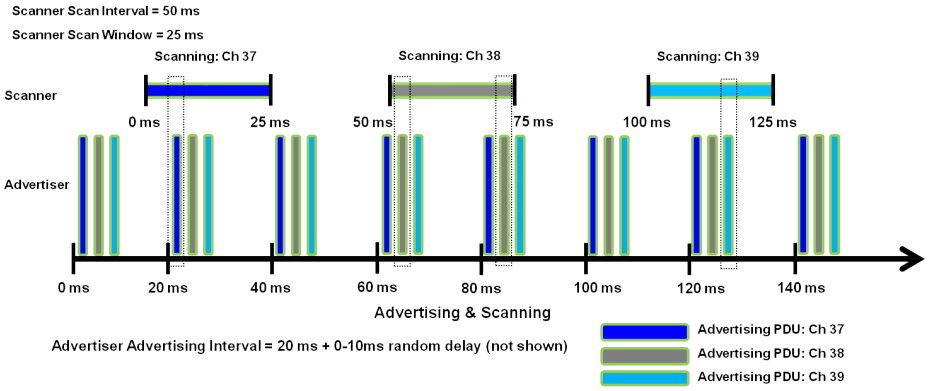


Figura 3.9: Processo de *Scan* e *Advertise* [49]

A camada de ligação na vertente de Mestre comunica com um ou mais dispositivos escravos e define os tempos de transmissão dos dados, enquanto na vertente de Escravo, comunica apenas com um dispositivo mestre nos tempos previamente definidos. Apesar de ser possível transmitir informação em pacotes de *advertisement* é preferencial, quando se pretende transmitir dados, a utilização de uma ligação Mestre-Escravo para este tipo de transações.

Para economizar energia, os Escravos estão por defeito no modo *standby* e acordam periodicamente para receber, se existirem, pacotes enviados pelo Mestre. O Mestre determina os instantes em que os escravos são obrigados

a acordar e, assim, coordena o acesso ao meio utilizando um esquema de acesso múltiplo por divisão de tempo. O Mestre também fornece ao Escravo as informações necessárias para o algoritmo de salto de frequências (incluindo o mapa de canais de dados a serem usados) e para a supervisão da conexão. Os parâmetros relacionados com a gestão de uma ligação são transmitidos na mensagem de solicitação de conexão e podem ser atualizados durante a ligação por vários motivos, como por exemplo, utilizar um novo mapa de canal de dados devido a uma alteração do padrão de interferência. Os parâmetros definidos no pedido de ligação são:

- Mapa do canais a utilizar;
- Intervalo de conexão - tempo periódico no qual os dispositivos comunicam (de 7.5 ms a 4 s);
- Número de eventos de conexão consecutivos que um Escravo não precisa "escutar"o Mestre, permanecendo no modo de *standby*;
- Tempo máximo sem resposta do outro dispositivo antes de uma conexão ser considerada interrompida.

Na figura 3.10 é demonstrado um caso em que os dispositivos Mestre e Escravo estão a trocar dados periodicamente a cada intervalo de conexão. No primeiro intervalo de conexão o dispositivo Mestre envia um pacote ao qual o dispositivo Escravo responde, no segundo intervalo ocorre a mesma situação mas em vez de um pacote, são enviados múltiplos pacotes. No terceiro intervalo de conexão o dispositivo Mestre tenta enviar um pacote ao qual não obtém um *acknowledged* pois o dispositivo Escravo permaneceu no estado de *standby*, como existe um número configurável de eventos consecutivos que um Escravo não precisa "escutar"o Mestre então a ligação não é quebrada. Neste caso a camada de ligação define que o dispositivo Mestre, como não recebeu o *acknowledged*, no próximo intervalo de conexão retransmita o mesmo

pacote até obter o *acknowledged* como é demonstrado no quarto intervalo de conexão.

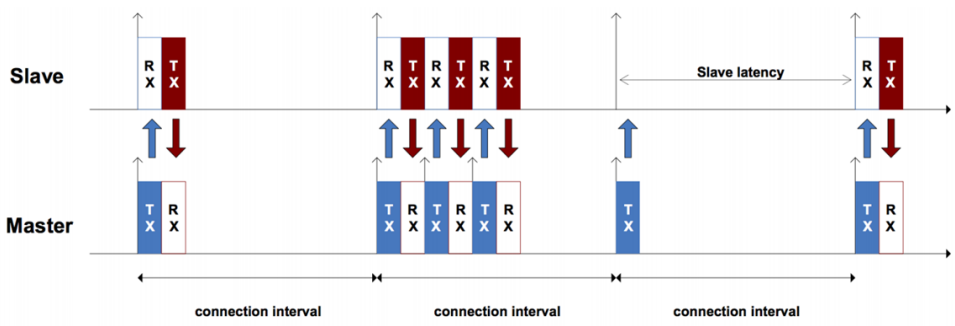


Figura 3.10: Troca de dados entre Mestre e Escravo [49]

Na figura 3.11 é demonstrado um processo de Conexão e *Broadcast*.
No processo de Conexão numa fase inicial, o Dispositivo A tem a função de ***Scanner*** e está à "escuta"de pacotes de *advertisement* enquanto o Dispositivo B tem a função de ***Advertiser*** e envia pacotes de *advertisement*. Quando o ***Scanner*** recebe os pacotes de *advertisement* inicia um pedido de estabelecimento de conexão com o ***Advertiser*** passando cada um deles a ter a função de Mestre e Escravo respetivamente.

No processo de *Broadcast*, o Dispositivo B tem sempre a função de ***Advertiser*** e envia pacotes de *advertisement*, enquanto o Dispositivo A e o Dispositivo N têm a função de ***Scanners*** e estão à "escuta"de pacotes de *advertisement* não estabelecendo uma conexão.

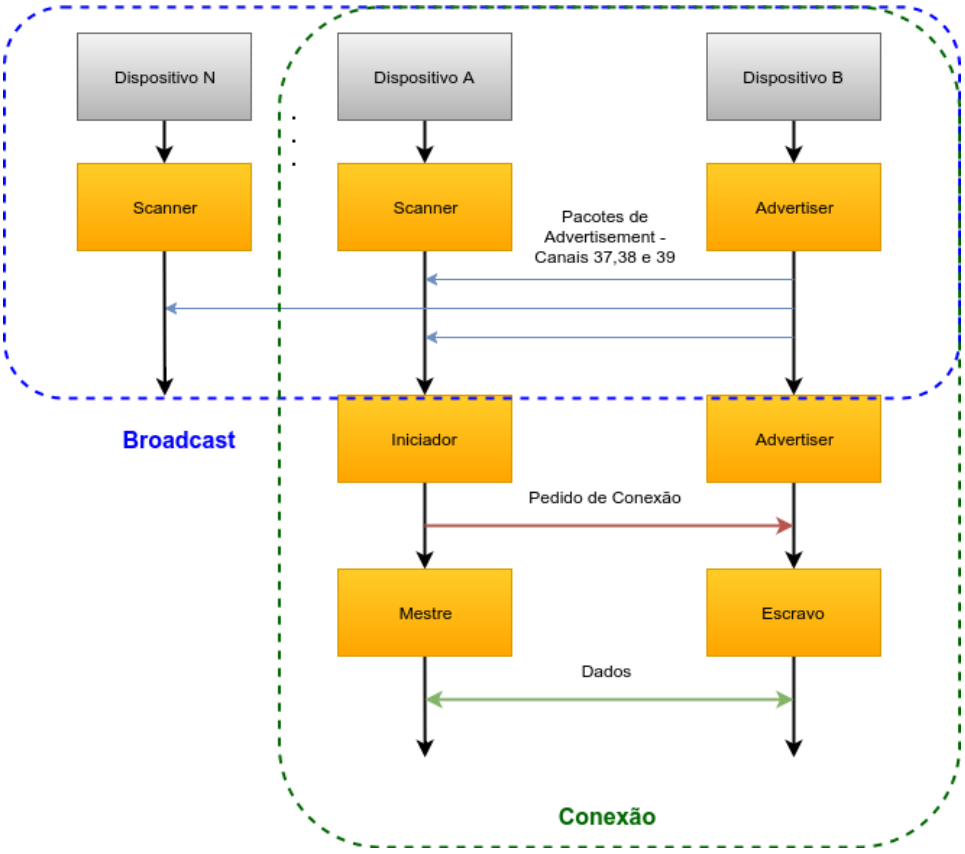


Figura 3.11: Camada de ligação: Processo de *Broadcast* e Conexão

Cada vez que um dispositivo *Advertiser* pretende transmitir, envia o mesmo pacote em cada um dos três canais de *advertising*. Existem dois tipos de pacotes de *advertisement* que podem ser enviados, os Conectáveis e Não Conectáveis. O que diferencia a situação de *Broadcast* ou de Conexão.

Dentro dos pacotes Conectáveis podemos dividir entre Genéricos ou Direcionados. Os pacotes Genéricos são enviados com o objetivo de procurar qualquer dispositivo *Scanner* ao alcance que pretenda estabelecer conexão. Por sua vez, os pacotes Direcionados são normalmente utilizados após um Escravo se ter conectado com um Mestre, entretanto a ligação é perdida, e uma vez que o endereço do dispositivo do seu par já é conhecido deseja

reconectar-se rapidamente ao mesmo.

Em relação ao processo de *Scan*, este processo pode ser feito em dois tipos: Passivo ou Ativo. No Passivo o dispositivo simplesmente recebe os pacotes de *advertisement*, enquanto que no Ativo o dispositivo pretende saber mais informação sobre o dispositivo que está a enviar os pacotes antes de decidir se pretende estabelecer uma ligação. Este processo de receber pacotes sobre o dispositivo que está a fazer *advertising* e obter informação sobre mesmo, como por exemplo, nome do dispositivo e *Universally Unique Identifier* (UUID) dos serviços, designa-se de Descoberta de um dispositivo.

A camada *Link Layer* só tem um formato de pacotes de dados usado tanto para *advertising* como para troca de dados como demonstra a figura 3.12.
Dentro do pacote de *Advertisement* temos o tipo Genérico e Direcionado. No Genérico podemos enviar várias estruturas de dados definidas no *standard* [50], como por exemplo, Nome Local Curto, Nome Local Completo, Lista de serviços, entre outros. Por sua vez, nos pacotes Direcionados como já se conhece os dados relativos ao dispositivo Mestre a conectar só se envia os endereços do Mestre e do Escravo.
No caso dos pacotes de dados são enviados sempre no mesmo tipo de pacote.

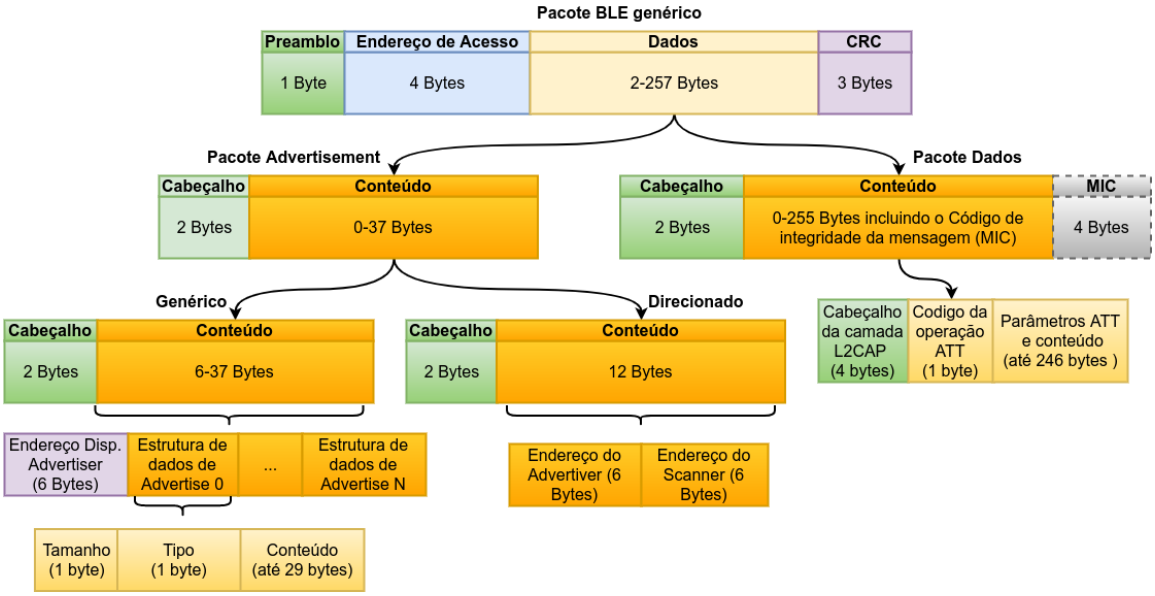


Figura 3.12: Estrutura dos pacotes na camada link layer

3.4.1.3 Security Manager

Uma conexão BLE opera num modo de segurança específico. Dentro de cada modo existem vários níveis de segurança. O modo e o nível de segurança necessários de uma conexão podem mudar ao longo do tempo da mesma.

Quando dois dispositivos que inicialmente não têm segurança pretendem fazer alguma operação que exija segurança, os dispositivos devem primeiro ser emparelhados. Este processo pode ser acionado, por exemplo, por um dispositivo Mestre que está a tentar obter os dados de uma característica de um Escravo que requer um acesso autenticado.
O emparelhamento envolve a autenticação da identidade dos dois dispositivos a serem emparelhados, geralmente por meio de um processo de compartilhamento de um chave de curta duração. Uma vez autenticada, a ligação é encriptada e chaves de longa duração são distribuídas para permitir que

numa nova reconexão a segurança seja reiniciada muito mais rapidamente. Se essas chaves forem guardadas para uma utilização futura, os dispositivos dizem-se *Bonded*.

O novo nível de segurança da conexão é baseado no método de emparelhamento realizado e isso é selecionado com base nos recursos de cada dispositivo. O nível de segurança de qualquer reconexão subsequente é baseado no nível alcançado durante o emparelhamento inicial.

Existem dois modos de segurança numa conexão que são apresentados nas tabelas 3.1 e 3.2.

Modo de Segurança	**Nível de Segurança**	**Autenticação**	**Encriptação**	**Proteção contra ataques de dispositivos entre o Mestre e o Escravo**	**Métodos de Autenticação**
1	1	Não	Não	Não	*Just Works*
1	2	Não	Sim	Não	*Just Works*
1	3	Sim	Sim	Sim	*Passkey Display* ou *Out of Band*
1	4	Sim	Sim	Sim	*Numeric Comparison*

Tabela 3.1: Modo 1 de Segurança

Modo de Segurança	**Nível de Segurança**	**Autenticação**	**Assinatura Digital**
2	1	Não	Sim
2	2	Sim	Sim

Tabela 3.2: Modo 2 de Segurança

Um procedimento de emparelhamento envolve uma troca de pacotes do protocolo *Security Manager* para gerar uma chave de encriptação de curta duração. Durante a troca de pacotes, os dois pares negociam um dos seguintes métodos de geração de uma chave de curta duração:

- *Just Works* é um modo projetado para possibilitar a conexão a dispositivos quando as interfaces com o utilizador são muito limitadas e impedem a inserção ou verificação de valores por parte do mesmo. Neste caso a chave de curta duração tem o valor 0. Este cenário é vulnerável a ataques;

- *Passkey Display* é utilizada quando as interfaces com o utilizador em ambos os dispositivos permitem pelo menos a exibição e a inserção de um valor numérico. Esta chave de curta duração não é complexa o suficiente para suportar ataques de força bruta pois é possível através de algorítimos descodificá-la [51];

- *Out of Band*, este método é aplicado quando a chave de curta duração foi compartilhada utilizando outra tecnologia que não o Bluetooth.

- Como dos 3 métodos apresentados só um apresenta segurança, o standard BLE 4.2 implementou o método *Numeric Comparison*, que utiliza um algoritmo designado de *Elliptic-curve Diffie–Hellman* (ECDH) para geração de chaves de longa duração e um novo método de emparelhamento seguro [52].

3.4.1.4 *Generic Access Profile*

A camada *Generic Access Profile* (GAP) é uma espécie de base de operações do BLE, coordena e define as operações fundamentais de acordo com o *standard* Bluetooth, tais como:

- Descobrir, estabelecer e terminar conexões a dispositivos;

- *Broadcast* de dados;

- Estabelecer conexões seguras (autenticação, emparelhamento ou *bonding*).

Esta camada é a interface direta com as aplicações e/ou perfis *Generic Attribute Profile* (GATT) (descritos na secção 3.4.1.6), sendo também responsável pela especificação dos papéis dos dispositivos, que podem operar numa ou mais funções do GAP ao mesmo tempo desde que seja suportado pela Camada de Ligação descrita na secção 3.4.1.2. O papel atribuído nesta camada impõe o comportamento do dispositivo. Dois pares de funções são definidos, permitindo que os dispositivos se comuniquem entre si. Os papéis definidos nesta camada são:

- *Brodcaster*/Observador
- Periférico/Central

O par Brodcaster/Observador implementa comunicações unidirecionais sem conexão. O *Broadcaster* apenas envia pacotes de *advertisement* não conectáveis e utiliza a função de *Advertiser* definida na Camada de Ligação.
O Observador, por sua vez, procura por *Broadcasters* através dos pacotes de *advertisement* que recebe e utiliza a função de *Scanner* da Camada de Ligação. Na figura 3.13 podemos ver o exemplo de uma tipologia *Brodcaster*/Observador, onde o dispositivo *Brodcaster* pode difundir a informação para n dispositivos *Scanners* ao seu alcance.

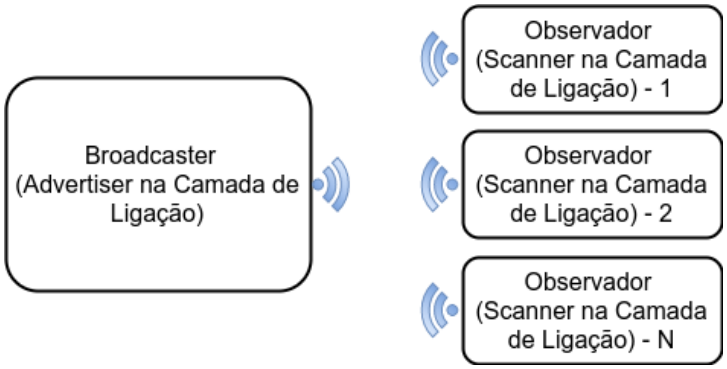


Figura 3.13: Topologia *broadcast*

Em relação ao par Periférico/Central, implementa comunicações bidireccionais orientadas para conexão. O Periférico envia pacotes de *advertisement* conectáveis e utiliza a função de Escravo definida na Camada de Ligação que consequentemente otimiza o dispositivo em termos de processamento e energia. O Central tem a capacidade de estabelecer e gerir conexões com dispositivos e utiliza a função de Master da Camada de ligação. Um dispositivo Central pode ligar-se a vários dispositivos simultaneamente. Na figura 3.14 podemos ver o exemplo de uma tipologia Periférico/Central.

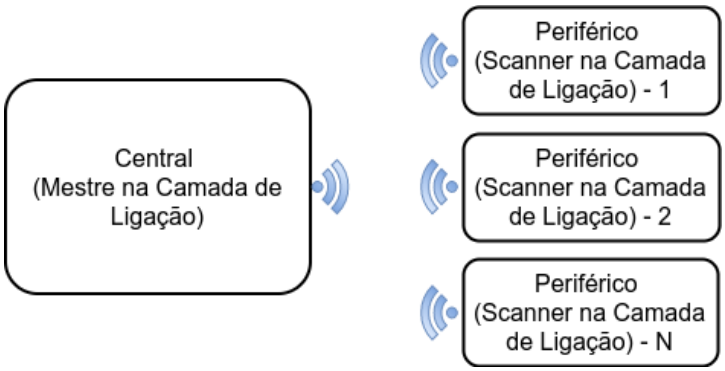


Figura 3.14: Topologia de Conexão

3.4.1.5 *Attribute Protocol*

O *Attribute Protocol* (ATT) é um protocolo Cliente/Servidor simples baseado em Atributos apresentados por um dispositivo. Um Atributo é uma pacote de dados que um Cliente solicita de um Servidor e consequentemente o Servidor envia para os seus Clientes.

Os Atributos são estruturas de dados que armazenam informações fornecidas pela camada GATT (descrita na secção 3.4.1.6). Cada Servidor contém os dados organizados sob a forma de Atributos que são compostos por 4 elementos como demonstra a figura 3.15, tais como:

- Endereço único do Atributo no Servidor - os valores do endereço cres-

cem normalmente numa sequência ordenada num Servidor e são descobertos pelo Cliente durante um procedimento de Descoberta. Para simplificar, pode ser considerado o número da linha em que está inserido na tabela de Atributos;

- UUID - define o tipo de Atributo (Característica, Serviço ou Perfil) que consequentemente vai determinar o tipo de dados presentes no conteúdo do Atributo;
- Conteúdo do Atributo - conteúdo de dados que é acessível por um Cliente. Também pode conter metadados sobre o Atributo (dependendo do tipo);
- Permissões do Atributo - operações permitidas no valor do Atributo (leitura, escrita ou nenhuma) e de segurança (encriptação e autorização).

Estrutura de um Atributo

Endereço do Atributo	Tipo de Atributo	Valor do Atributo	Permissoes do Atributo
2 Bytes	2 ou 16 Bytes	Tamanho variável (até 512 bytes)	Tamanho específico da implementação

Figura 3.15: Estrutura de um Atributo

No caso de uma operação de leitura, o Cliente deve analisar o valor e entender o tipo de dados com base no UUID do Atributo. Por outro lado, durante uma operação de escrita, espera-se que o Cliente forneça dados que correspondam ao tipo de Atributo e que o Servidor seja livre para rejeitar a operação se não for esse o caso.

3.4.1.6 *Generic Attribute Profile*

O GATT é uma camada construída em cima do ATT e define como enviar e organizar os Atributos da camada ATT para que dois dispositivos BLE transfiram dados entre si.

Nesta camada é definido o par de funções:

- Servidor/Cliente

O Servidor contém os dados a serem monitorizados organizados em Atributos e recebe solicitações de leitura de um Cliente enviando respostas de volta. Os papéis de Mestre/Escravo atribuídos na camada de ligação são independentes dos papéis de Cliente/Servidor definidos nesta camada.
Normalmente um Servidor GATT está associado às funções de um Periférico na camada GAP e consequentemente um Escravo na camada de Ligação. O Cliente faz solicitações de leitura de Atributos a um Servidor e interpreta as respostas. Normalmente um Cliente GATT está associado às funções de um Central na camada GAP e consequentemente um Mestre na camada de Ligação.

O GATT estabelece uma hierarquia para organizar os Atributos. Eles são organizados num Perfil de Servidor GATT e são agrupados da seguinte maneira (cf. figura 3.16):

- Serviços
  - Características
    - Atributos (Declaração, Valor e Descritor)

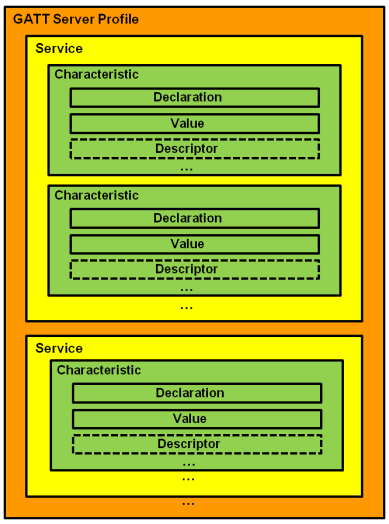


Figura 3.16: Hierarquia de um Perfil de um Servidor GATT [49]

Um Serviço é um conjunto de dados obtidos através de uma determinada função, como por exemplo, informação obtida de um sensor de temperatura. As características são essencialmente contentores para os dados, contêm no mínimo dois Atributos:

- Atributo da declaração da característica - contém as permissões da característica
- Atributo com o valor da característica - contém o valor da característica em si.

Os Descritores das características são Atributos usados para expandir ainda mais os metadados contidos no Atributo da declaração da característica, que dividem-se em:

- Propriedades estendidas
- Descritor da característica do utilizador - contém uma descrição legível pelo usuário para a característica em que é colocado, como por exemplo, "Temperatura na sala de estar em ºC".

- *Client Characteristic Configuration Descriptor* (CCCD) - fornece um mecanismo para habilitar as atualizações do valor das características iniciadas pelo Servidor, em que o Servidor pode enviar de forma assíncrona esses valores para o Cliente. Exitem dois tipos de atualizações possíveis:
  - Notificações, em que o Cliente não confirma a receção da atualização;
  - Indicações, em que o Cliente confirma a receção da atualização;

Uma analogia que se pode fazer é pensar no Perfil do GATT como uma sala de armazenamento em que os gabinetes são os Serviços e as gavetas são Características que contêm várias informações. Algumas das gavetas também podem ter fechaduras, restringindo o acesso à sua informação.

O dispositivo Cliente tem a possibilidade de aceder a estes Atributos através de operações de leitura e escrita, consoante as permissões implementadas. Na figura 3.17 temos um exemplo usual da estrutura de um Serviço disponibilizado por um Servidor.
Segundo as normas definidas pela *Bluetooth SIG*, o UUID que define um Atributo como serviço tem o valor 0x2800. Todos os Atributos seguintes pertencem a este serviço até ser encontrado novamente o mesmo UUID, significando assim o início de um novo serviço.
Cada serviço no campo Valor tem outro UUID, mas que neste caso especifica o tipo de serviço (e.g. sensor de pressão, sensor de humidade).
As características também têm um UUID específico para as representar, que é o 0x2803. Este valor também delimita uma característica, sendo que cada uma delas tem no mínimo dois Atributos: um para a identificar e outro para definir o seu valor.
Além destes dois Atributos, uma característica pode ter mais Atributos, ou

seja, os Descritores.

Existem vários descritores já definidos pela norma, em relação ao CCCD, cujo UUID é 0x2902, é um bitmap de 16 bits que podem ser lidos e escritos. É um Atributo como outro qualquer do lado do Servidor, sendo que os seus dois primeiros bits já estão definidos pela especificação do GATT, servindo para configurar se o valor da característica ao qual está associado será enviado para o Cliente sob a forma de notificação ou indicação. [54], [55], [56], [57]

Our Service	Handle	UUID Type of attribute	Attribute permission	Attribute value
Service Declaration	0x000X	**Service declaration** Standard $UUID_{Service}$ 0x2800	Read Only, No Authentication, No Authorization	Our custom service UUID 0x0000F00D-1212-EFDE-1523-785FEF13D123
Characteristic Declaration	0x000X	**Characteristic declaration** Standard $UUID_{characteristic}$ 0x2803	Read Only, No Authentication, No Authorization	Properties: Notify, Read, Write Value Handle (0x000X), Our custom characteristic UUID 0x0000BEEF-1212-EFDE-1523-785FEF13D123
Characteristic Value Declaration	0x000X	**Our Characteristic** UUID found in the Characteristic declaration value 0x0000BEEF-1212-EFDE-1523-785FEF13D123	Read Only, No Authentication, No Authorization (Configured by us)	Temperature value (of type int32_t, presented in array of 4 bytes. E.g. 0x00-00-00-65)
Descriptor Declaration	0x000X	**Client Characteristic Configuration Descriptor (CCCD)** Standard $UUID_{CCCD}$ 0x2902	Read and write, No Authentication, No Authorization	Notification enabled 0x00-XX

Figura 3.17: Exemplo da tabela de Atributos de um Serviço [57]

Capítulo 4

Arquitetura do Sistema Mobi.me

Neste capítulo pretende-se descrever a arquitetura do sistema existente na empresa de modo a detalhar o que já existia implementado previamente ao desenvolvimento desta tese.

4.1 A rede Mobi.me

O CEiiA desenvolveu a rede Mobi.me para uma gestão totalmente integrada e centrada no utilizador dos serviços de mobilidade, direcionados para os veículos elétricos, de determinadas regiões ou cidades.
O Mobi.me conecta todos os dispositivos de mobilidade e a infraestrutura de transporte como exemplifica a figura 4.1, estabelece um ecossistema onde uma sólida plataforma de gestão de transações permite aos utilizadores aceder a serviços de diferentes operadores de mobilidade através de um único mecanismo de autenticação, bem como, escolher o método de pagamento preferencial. O Mobi.me também está conectado ao sistema de energia, pos-

sibilitando a gestão do carregamento de veículos elétricos.

Atualmente a medição de qualidade do ar do Mobi.me é feita de forma indireta, ao estimar a quantidade de CO2 poupada por cada veículo elétrico que substitui um veículo a combustão.

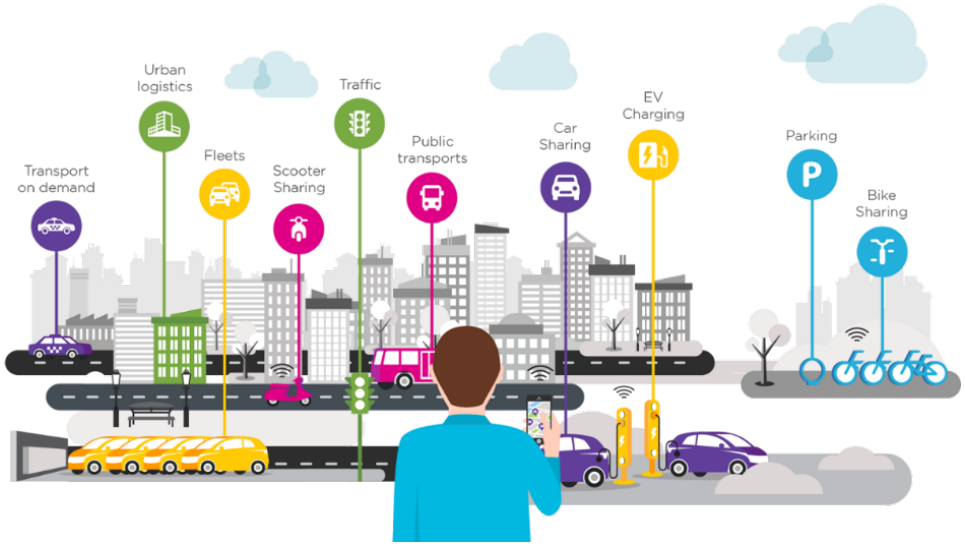


Figura 4.1: Rede Mobi.me

Nas figuras 4.2, 4.3 e 4.4 pode-se ver um dos exemplos da utilização real do sistema Mobi.me num serviço de aluguer de *scooters* elétricas ao minuto. A figura 4.2 demonstra em termos geográficos a intensidade da utilização dos veículos (a azul estão representadas as zonas menos utilizadas e a vermelho as zonas com maior intensidade de utilização), por sua vez, a figura 4.3 representa essa intensidade de utilização em termos temporais em cada dia da semana e por fim, a figura 4.4 representa a média das horas de utilização ao longo de um dia.

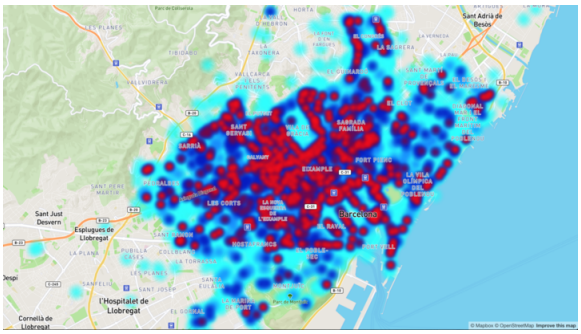


Figura 4.2: Mapa de intensidade de utilização do sistema Mobi.me em Barcelona

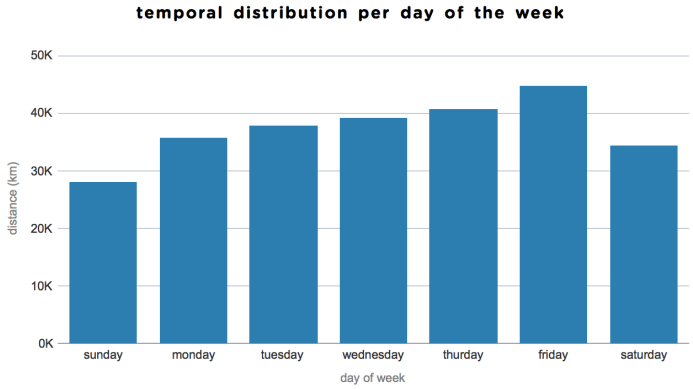


Figura 4.3: Gráfico de utilização do sistema Mobi.me em Barcelona: Quilómetros por dia da semana

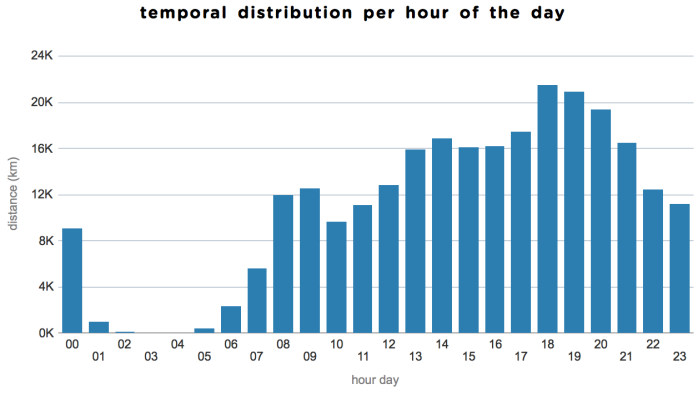


Figura 4.4: Gráfico de utilização do sistema Mobi.me em Barcelona: Quilómetros por hora do dia

4.1.1 Arquitetura do Sistema

Na figura 4.5 está ilustrada a arquitetura do sistema Mobi.me previamente ao desenvolvimento desta dissertação. Pretende-se demonstrar uma visão geral do fluxo de informação no sistema antes de explicar cada um dos blocos individualmente.

O CEiiA desenvolveu dispositivos designados de uMDC que são um dispositivos eletrónicos que permitem a interação dos utilizadores com os veículos. Um uMDC instalado num veículo, publica periodicamente dados sobre o seu estado atual num tópico do *Broker*, que por sua vez, reencaminha os dados para o Servidor. Quando o Utilizador pretende interagir com o veículo envia um pedido ao Servidor através da aplicação, o Servidor publica esse mesmo pedido no tópico do *Broker* ao qual o uMDC em questão está subscrito e consequentemente é executada a ação pretendida (e.g. alterar o nível da assistência do motor da bicicleta).

Desta forma, é possível disponibilizar serviços de partilha, *tracking* ou de logística urbana como equipamentos de carregamento de veículos elétricos.

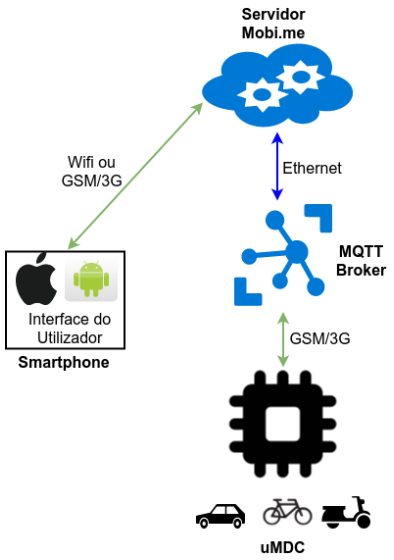


Figura 4.5: Arquitetura do sistema Mobi.me

4.1.2 *Micro Mobility Device Controller*

O uMDC é um dispositivo eletrónico inteligente, representado na figura 4.6, projetado para ser instalado em diferentes tipos de veículos, como carros, *scooters* e bicicletas.

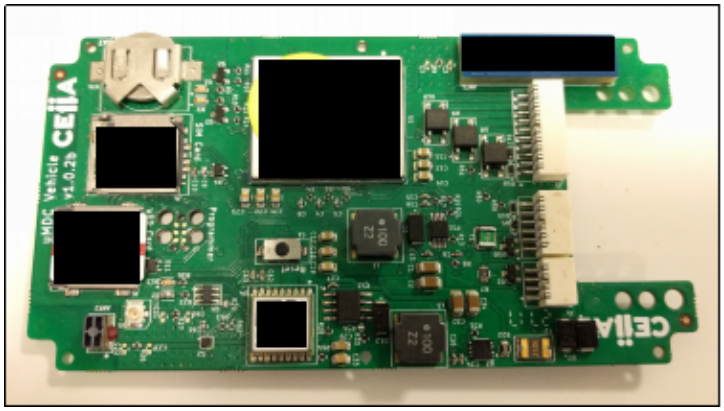


Figura 4.6: Exemplo de um uMDC

Quando instalado/integrado num veículo, fornece serviços de partilha e *tracking* para os utilizadores, que através de uma aplicação para *smartphone* podem interagir com o mesmo.

4.1.2.1 *Hardware*

As entradas e saídas elétricas do uMDC detetam e acionam partes específicas do veículo para fornecer funcionalidades de partilha do mesmo, como por exemplo, abrir/fechar ou ligar/desligar o mesmo. O uMDC é alimentado pela bateria do veículo e é conectado, se existir, ao barramento *Controller Area Network* (CAN) do mesmo para recolher dados relevantes do veículo. Este dispositivo utiliza módulos GPS e GSM para garantir precisão de posicionamento e relatórios de dados em tempo real para o Servidor. Como unidade de processamento este dispositivo conta com um microcontrolador

da Nordic modelo NRF52832 que contém integrado um módulo de BLE. O acelerómetro serve para indicar a ocorrência de movimento em veículos de menor porte, como por exemplo, bicicletas. O cartão de memória é utilizado principalmente para guardar os *logs* do sistema. Os principais módulos do uMDC estão representados na figura 4.7.

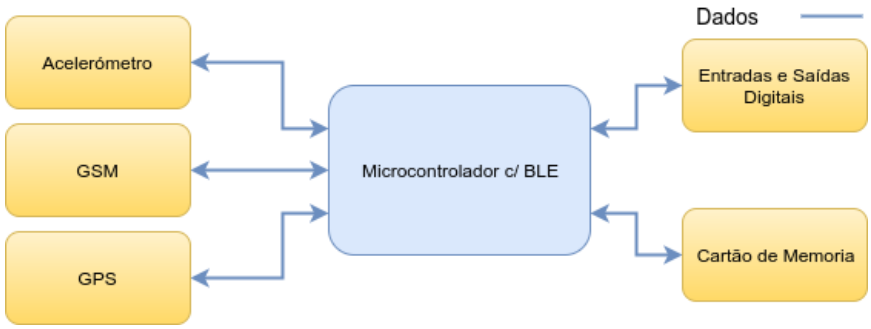


Figura 4.7: Diagrama de blocos geral do uMDC

4.1.2.2 *Software*

Devido à complexidade e ao número de diferentes módulos presentes no sistema, é necessária uma arquitetura de *software* simples, clara e modular. Para garantir essa simplicidade ao lidar com os vários módulos, utiliza-se um sistema baseado em máquinas de estado. Desta forma, cada módulo terá um modo de funcionamento bem definido com condições de transição claras entre estados. Para garantir ainda mais a separação de cada módulo do sistema e estruturar os diferentes componentes, os diferentes níveis do sistema serão separados nas seguintes categorias: *Managers*, Módulos e *Drivers* como demonstra a figura 5.11.

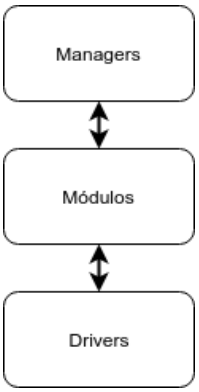


Figura 4.8: Arquitetura de *software* do uMDC

Os *Managers* implementarão a lógica do sistema, assegurando que as ações corretas sejam tomadas a cada momento e que os eventos acionem as transições corretas. Por exemplo, o *Manager* que gere uma bicicleta é responsável por decidir quando o dispositivo recolhe dados, reporta dados, vai "dormir", "acorda", entre outas ações.

Os Módulos são responsáveis por responder às diferentes solicitações dos *Managers* e redirecioná-las para o *Driver* correto. Os diferentes Módulos serão responsáveis por garantir que os *Managers* não acessam o *hardware* errado e que as ações solicitadas possam ser executadas naquele momento. Exemplos de módulos do uMDC podem ser: GPS, GSM, entre outros.

Os *Drivers* são responsáveis pela interface direta com o hardware. Ter uma camada entre os Módulos e o *hardware* permite uma interação mais segura entre as partes. Os *Drivers* garantem que todas as condições são verificadas antes de executar qualquer ação e fornecerá uma API clara que pode ser chamada por qualquer módulo. Ter um *Driver* também facilita a alteração da definição do *hardware* sem quebrar a compatibilidade com o que é implementado no restante do sistema. Por exemplo, se uma saída precisar ser

acionada, o Módulo chamará o *Driver* que interage com as entradas e saídas que por sua vez, será responsável por definir o valor correto no registro correto, habilitando ou desabilitando a saída pretendida.

4.1.3 *Broker Message Queuing Telemetry Transport*

A comunicação entre o Servidor e o uMDC é estabelecida utilizando um protocolo de comunicação proprietário sobre *Message Queuing Telemetry Transport* (MQTT).
O MQTT fornece uma maneira escalável e económica de conectar dispositivos pela *Internet*. É capaz de entregar mensagens quase em tempo real e tem opções que garantem a sua entrega. Este protocolo destaca-se quando o objetivo é conectar milhares de dispositivos com recursos limitados, que recebem e enviam notificações praticamente instantâneas.
O MQTT mantém a largura de banda a um mínimo absoluto, foi feito para manter uma conexão entre os dispositivos pretendidos a um custo mínimo. É baseado num sistema de publicação/subscrição o que permite uma fácil transmissão de mensagens de um publicador para vários subscritores ou de vários publicadores para alguns subscritores.
O *Broker* é um elemento fundamental numa rede MQTT, pois é o elemento que gere todas as publicações e subscrições. Apesar de o *Broker* representar um elo de fragilidade na rede ao centralizar as comunicações, ele permite um desacoplamento entre as partes comunicantes, algo que não é possível em modelos de comunicação do tipo Cliente/Servidor.

Na figura 4.9 é demonstrado um exemplo da utilização do MQTT num sistema de sensorização de temperatura, onde o sensor publica no tópico designado de *temperature* o valor de 21ºC e os dois dispositivos subscritos a esse tópico recebem quase em tempo real esse mesmo valor.

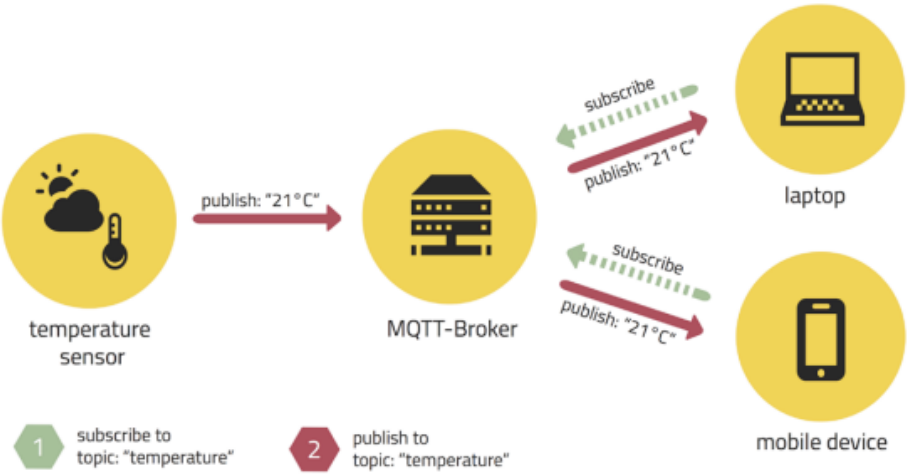


Figura 4.9: Broker MQTT - Sistema publicação/subscrição [58]

Além disso, a conexão é construída a partir do lado do Cliente, o que possibilita uma comunicação bidireccional sem problemas de conversão dos endereços de rede. Trata-se de um protocolo muito leve e binário, que se destaca quando se transferem dados por redes *wireless*, em comparação com protocolos como o *Hypertext Transfer Protocol* (HTTP), visto que o MQTT adiciona uma sobrecarga mínima aos pacotes de dados. Outro aspecto importante é que o MQTT foi projetado para ser fácil de implementar no lado do Cliente o que encaixa perfeitamente para dispositivos com recursos limitados. A comunicação MQTT é estabelecida com recurso a um *socket*. O *standard* para MQTT sobre *Secure Socket Layer* (SSL) define a utilização de um *socket* TCP/ IP para a porta 8883.

4.1.3.1 Segurança

Em relação à segurança, implementou-se o MQTT sobre SSL. O protocolo SSL visa principalmente fornecer privacidade e integridade de dados

entre dois dispositivos de comunicação. O SSL consiste numa chave pré-compartilhada, instalada no uMDC durante a produção. Essa chave contém três partes:

- Certificado de Autorização - é um certificado para autenticar o Servidor e garantir uma comunicação encriptada;
- Certificado do Cliente - é utilizado para autenticar o Cliente do lado do Servidor;
- Chave do Cliente - é utilizado para desencriptar a comunicação com o Servidor

Além do certificado de Cliente o uMDC também utiliza um *username* e *password* para se autenticar com o *Broker*.
O *socket* utilizado para comunicar por SSL é sempre iniciado pelo uMDC. O uMDC utiliza-o para garantir que apenas o Servidor possa desencriptar os dados transferidos.

4.1.3.2 *Quality of Service*

Há alguns tópicos a serem considerados ao implementar o MQTT. Um desses tópicos, diferentes de outras abordagens (e.g. HTTP), é a implementação do *Quality of Service* (QoS). É fundamental escolher um QoS adequado para o tipo de aplicação pretendido. Um maior QoS oferece mais previsibilidade, mas também exigirá mais mensagens, consumo de dados, tempo, memória e capacidade de processamento. Os níveis de QoS são:

- 0 - A mensagem no máximo é entregue uma vez, ou seja, o publicador envia uma vez e não garante que a mensagem é entregue;

- 1 - A mensagem é entregue pelo menos uma vez, ou seja, o publicador garante que a mensagem chega ao destino mas não garante que só uma mensagem foi entregue, podem ter sido entregues múltiplas mensagens iguais;

- 2 - A mensagem só é entregue uma única vez.

É importante frisar que a mensagem flui de um Cliente para o *Broker* e depois para outros Clientes. Cada mensagem tem o seu próprio QoS. Se algum Cliente especificar ao *Broker* que deseja receber mensagens com com QoS 1, mas os dados estiverem a ser enviados para o *Broker* com QoS 0, o QoS da comunicação será 0, porque o *Broker* não envia a mensagem novamente. Além disso, o Publicador nunca sabe quem recebeu a mensagem e o Subscritor, por sua vez, nunca sabe quem a enviou. Para garantir isso, deve ser implementada uma lógica sobre o MQTT.

4.1.3.3 *Concise Binary Object Representation*

Para o envio de dados é utilizado um modelo *Concise Binary Object Representation* (CBOR) sobre o MQTT. O CBOR foi motivado pelo aparecimento do IoT. Este é um formato de dados cujos objetivos são o envio de mensagens pequenas com o mínimo de *overhead* possível.
Ao utilizar o CBOR, reduzimos drasticamente a quantidade de dados que flui na rede e, devido a como ele é construído, também reduzimos extremamente o esforço de codificação/descodificação de dados no microcontrolador.
As duas principais estruturas CBOR utilizadas são os *arrays*, que podem ser de comprimento definido ou indefinido. Os *arrays* de comprimento definido são usados para todas as comunicações que chegam ao uMDC, enquanto os *arrays* de comprimento indefinido são usados para as mensagens enviadas pelo uMDC.

Por exemplo na tabela 4.1 é demonstrada a comparação do conteúdo de uma mensagem enviada em CBOR e em *JavaScript Object Notation* (JSON), que é outro tipo de modelo de dados mundialmente conhecido. Uma mensagem que em JSON ocupa 23 bytes em CBOR a mesma mensagem ocupa 15 bytes.

JSON	CBOR
[0,1,2,19999, "abcdefg"]	85 # *Array* de 5 posiçoes 00 # Número 0 01 # Número 1 02 # Número 2 19 4E1F # Número 19999 67 # Texto com 7 posições 61626364656667 # "abcdefg"

Tabela 4.1: Conteúdo CBOR versus JSON

4.1.4 *Servidor*

O Servidor é a plataforma que conecta e gere todos os dispositivos do sistema e permite ao utilizador interagir com os mesmos em forma de Serviços. O Servidor também disponibiliza uma interface *web* representado na figura 4.10 onde é possível visualizar o estado de cada dispositivo em tempo real, como por exemplo, as suas configurações, localização, última vez que reportou e recebeu dados, entre outros.

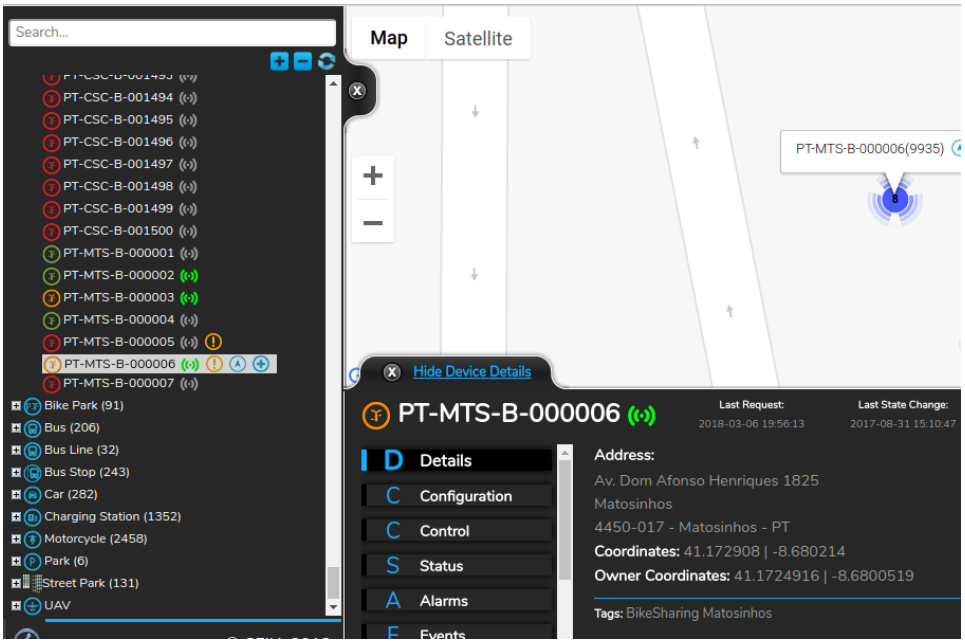


Figura 4.10: Servidor Mobi.me

4.1.5 *Smartphone*

O *Smarphone* é o ponto de interação do utilizador com sistema. Através de uma aplicação o Mobi.me disponibiliza serviços de mobilidade, como por exemplo, serviço de partilha de bicicletas, no qual um utilizador registado na rede pode usufruir do mesmo. O utilizador através da aplicação pode, por exemplo, alterar o nível da assistência do motor da bicicleta em utilização, carregar ou parquear uma bicicleta nas estações existentes para o efeito. Na figura 4.11 é demonstrado um dos interfaces disponibilizado ao utilizador para interagir com uma bicicleta.

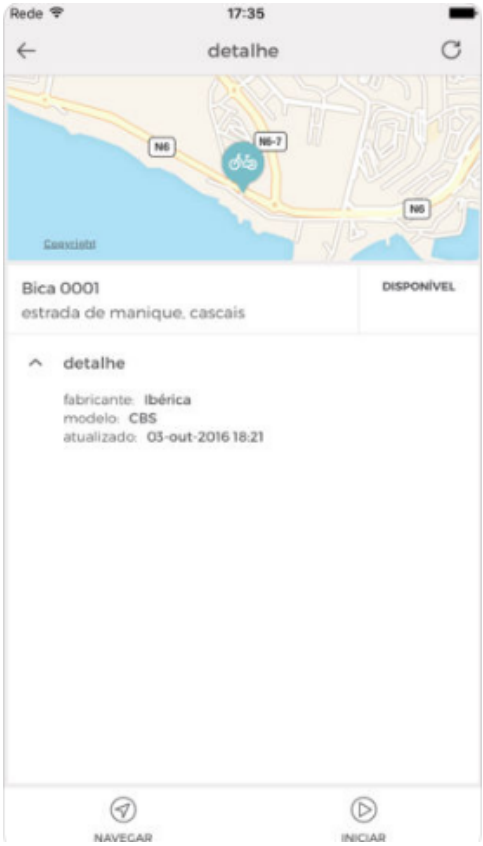


Figura 4.11: Aplicação do sistema Mobi.me

Capítulo 5

Arquitetura do Sistema Implementado

Neste capítulo pretende-se demonstrar o sistema implementado e sua integração com o que existia na empresa, criando uma visão geral e integrada do mesmo. Também são descritas as opções tomadas no decorrer do projeto e o princípio de funcionamento de cada um dos blocos desta implementação.

5.1 Arquitetura do sistema implementado

Tendo por base os requisitos e o principal caso de uso do sistema estruturou-se a arquitetura do mesmo demostrada na figura 5.1.
A Caixa de Sensores, como o nome indica é o dispositivo que vai conter os sensores a utilizar e ser responsável por recolher e transmitir os dados tanto para o uMDC como para o *Smartphone* via BLE. A Caixa de Sensores deve calcular métricas de cada um dos tipos de dados recolhidos a cada ciclo de leitura/envio dos mesmos. Delineou-se que a Caixa de Sensores deve calcular

os valores médios, máximos o mínimos de cada tipo de dados.

O uMDC e o utilizador através do *Smartphone* devem estar periodicamente à procura das Caixas de Sensores para estabelecer uma conexão com as mesmas (servir de *gateway* entre a Caixa de Sensores e o Servidor da mesma que analisa os dados), recolher os dados disponíveis, adicionar informações relativas à data/hora em que os mesmos foram recolhidos e reportar para o sistema via *GSM/3G* publicando num tópico do *Broker*, estes dois dispositivos também devem enviar coordenadas GPS válidas periodicamente.

No topo da arquitetura vai-se criar um Servidor de Caixas de Sensores, que através de um Cliente *MQTT* subscrito aos tópicos que tanto o uMDC como o *Smartphone* vão publicar para, de forma independente do Servidor do Mobi.me, interpretar os dados recebidos e demostrá-los num interface gráfico de forma simples e intuitiva. Os dados devem ser guardados e organizados entre Caixas de Sensores fixas e móveis, o utilizador deve ser capaz através de um interface gráfico de visualizar os últimos dados recolhidos e filtrar por tipo de sensor.

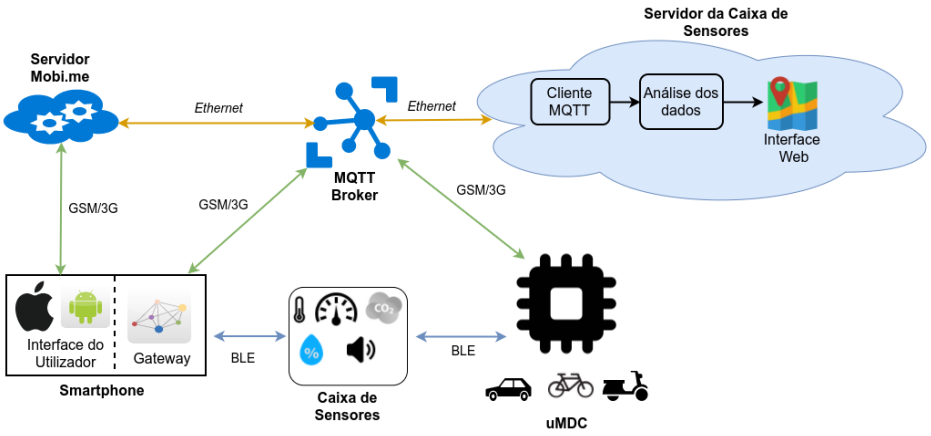


Figura 5.1: Arquitetura do Sistema Implementado

5.2 Caixa de sensores

Visto que a Caixa de Sensores é o bloco fundamental do sistema implementado e vai influenciar o modo de funcionamento de outros módulos, começou-se por definir a arquitetura deste bloco de maneira a cumprir com os requisitos do sistema definidos na secção 1.2.
Na figura 5.2 está ilustrada a arquitetura da Caixa de Sensores de maneira a cumprir com os requisitos estabelecidos, começando pela sua alimentação, a Caixa de Sensores deverá incorporar uma bateria com um sistema de carregamento para garantir a sua autossustentabilidade energética. A Entrada *Direct Current* (DC) pode advir da bateria de um veículo ou de um painel solar.
Tem de disponibilizar os barramentos de comunicação mais comuns neste tipo de aplicações para ligar vários tipos de sensores, tais como, ADC, I2C, SPI e UART. Para validar o conceito desta Caixa de Sensores utilizou-se três sensores com diferentes protocolos de comunicação, o sensor MH-Z16 [59] para detetar a concentração de CO2 que comunica via UART, o sensor BME280 [60] de temperatura, humidade e pressão que comunica via I2C e o sensor analógico *Grove Sound Sensor* [61] para detetar a intensidade do som ambiente. Para não ficarmos restritos ao nível de tensão do microcontrolador colocou-se conversores para suportar sensores com nível de tensão entre 3.3 V e 5 V.

Como unidade de processamento e para ser possível comunicar tanto com o uMDC como com o utilizador, escolheu-se o mesmo microcontrolador existente no uMDC pois além de disponibilizar todos os barramentos acima descritos, contempla um módulo BLE para comunicação *wireless*. O *System on a chip* (SoC) nRF52832 é um SoC multiprotocolo, altamente flexível (os pinos são configuráveis via *software*) e de baixo consumo, ideal para aplicações

Bluetooth de baixa potência. O SoC nRF52832 [62] é construído em torno de um *Central Processing Unit* (CPU) *Advanced RISC Machine* (ARM) Cortex -M4F de 32 bits com 512kB de *Read Only Memory* (ROM) + 64kB de *Random Access Memory* (RAM). Inclui um transceptor de 2.4GHz embutido que suporta BLE e uma *stack* proprietária. Ao escolher esta unidade de processamento contemplamos os requisitos do sistema em termos de comunicação pois assim é possível comunicar com o uMDC e com o utilizador visto que atualmente quase todos os *Smartphones* suportam BLE.

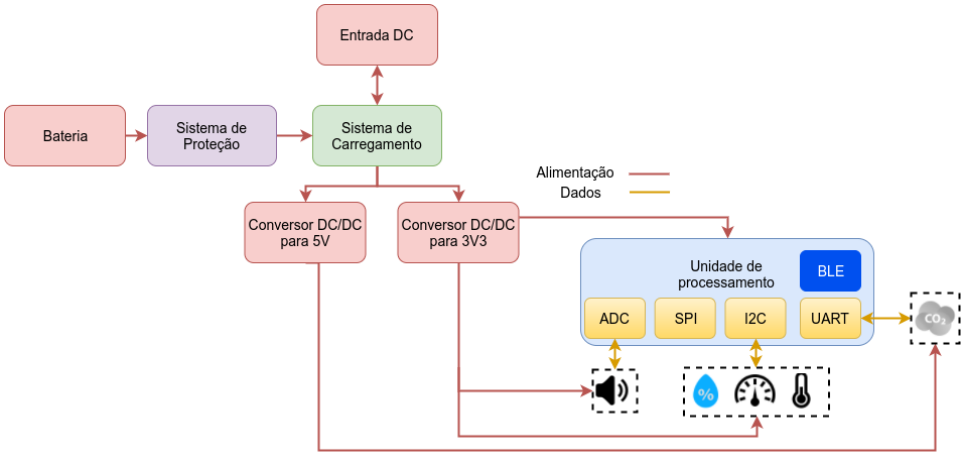


Figura 5.2: Definição da arquitetura da Caixa de Sensores

5.2.1 Sistema de Carregamento

Foi delineado um sistema com base no *Integrated Circuit* (IC) BQ24032A da *Texas Instruments* [63] que é responsável tanto pela gestão do carregamento da bateria como pela alimentação do restante sistema.

Este IC oferece uma gestão autónoma da fonte de energia a utilizar, entre uma entrada DC de até 16 V e uma porta *Universal Serial Bus* (USB) dando prioridade à que foi definida pelo utilizador, ambas com uma corrente máxima de 2 A.

O BQ24032A tenta alimentar a carga com uma tensão constante de 4.4 V

e a corrente solicitada pela mesma. Se a corrente solicitada pela carga for superior à disponibilizada pela entrada e consequentemente a tensão de saída cair para um valor abaixo da tensão pretendida, o IC vai diminuir a corrente de carregamento da bateria ou até parar o carregamento se necessário para estabilizar a tensão da saída. Por sua vez, se a corrente solicitada pela carga for inferior à disponibilizada pela entrada o BQ24032A vai utilizar essa potência de sobra para carregar a bateria até uma corrente máxima definida pelo utilizador.

Se não estiver presente nenhuma das duas possíveis fontes de alimentação, o IC alimenta o sistema recorrendo à bateria baixando a tensão de saída para o nível disponibilizado pela mesma.

Na figura 5.3 é apresentado o diagrama geral de funcionamento do BQ24032A. Os *Field Effect Transistor* (FET)s Q1, Q2, e Q3 são responsáveis por gerir o sistema, onde Q1 e Q3 têm a funcionalidade de ativar a entrada requisitada, se possível, e o FET Q2 serve tanto para alimentar o sistema através da bateria como para carregar a mesma.

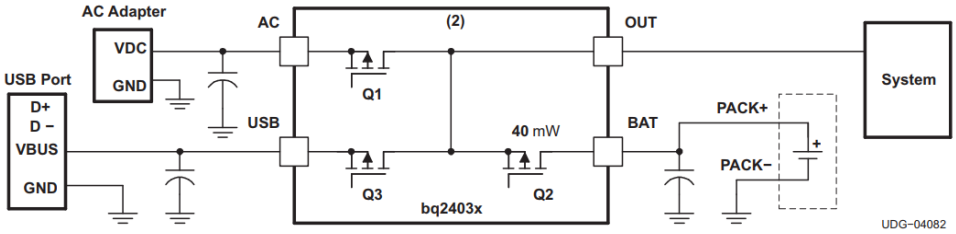


Figura 5.3: Diagrama Geral do IC BQ24032 [63]

O carregamento completo de uma bateria normalmente contempla três fases. A fase inicial de Pré condicionamento que é utilizada quando a bateria está abaixo da sua tensão mínima para detetar se a mesma está funcional, nesta fase é injetada uma corrente baixa e verifica-se que a bateria reage elevando rapidamente a tensão aos seus terminais. A fase de Carregamento Constante, em que a bateria é carregada a uma corrente constante até atingir a sua ten-

são máxima. A fase de Terminação em que a corrente de carregamento tende para 0 à medida que o carregamento é completado [64]. Na figura 5.4 demonstra um exemplo do perfil de carregamento de uma bateria de lítio. No estado designado de *Pre-Conditioning Phase* se a tensão da bateria estiver abaixo dos 3 V (valor tabelado) o IC vai tentar carregar a bateria com uma corrente de carregamento de acordo com a equação 5.1 em que os valores $V_{(PRECHG)}$ e $K_{(SET)}$ estão tabelados. O valor $R_{(SET)}$ foi dimensionado de acordo com a equação 5.2, para definir uma corrente de carregamento estipulada de $\approx 225\,\text{mA}$, que tendo em conta a capacidade da bateria a utilizar, que neste caso é de 2000 mAh o IC vai demorar cerca de $\approx$ 9h a fazer uma carga completa.

No estado *Pre-Conditioning Phase* o BQ24032 ativa um contador que define o tempo que a bateria tem para subir a sua tensão para um valor acima de 3 V, este tempo é um décimo do tempo de carregamento (50 minutos), caso a bateria não suba a sua tensão para o valor esperado significa que estamos perante uma falha grave da bateria e o seu carregamento não é possível. A bateria utilizada é de lítio polímero que têm uma tensão nominal de 3.7 V e normalmente são utilizadas neste tipo de aplicações.

O *Dynamic Power-Path Management* (DPPM) é responsável pela gestão do carregamento da bateria durante a fase de *Current Regulation Phase*. Nesta fase a bateria irá carregar com o valor estipulado de 225 mA ou menos, tendo em conta a potência disponível. Se a tensão no pino de saída baixar para um valor predefinido designado de tensão de DPPM), definida com o valor de 4.2 V, devido a uma quantidade limitada de corrente disponível ou remoção da alimentação de entrada, a corrente de carregamento da bateria será reduzida até que a tensão de saída estabilize. O DPPM tenta alcançar uma condição estável em que a carga receba a corrente necessária e a bateria seja carregada com a corrente restante.

Na fase de *Voltage Regulation and Charge Termination Phase* a bateria quando atinge a tensão de 4.2 V definida pelo IC, a corrente de carregamento diminui gradualmente até atingir a capacidade máxima da bateria concluindo o carregamento.

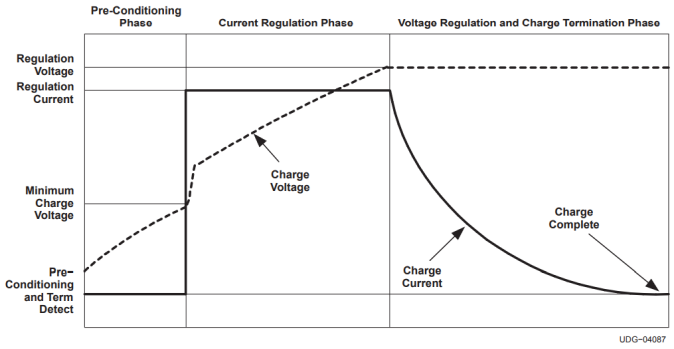


Figura 5.4: Perfil de carregamento do IC BQ24032 [63]

$$Io_{(PRECHG)} = \frac{V_{(PRECHG)} * K_{(SET)}}{R_{(SET)}} \tag{5.1}$$

$$Io_{(SET)} = \frac{V_{(SET)} * K_{(SET)}}{R_{(SET)}} \tag{5.2}$$

Disponibilizou-se um pino para reportar o nível da bateria que atualmente não é utilizado, pois pretende-se que a Caixa de Sensores seja autossustentável o suficiente para não termos que desligar os sensores por falta da mesma.

5.2.2 Sistema de Proteção

Juntamente com o sistema de gestão de carregamento a *Texas Instruments* recomenda a utilização de um IC designado de BQ2970 [65] que é um dispositivo de proteção de células de baterias de lítio contra situações de sobretensão, subtensão, carregamento ou descarregamento muito rápido. Ba-

sicamente este IC em condições de carregamento/descarregamento normais permite a ligação do *Ground* da bateria ao do restante sistema. Se o IC detetar alguma falha no sistema, através dos *Metal Oxide Semiconductor Field Effect Transistor* (MOSFET)s representados na figura 5.5 de CHG e DSC é possível desabilitar a ligação do *Ground* da bateria ao resto do sistema desabilitando a mesma.

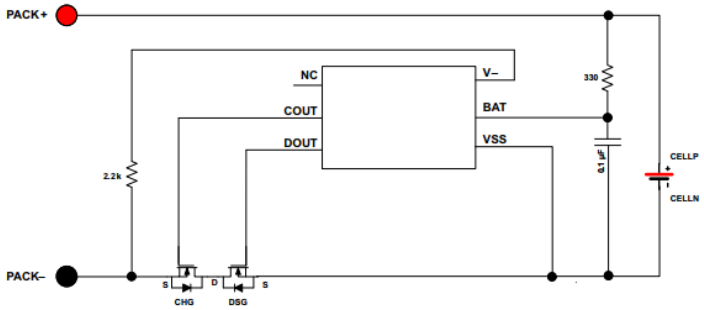


Figura 5.5: Diagrama Geral do IC BQ2970 [65]

5.2.3 Conversores DC/DC

Foram definidos dois conversores DC/DC para obtermos 2 níveis de tensão fundamentais, 3.3 V para alimentar o microcontrolador e sensores e 5 V para o sistema não estar limitado a sensores com uma tensão de alimentação de 3.3 V. Nesta secção importa destacar que foram escolhidos 2 conversores compatíveis com o sistema de carregamento. Na figura 5.6 demonstra-se as curvas de caraterística para os conversores de 3.3 V e 5 V, onde se demonstra que o conversor de 5 V fornece cerca de 850 mA e o conversor de 3.3 V fornece cerca de 1200 mA à tensão minima da bateria (3 V).

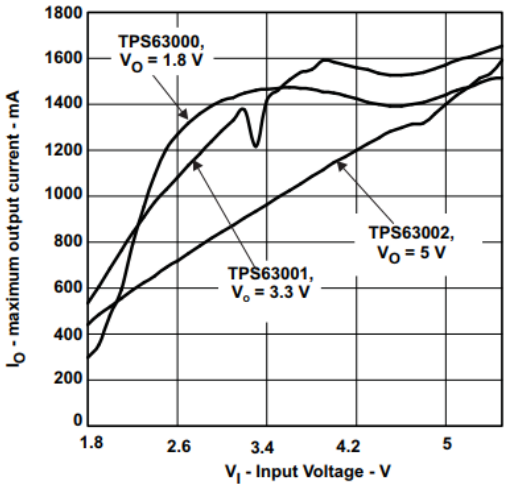


Figura 5.6: Curva de característica dos conversores DC/DC [65]

5.2.4 Modo Poupança de Energia

Devido a este sistema ter restrições energéticas é fundamental existir a possibilidade via *hardware* de reduzir o consumo energético e só deixar o microcontrolador em funcionamento. Visto que o conversor DC/DC para 3.3 V deve estar sempre em funcionamento para fornecer a alimentação necessária ao microcontrolador, utilizou-se o *TPS27081A* [67] que é um IC que funciona como um interruptor, como demonstra a figura 5.7, para habilitar/desabilitar este nível de tensão para os outros periféricos, neste caso sensores ligados a este nível de tensão. Quando se coloca o nível lógico 1 (acima de 1 V) na entrada designada de *General Purpose Input/Output (GPIO)/Logic* faz com que o MOSFET Q2 conduza o que consequentemente leva a que o MOSFET Q1 também conduza ligando o *VIN* ao *VOUT*.

Como o conversor DC/DC para 5 V tem um pino de habilitar/desabilitar e não alimenta nenhum componente que tenha que estar permanente ligado, o controlo da alimentação dos periféricos ligados a este nível de tensão é feito diretamente ao nível do conversor.

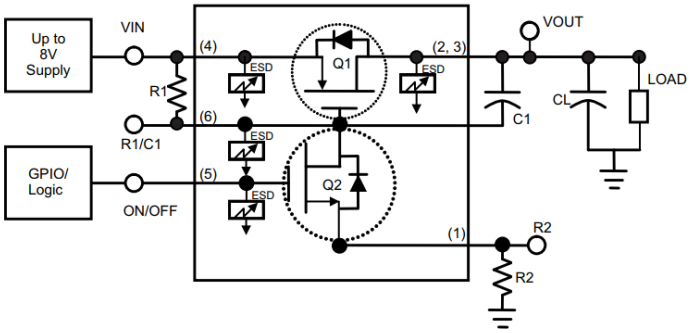


Figura 5.7: Diagrama Geral do IC TPS27081A [67]

5.2.5 Comunicação com os sensores

Visto que a Caixa de Sensores disponibiliza os barramentos de comunicação mais comuns neste tipo de aplicações para ligar vários tipos de sensores. Foi delineado a utilização de *level shifters* bidireccionais de 3.3 V para 5 V e vice-versa em todos os canais de comunicação com os sensores para os mesmos não estarem limitados ao nível de tensão do microcontrolador.

5.2.6 Arquitetura de *Software*

Foi definida uma arquitetura simples e modular de modo a ser possível adicionar e alterar sensores com o menor impacto possível e ser compatível com a estrutura disponibilizada pela API da Nordic. O objetivo é construir a arquitetura de *software* de modo a que, para adicionar um novo sensor exista o menor número de dependências possível.

Na figura 5.8 está demonstrada a arquitetura global de *software* incluindo a API disponibilizada pela Nordic. Na camada mais abaixo temos o *hardware*, ou seja, o microcontrolador disponibilizado pela Nordic. O SoftDevice é uma camada composta por 3 componentes, a *stack* BLE que contém toda a implementação do protocolo, bibliotecas para recursos de *hardware* parti-

lhados entre a Aplicação e o SoftDevice, como por exemplo, os *Timers* e a Memória Flash. Se o SoftDevice estiver ativo deve ser utilizada esta API para os recursos partilhados e não a disponibilizada pelos *Drivers* dos Periféricos. O SoftDevice Manager serve para habilitar/desabilitar o SoftDevice consoante a necessidade, por exemplo, numa aplicação que não utilize BLE o SoftDevice não necessita de estar ativo.

A API do SoftDevice é composta por 2 blocos e está disponível para a camada do software desenvolvido como uma interface em linguagem de programação C. A API de Protocolo é utilizada para aplicações sem fio. A API nRF, expõe as funcionalidades do SoftDevice Manager (habilitar/desabilitar) e das bibliotecas que podem ser partilhadas por Aplicação e SoftDevice. Como o SoftDevice é fornecido como um binário pré-compilado, os *SuperVisor Calls* (SVC) são um método utilizado para expôr as funções do SoftDevice sem ter nenhuma dependência entre o *Software* desenvolvido e manter a implementação do protocolo BLE proprietária e de acordo com o *standard*. O SoftDevice transmitirá ações para a Aplicação utilizando eventos, como por exemplo, quando recebe um novo pacote, quando uma conexão foi estabelecida, quando a conexão foi terminada, etc. Os eventos são o *feedback* recebido pela Aplicação após chamar a API do SoftDevice.

Os *Drivers* dos periféricos é uma camada que disponibiliza uma API de *Drivers* para configurar, comunicar e gerir cada um dos periféricos do microcontrolador. O *Driver* do GPIO permite configurar um pino como entrada ou saída, ler ou escrever nesse mesmo pino. O *Driver* dos *Timers* permite configurar e iniciar um contador com determinada frequência, se é para ser executado uma única vez ou periodicamente. Os *Drivers* de comunicação como o I2C ou a UART disponibilizam funções para ler e escrever sobre determinado protocolo.

Para a camada do *software* desenvolvido da Caixa de Sensores delineou-se uma estrutura similar à que existe no uMDC, que contempla uma organiza-

ção dividida em *Drivers*, Módulos e *Managers* representada na figura 5.9. Previamente a desenvolver a camada de Aplicação, uma grande parte do trabalho em termos de *software* consiste em estudar o funcionamento da API disponibilizada pela Nordic de modo a adaptar da melhor forma à Aplicação a desenvolver.

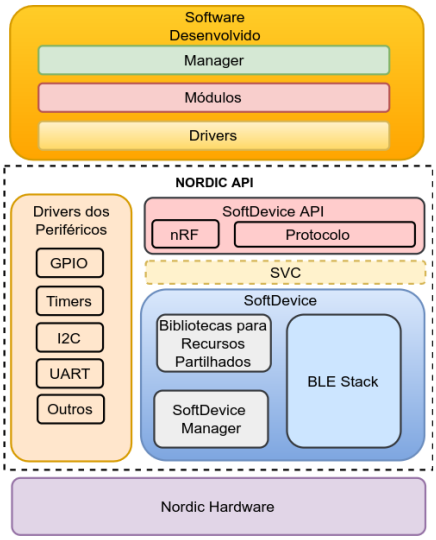


Figura 5.8: Arquitetura global do *software*

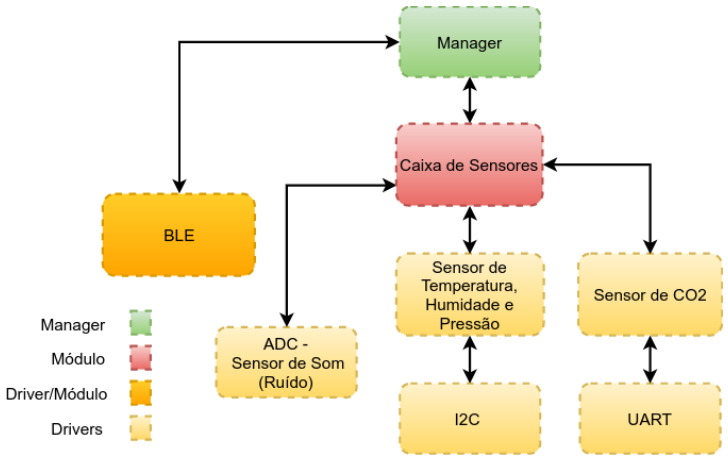


Figura 5.9: Caixa de sensores: Arquitetura de *software*

Na camada dos *Drivers* da Aplicação pretende-se desenvolver uma camada de abstração para a API disponibilizada pela Nordic quer dos periféricos, quer do SoftDevice. Esta camada além de ser um *Wrapper* das funções da Nordic, também pretende implementar a lógica necessária para interagir com cada um dos periféricos.

Os *Drivers* de comunicação I2C, UART e ADC devem disponibilizar uma API simples e clara para configurar e interagir diretamente com os periféricos permitindo operações de escrita e leitura sobre determinado protocolo de comunicação.

Por sua vez, os *Drivers* que implementam a lógica de funcionamento de cada um dos sensores, contemplam as mensagens necessárias a enviar para configurar/calibrar e requisitar leituras do sensor, devendo também interpretar e validar as mensagens enviadas pelo mesmo. Deste modo é possível ter vários sensores que falem, por exemplo, UART e consequentemente utilizem o mesmo *Driver* de comunicação mas cada um com as suas próprias especificidades. Na figura 5.10 é demonstrado o princípio de funcionamento delineado para a interação entre o *Driver* do sensor de CO2 e o *Driver* de comunicação UART. É importante realçar que o código desenvolvido da Caixa de Sensores vai ser em C/C++ de forma a aproveitar o conceito de herança (uma classe herda as funcionalidades de outra) e classes abstratas, ou seja, classes com funções virtuais (sem implementação) que permitem que cada sensor tenha a sua própria implementação de determinada função. Na figura 5.10 o sensor de CO2 tem a sua própria implementação da função de análise das mensagens recebidas via UART de forma a poder gerir esses dados e validá-los de acordo com as características deste sensor em particular. Utiliza a função de escrita de mensagens da UART para calibrar e requisitar pedidos de leitura.

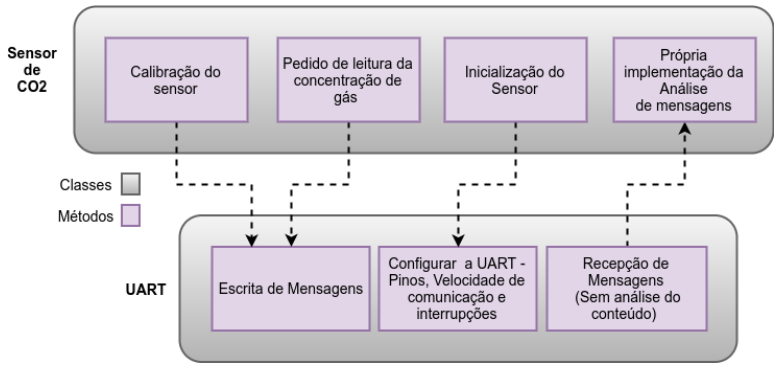


Figura 5.10: Caixa de sensores: Arquitetura dos *Drivers* - Sensor de CO2 + UART

A interação entre os *Drivers* de I2C e Sensor de Temperatura, Humidade e Pressão deve ter um funcionamento muito idêntico ao representado na figura 5.10 entre o *Driver* da UART e o sensor de CO2, onde o I2C vai disponibilizar funções de leitura e escrita de acordo com o protocolo e o Sensor de Temperatura, Humidade e Pressão implementa as funções de acordo com as especificidades do sensor, como por exemplo, funções de calibração, pedidos de leitura e análise das mensagens recebidas.
O *Driver* ADC deve conter o sensor de intensidade do som/ruído, pois o mesmo não é mais do que a leitura do valor de um pino do microcontrolador, logo não existe a necessidade de criar um *Driver* para esse efeito. Este *Driver* deve simplesmente permitir a configuração do pino e da frequência de amostragem à qual se pretende efetuar as leituras.

Da maneira que foi delineada a arquitetura, para obtermos dados de um novo sensor é necessário criar um novo *Driver* que vai herdar, se necessário, as funcionalidades de um *Driver* de comunicação (e.g. I2C) já existente, implementar as funções necessárias para interagir com o sensor em questão e disponibilizar essa API para o módulo da Caixa de Sensores gerir o seu funcionamento.

O *Driver*/Módulo BLE é representado como tendo duas funcionalidades pois a Nordic disponibiliza uma API BLE que interage diretamente com o *hardware*, mas em termos de *software* a desenvolver é a primeira camada a interagir com o módulo de *hardware* BLE. Este *Driver*/Módulo deve ser genérico, pois deve ser o mesmo tanto para a Caixa de sensores como para o uMDC. É responsável por disponibilizar as funções necessárias e gerir a comunicação com outros dispositivos Bluetooth, como por exemplo, disponibilizar as funções para: inicializar a *stack* disponibilizada pela Nordic, configurar se o dispositivo vai fazer *Scan* ou *Advertise*, se após estabelecer conexão é um dispositivo Periférico ou Central e consequentemente ter o papel de Mestre ou Escravo na conexão, se é um Servidor ou Cliente GATT, ou seja, se vai disponibilizar ou requisitar atributos. Neste caso em concreto, a Caixa de sensores vai estar periodicamente a enviar pacotes de *advertisement* para tentar estabelecer uma conexão com um dispositivo Central. Após estabelecer essa conexão a Caixa de Sensores como dispositivo Periférico vai assumir o papel de Escravo na ligação. Na camada GATT vai assumir o papel de Servidor disponibilizando um Serviço com N características, três por cada tipo de dados a reportar, tais como, uma característica para a temperatura máxima, outra para a temperatura média e outra temperatura mínima e assim sucessivamente até contemplar todos os tipos de dados. Também devem ser disponibilizadas 3 características extra, uma para o número de série da Caixa de Sensores que disponibilizou os dados, outra para o tempo que passou desde a recolha dos dados pela Caixa de Sensores até ao instante que o uMDC ou *Smartphone* recolheram os dados e por fim, uma para questões de escrita, tais como, configuração do ciclo periódico de recolha/envio dos dados, o número de leituras de cada um dos sensores e para a inserção da chave de segurança.

O módulo da Caixa de Sensores é responsável por inicializar, gerir todos os *Drivers* e definir os momentos em que cada um deve interagir com o *hardware*, como por exemplo: configurar os períodos de envio dos dados, o *advertising*, se é o momento para ler algum dos sensores em questão, se é para ligar/desligar a alimentação dos sensores para poupar energia ou se é necessário calcular métricas e fazer *advertising* para tentar reportar os novos dados.

A aplicação é a camada ao mais alto nível que serve para inicializar os módulos pretendidos. Se existir uma conexão estabelecida via BLE e o módulo da Caixa de sensores indicar que tem novos valores para reportar, deve enviá-los para o *Smartphone* ou para o uMDC.

5.3 *Micro Mobility Device Controller* e *Smartphone*

Tanto o uMDC como o *Smartphone* vão ter comportamentos similares nesta arquitetura, fazendo de *gateway* para os dados reportados pela Caixa de Sensores, apesar de serem desenvolvidos com uma linguagem de código diferente o uMDC em C/C++ e o *Smartphone* em *Android*.

5.3.1 *Micro Mobility Device Controller*

O uMDC vai tentar estabelecer conexões com a Caixa de sensores quando esta estiver instalada no veículo para reportar os dados ao longo de uma viagem ou quando o veículo estiver estacionado. Se o veículo em questão não tiver nenhuma Caixa de Sensores instalada, o uMDC vai estar periodicamente à procura de possíveis Caixas de sensores Fixas.
Delineou-se a arquitetura descrita na figura 5.11 para o *software* utilizado no

uMDC. O *Driver*/Módulo BLE é o mesmo utilizado na Caixa de Sensores com as mesmas funcionalidades, ou seja, gerir a comunicação com outros dispositivos Bluetooth e definir qual o papel que este dispositivo vai ter na comunicação. Neste caso em específico vai fazer *Scan* por pacotes de *advertisement* até encontrar uma Caixa de Sensores, após estabelecer conexão assume o papel de dispositivo Central e consequentemente vai ter o papel de Mestre na ligação, bem como, ser um Cliente GATT, ou seja, vai requisitar o valor das características disponibilizados pela Caixa de Sensores de modo a obter cada um dos tipos de dados recolhidos, como por exemplo, temperatura máxima, mínima e média, Pressão máxima, mínima e média, entre outros.

Após obter os dados via BLE o uMDC também deve adicionar informação relativa à data/hora em que os dados foram recolhidos, deve comparar a data e hora atual do uMDC (obtida através do serviço *Network Time Protocol* (NTP)) com o tempo que já passou desde que os dados foram recolhidos pela Caixa de Sensores. Periodicamente, o uMDC deve publicar as coordenadas GPS válidas de modo a que quando publicar os dados recolhidos através da Caixa de Sensores (já referenciados temporalmente), estes possam, no Servidor da Caixa de Sensores que está a analisar as mensagens publicadas, ser associados a uma coordenada GPS válida. Todas estas mensagens são enviadas via módulo de GSM para o *Broker* MQTT.

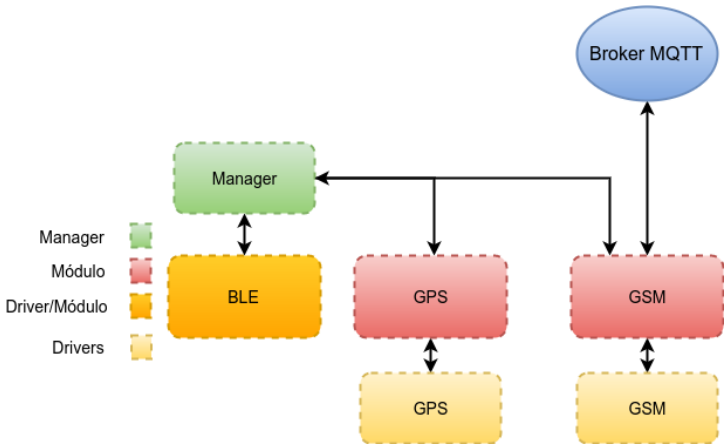


Figura 5.11: Arquitetura de *software* do uMDC

5.3.2 *Smartphone*

O *software* do *Smartphone* incidiu sobre o desenvolvimento de um *gateway* para que o mesmo tenha um papel similar ao uMDC, para isso delineou-se a arquitetura demonstrada na figura 5.12, a Aplicação após obter as permissões necessárias para interagir com os módulos de BLE, GPS e Interface de rede, deve estar periodicamente à procura de Caixas de Sensores. Quando conseguir estabelecer uma conexão BLE com uma Caixa de Sensores vai fazer o papel de Cliente GATT e requisitar o valor das características disponibilizados pela mesma, deve também, como no uMDC, adicionar informação relativa a data/hora em que os dados foram recolhidos. Em paralelo, o *Smartphone* também deve ir atualizando as últimas coordenadas GPS válidas de modo a que quando pretender publicar os dados recolhidos através da Caixa de Sensores possa publicar as últimas coordenadas válidas de modo a cumprir com o que está implementado no uMDC. Como os *Smartphones* têm uma boa capacidade de armazenamento não precisam de publicar os dados recolhidos com uma periodicidade tão baixa como o uMDC no *Broker* MQTT, podem periodicamente verificar se têm uma ligação à *internet* e ao *Broker* e publicar os dados que foram sendo armazenados. Estas mensagens

são enviadas via interface de rede (WiFi ou 3G) para o *Broker* MQTT através de um protocolo proprietário da empresa.

No futuro este processo do lado do utilizador deve ser transparente para o mesmo ao incorporar este serviço numa aplicação *Android* já existente no CEiiA), atualmente pretende-se desenvolver uma aplicação em que com o mínimo de interação seja possível fazer este processo de forma autónoma.

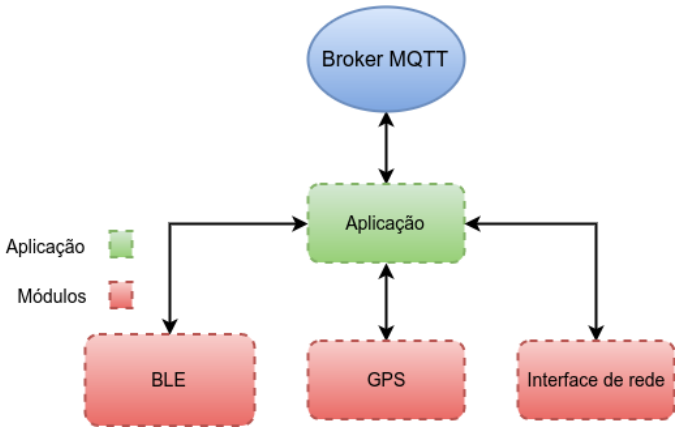


Figura 5.12: Arquitetura de *software* do *Smartphone*

5.4 Servidor da Caixa de Sensores

Para provar o conceito de funcionamento deste sistema sem influenciar o comportamento atual do Servidor do Mobi.me, foi implementado outro Servidor que através de Cliente *MQTT* em Python, descrito na figura 5.13, que vai subscrever aos tópicos que tanto o uMDC como o *Smartphone* vão publicar, para interpretar/validar os dados recebidos, organizá-los e demonstra-los num interface gráfico. Os dados vão ser organizados por Caixas de Sensores fixas ou móveis (recorrendo a uma lista com as Caixas de Sensores fixas existentes).

Dentro das Caixas de Sensores fixas os dados recebidos devem ser agrupados pelo número de série da mesma visto que já sabemos as suas coordenadas

(obtidas através da lista). Nas Caixas de Sensores móveis os dados devem ser agrupados relativamente á ultima coordenada reportada pelo dispositivo que recolheu os dados da Caixa de Sensores.

Após os dados estarem organizados e validados, um interface gráfico em *javascript* com base num mapa deve representar esses mesmos dados de forma simples e intuitiva, sendo possível filtrar por tipo de sensor. Também deve ser possível visualizar a evolução temporal de cada um dos tipos de dados recolhidos pelas Caixas de Sensores através de um *script* que ao inserir o *serial* da Caixa de Sensores e o tipo da mesma, se é fixa ou móvel, apresenta um gráfico com a evolução temporal desse parâmetro.

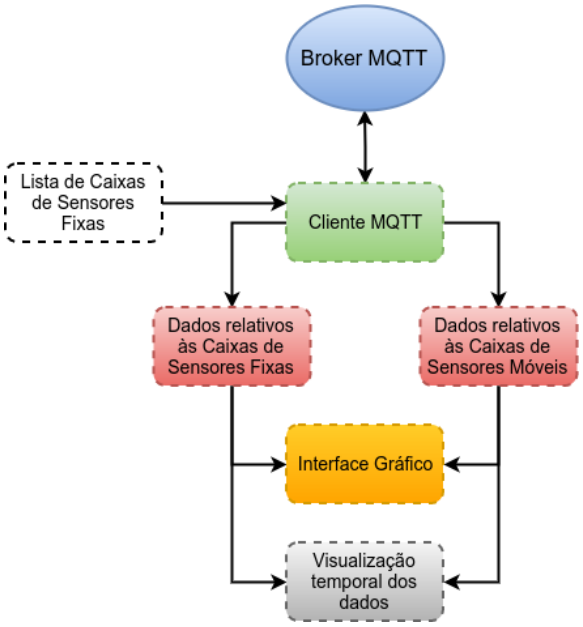


Figura 5.13: Arquitetura do Servidor da Caixa de Sensores

Capítulo 6

Implementação

Após definir os requisitos, casos de uso e delinear a arquitetura geral do sistema a implementar, é fundamental explicar o que foi implementado em cada um dos blocos previamente definidos na secção 5.1 em termos de *hardware* e *software*, Caixa de Sensores, uMDC, *Smartphone* e Servidor da Caixa de Sensores.

6.1 Caixa de sensores

A Caixa de Sensores é o bloco fundamental do sistema e o único onde foi implementado *hardware* e *software*. Nesta secção são apresentados os principais blocos do *hardware* e fluxogramas do *software* implementado.

6.1.1 *Hardware*

Nesta sub-secção vão ser apresentados os principais blocos de *hardware* desenhados que permitem cumprir os requisitos do sistema que constituem a PCB final da Caixa de Sensores.

Na figura 2 está representado o esquemático simplificado utilizado para a gestão de carregamento (esquemático completo demonstrado no Anexo B) onde se pode ver as decisões tomadas nesta implementação, como por exemplo:

- Pino PSEL com o valor lógico 1 que define a entrada DC como prioritária em relação ao USB, ou seja, se os dois estiverem ligados utiliza a entrada DC;
- Pino ISET1 com a resistência de 4700 Ω que define uma corrente de carregamento de 225 mA;
- Pino ISET2 com o valor lógico 1 que define uma corrente fornecida pela entrada USB de 500 mV;
- Pino DPPM com uma resistência 37 400 Ω que de acordo com a equação 6.1 define a tensão de DPPM. Os valores de $I_{(DPPM)}$ e SF estão tabelados no *datasheet* do IC;
- Criação de um ponto para a monitorização do estado atual da bateria através do microcontrolador (*Label BAT SENSE*).

$$V_{(DPPM-REG)} = I_{(DPPM)} * R_{(DPPM)} * SF \tag{6.1}$$

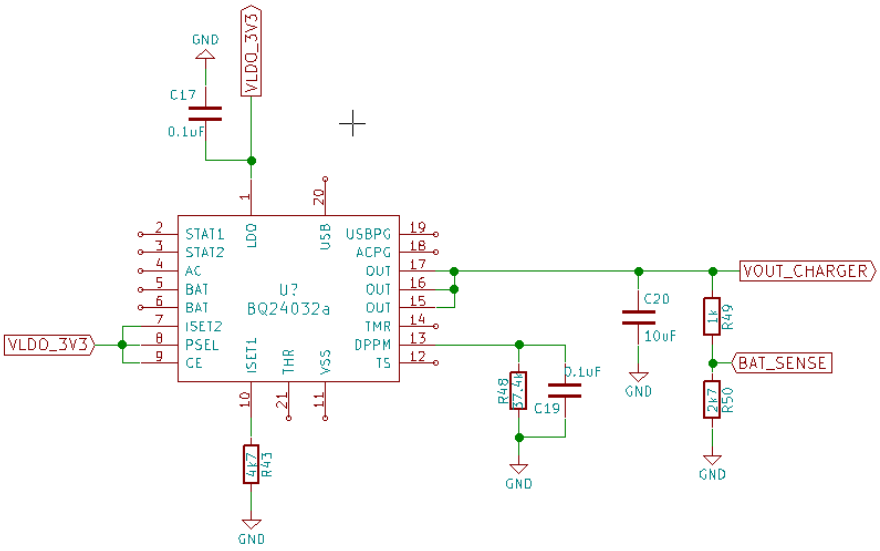


Figura 6.1: Esquemático simplificado do sistema de carregamento da Caixa de Sensores

Em relação ao sistema de proteção com base no IC BQ2970 [65] descrito na secção 5.2.2, o mesmo foi inserido no esquemático entre a bateria e o BQ24032A e utilizou-se a montagem típica descrita no *datasheet*.

Nos conversores DC/DC para 3.3 V e 5 V [66] também se utilizou a montagem típica descrita no *datasheet*, tendo em conta que o conversor para 3.3 V tem de estar sempre ligado pois é o nível de alimentação do microcontrolador. O *enable* deste nível de tensão para outros periféricos é feito pelo IC TPS27081A [67] descrito na secção 5.2.4, que é controlado por um pino do microcontrolador.
O conversor DC/DC para 5 V tem o pino de habilitar/desabilitar ligado diretamente ao microcontrolador pois não existe nenhum periférico crítico ligado a este nível de tensão.

Para o desenho do esquemático do microcontrolador representado no Anexo

B seguiu-se a *datasheet* disponibilizada pelo fabricante [62] em termos de condensadores de entrada, cristais e antena BLE para garantir o correto funcionamento do mesmo. Disponibilizou-se uma UART, um barramento I2C, dois canais ADC, um barramento SPI com possibilidade de ligação de 2 dispositivos, um pino para monitorizar o nível da bateria, um pino para fazer o *enable/disable* da alimentação sensores, também se adicionou um LED de *debug* e o interface para programação do microcontrolador.

Nos barramentos de comunicação para fazer o interface com cada um dos possíveis sensores desenhou-se *level shifters* bi-direcionais para que os sensores a utilizar não estejam limitados ao nível de alimentação do microcontrolador. Na figura 6.2 está representado um desses casos, nomeadamente o do barramento I2C, em que como nos outros barramentos, é possível escolher entre o nível de tensão 3.3 V e 5 V para a alimentação do sensor em questão. Recorrendo ao exemplo demonstrado na figura 6.2 para explicar princípio de funcionamento destes *level shifters*, o Utilizador escolhe através do *Jumper* JP1 o nível de alimentação do sensor (3.3 V ou 5 V), o que faz com que no caso da linha *Serial Data* (SDA) o MOSFET Q4 no lado do sensor (*Drain* - pino 3) tenha uma resistência de *pull-up* R12 ao nível de alimentação escolhido. Do lado do microcontrolador (*Source* - pino 2) a resistência de *pull-up* R8 está sempre ao nível de alimentação do mesmo (3.3 V), a *Gate* do MOSFET Q4 também está sempre a este nível. Assim sendo, quando o microcontrolador tentar enviar o nível lógico 1 o MOSFET Q4 não conduz e a resistência de *pull-up* R12 faz com que o sensor receba um 1; Ao enviar o nível lógico 0 o MOSFET Q4 conduz e o sensor recebe o nível lógico 0. No sentido inverso, quando o sensor enviar o nível lógico 1, o MOSFET Q4 não conduz e a a resistência de *pull-up* R8 faz com que o microcontrolador receba um 1; Para enviar um 0 o MOSFET Q4 conduz pelo diodo e faz com que o microcontrolador receba um 0.

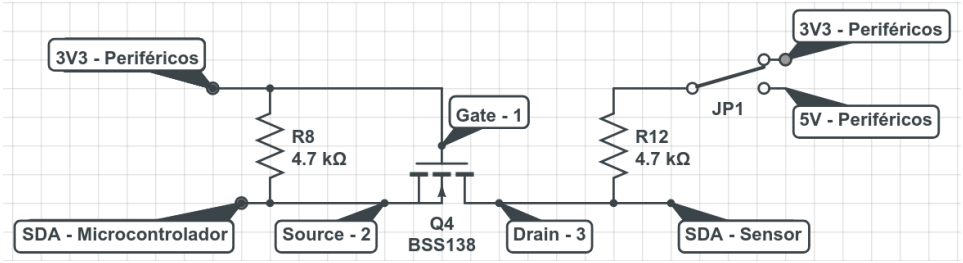


Figura 6.2: Esquemático dos *Level Shifters* entre microcontrolador e os barramentos de comunicação

Em relação ao desenho da PCB representada nas figuras 6.3 e 6.4 seguiu-se a recomendação dos *layouts* descritos nos *datasheets* dos componentes considerados essenciais para o bom funcionamento do sistema, nomeadamente o IC responsável pelo carregamento da bateria, o microcontrolador, os dois conversores DC/DC. Em relação à antena Bluetooth desenhou-se a pista de modo a ter uma impedância de 50 Ω e que a antena esteja posicionada numa das extremidades da PCB sem nenhum componente à sua volta, só plano de massa de forma a potenciar a sua performance[68], [69], [70]. No lado superior da PCB estão situados os conectores para ligar os sensores ao tipo de barramento de comunicação pretendido, os *headers* situados abaixo de cada conector servem para selecionar o nível da tensão da alimentação do mesmo. No lado esquerdo temos as entradas de alimentação, nomeadamente, USB, a entrada DC até 16V, um conector auxiliar para em caso de algum problema fazer um *bypass* ao IC da gestão de carregamento da bateria e alimentar diretamente os conversores DC/DC, e por fim, um conector para ligar a bateria. Ao longo da PCB é possível visualizar alguns *headers* auxiliares, como por exemplo, JP8, JP9, JP10 que foram desenhados com o intuito de isolar cada um dos blocos principais da PCB para na fase de assemblagem e de testes de *hardware* ser possível verificar o funcionamento de cada bloco de forma isolada. Esta PCB foi desenhada com as dimensões necessárias para ser co-

locada numa caixa pequena e compacta previamente escolhida de modo a facilitar a sua instalação.

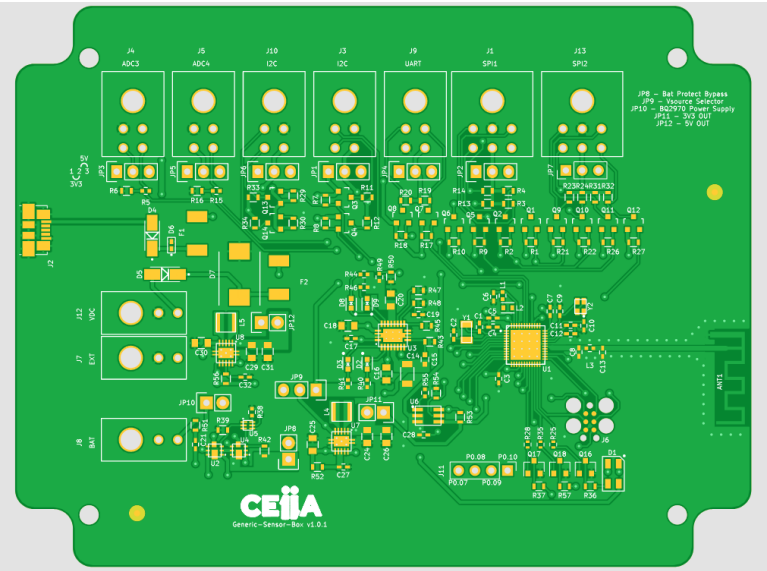


Figura 6.3: Desenho da PCB da Caixa de Sensores - Lado Superior

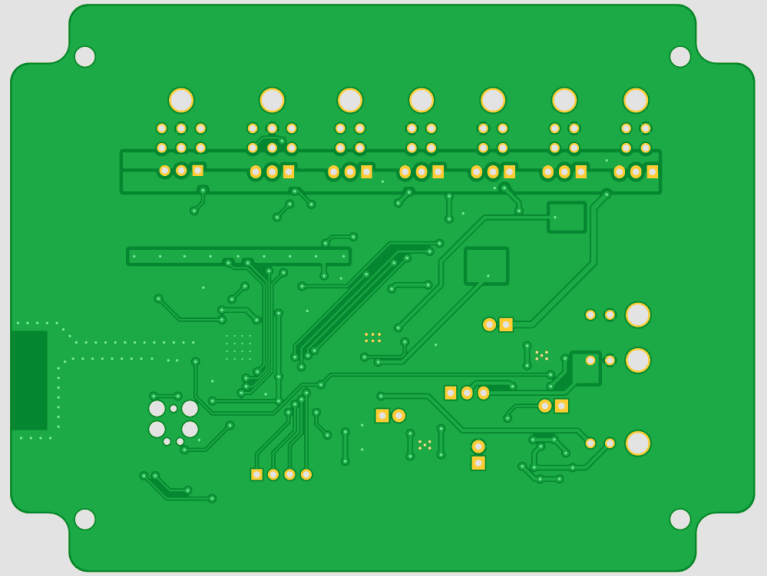


Figura 6.4: Desenho da PCB da Caixa de Sensores - Lado Inferior

Na figura 6.5 está representada a Caixa de Sensores na sua versão final

assemblada com todos os periféricos posicionados.

Na figura 6.6 pretende-se demonstrar os furos realizados na Caixa de Sensores para permitir a sua interação com o meio que a rodeia, permite a circulação de ar bem como a sua ligação ao painel solar se necessário.

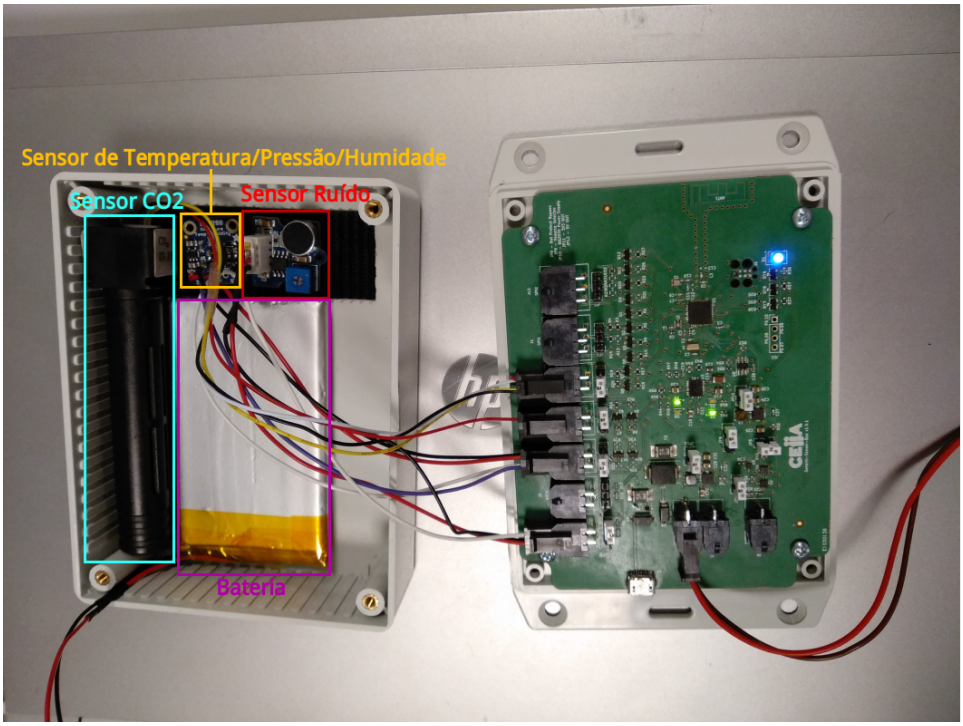


Figura 6.5: Caixa de sensores completa

Figura 6.6: Caixa de sensores furos para entrada de ar

6.1.2 *Software*

Nesta sub-secção é apresentada de forma detalhada o *software* desenvolvido para a Caixa de Sensores de modo a cumprir com a arquitetura delineada na sub-secção 5.2.6. Na figura 6.8 está representado um fluxograma geral do ciclo de funcionamento da Caixa de Sensores.

Na fase de inicialização (representada no fluxograma 6.8) a Caixa de Sensores ativa a alimentação dos sensores e configura-os, inicializa a *stack* BLE da Nordic e configura o dispositivo para ter o papel de Periférico na camada GAP e consequentemente ser o Escravo de uma eventual conexão. Também é inicializado o serviço BLE da Caixa de Sensores que vai conter as 18 características, três por cada tipo de dados a reportar (valor máximo, médio e mínimo), uma para o número de série da Caixa de Sensores, outra para o tempo que passou desde que os dados foram recolhidos e por fim, uma para questões de escrita, tais como, configuração do ciclo periódico de recolha/envio dos dados, o número de leituras de cada um dos sensores e para a inserção da chave de segurança. Antes de entrar no ciclo principal, obtém o número de série do microcontrolador e configura com valores por defeito a periodicidade dos ciclos de leitura, janela de *advertisement* e número de leituras de cada sensor.

Com o *software* desenvolvido é possível configurar parâmetros da Caixa de Sensores via BLE com o objetivo de reconfigurar a Caixa de Sensores sem re-programar o microcontrolador.

Estas configurações são introduzidas numa característica de escrita disponibilizada pela Caixa de Sensores que permite numa fase inicial validar a conexão através de uma chave de segurança e numa segunda fase após a validação da ligação permite alterar os parâmetros de configuração de acordo com a especificação demonstrada na figura 6.7. O *flow* de inserir a chave de segurança e configurar a Caixa de Sensores está descito na figura 6.9. Os

valores com que se pretende configurar a Caixa de Sensores são inseridos na característica para o efeito, o tempo do ciclo de leitura/*report* dos dados e a janela de *advertisement* são inseridos em segundos, também é inserido o número de leituras de cada sensor por ciclo, o que indiretamente permite calcular de quanto em quanto tempo necessitamos de ler cada sensor dividindo o tempo do ciclo de leitura/*report* pelo número de amostras de cada sensor. Por exemplo, se definirmos o ciclo de leitura/*report* dos dados com 60 segundos, a janela de *Advertising* 15 segundos, 30 leituras do sensor de ruído, 2 leituras do Sensor de Temperatura/Humidade/Pressão e 1 leitura do sensor de CO2 a Caixa de Sensores vai recolher dados de 60 em 60 segundos periodicamente e após esse tempo irá tentar reportar os mesmos durante 15 segundos. Em relação aos sensores, vai efetuar uma leitura do ruído a cada 2 segundos, de Temperatura/Pressão/Humidade a cada 30 segundos e de CO2 a cada 60 segundos antes de reportar os dados.

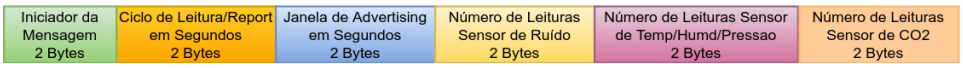


Figura 6.7: Trama de Configuração da Caixa de Sensores

A noção de tempo na Caixa de Sensores é obtida através de um Driver que é responsável por estimar quantos milissegundos passaram desde o arranque do microcontrolador a partir da diferença de tempo conhecida entre dois ticks do microcontrolador. Após a Caixa de Sensores ser configurada e tendo a noção de contagem do tempo em milissegundos é possível determinar quando é necessário realizar cada tarefa.

A Caixa de Sensores com o objetivo de consumir a menor quantidade de energia só liga a alimentação dos sensores o tempo necessário antes de efetuar uma leitura, este processo está representado na figura 6.8 como Preparação dos sensores. No caso particular do sensor de CO2, o mesmo é ligado

aproximadamente 30 segundos antes de efetuar a leitura devido ao tempo de pré-aquecimento do sensor.
O processo de leitura dos sensores descrito na figura 6.10 efetua a leitura de cada sensor de acordo com os tempos configurados.

O *core* da Caixa de Sensores é o ciclo em que está constantemente a verificar se é necessário efetuar a leitura de algum sensor, se é tempo de calcular a estatística do intervalo de leitura para reportar os dados, ou seja, os valores máximos, médios e mínimos lidos em cada intervalo, se deve começar a fazer *Advertising* para tentar estabelecer uma conexão com o uMDC ou com o *Smartphone* e assim respeitar a sua configuração como descrito na figura 6.8.

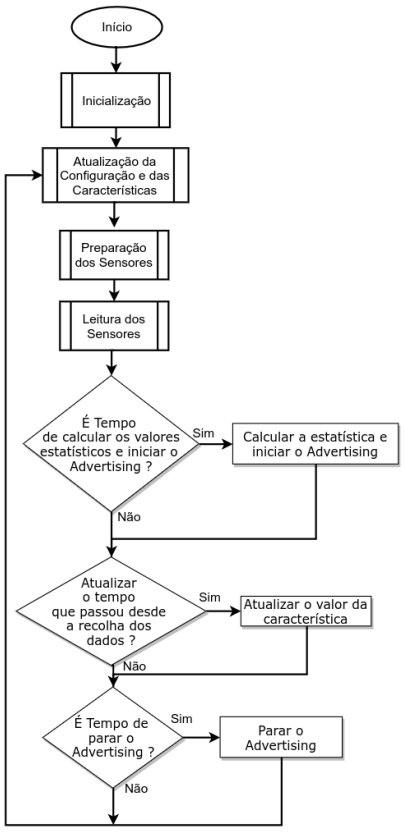


Figura 6.8: Fluxograma de *Software* da Caixa de Sensores - Ciclo Principal

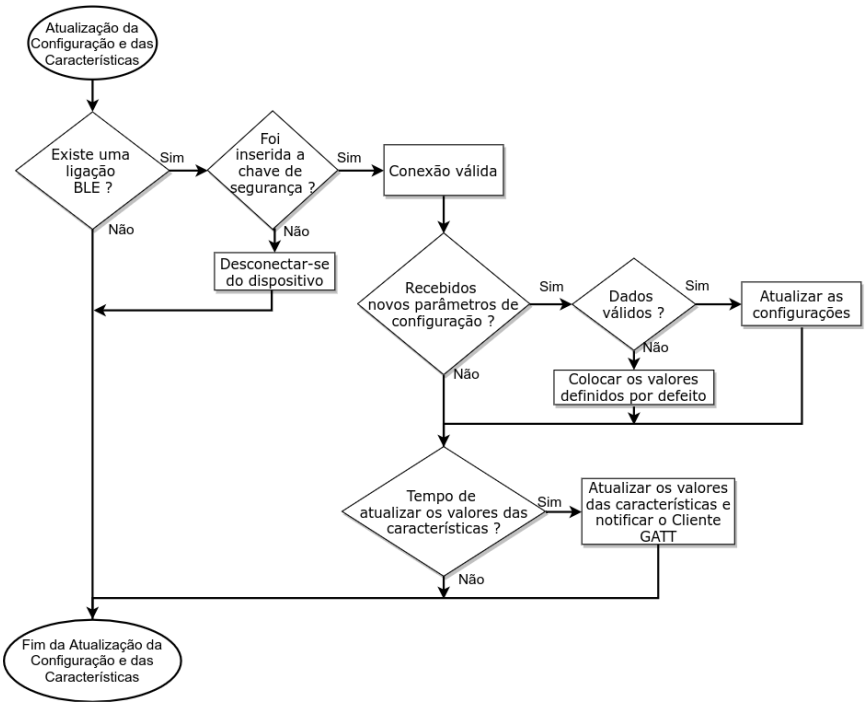


Figura 6.9: Fluxograma de *Software* da Caixa de Sensores - Atualização da Configuração e das Características

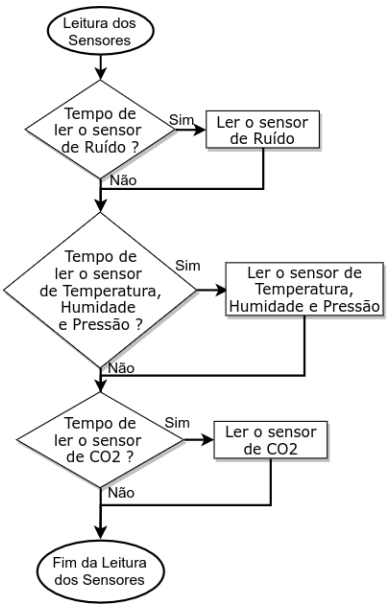


Figura 6.10: Fluxograma de *Software* da Caixa de Sensores - Leitura dos Sensores

6.2 *Micro Mobility Device Controller - Software*

Nesta secção pretende-se descrever o *software* desenvolvido para o uMDC. Na figura 6.11 está representado um fluxograma para demonstrar o funcionamento do *software* do uMDC de acordo com a arquitetura demonstrada na secção 5.3. Neste caso, o uMDC tem como principal objetivo servir de ponte entre a Caixa de Sensores e o resto do sistema visto que a mesma não tem um modo de comunicação direto com o sistema devido às restrições energéticas.

Na fase de inicialização representada na figura 6.11 o uMDC inicializa a *stack* BLE da Nordic e configura o dispositivo para ter o papel de Central na camada GAP e consequentemente o papel de Mestre após estabelecer ligação, também define as configurações da ligação, como por exemplo, o intervalo de conexão.

No ciclo principal o uMDC, na tarefa designada de Processo de Conexão e Leitura/Notificação das Características detalhada na figura 6.12, tem de ser capaz de estar periodicamente à "escuta" de pacotes de *advertisement* provenientes de Caixas de Sensores para tentar estabelecer uma ligação, após estabelecer a ligação o uMDC deve ser capaz de iniciar o processo de Conexão, ou seja, a descoberta da tabela de atributos do Servidor GATT (Caixa de Sensores), reconhecer através do UUID quais atributos existentes na tabela (Serviços e Características), guardar as posições de cada um dos atributos requeridos da tabela e inserir a chave de Segurança na Característica para o efeito, só assim o uMDC estabelece uma conexão com sucesso à Caixa de Sensores.
Após estabelecer conexão o dispositivo deve ler os dados disponibilizados

pela Caixa de Sensores em forma de características, após obter os dados da Caixa de Sensores a primeira vez, deve indicar que pretende receber futuras alterações (notificações) dos valores dessas mesmas características e manter a ligação o máximo de tempo possível de modo a receber possíveis atualizações.

A noção de tempo no uMDC pode ser obtida de duas formas: utilizando os *ticks* do microcontrolador como é feito na Caixa de Sensores e através do módulo de GSM.

De acordo com a descrição do restante ciclo principal representado na figura 6.11 o uMDC deve ser responsável por gerir todo o processo de receção dos dados da Caixa de Sensores e envio das mensagens para o *Broker* MQTT através de um protocolo proprietário da empresa.
Periodicamente deve verificar se é para guardar as mensagens enviadas pela Caixa de Sensores e adicionar uma referência temporal às mesmas (data e hora).
A associação dos dados recolhidos pela Caixa de sensores com as coordenadas GPS é feita no Servidor da Caixa de Sensores que está a analisar as mensagens reportadas pelo uMDC.
Como uma das principais funcionalidades do uMDC é ser um dispositivo de *tracking* de veículos o mesmo já reportava para o *Broker* as coordenadas GPS e pretendeu-se manter a compatibilidade com o que já estava implementado.

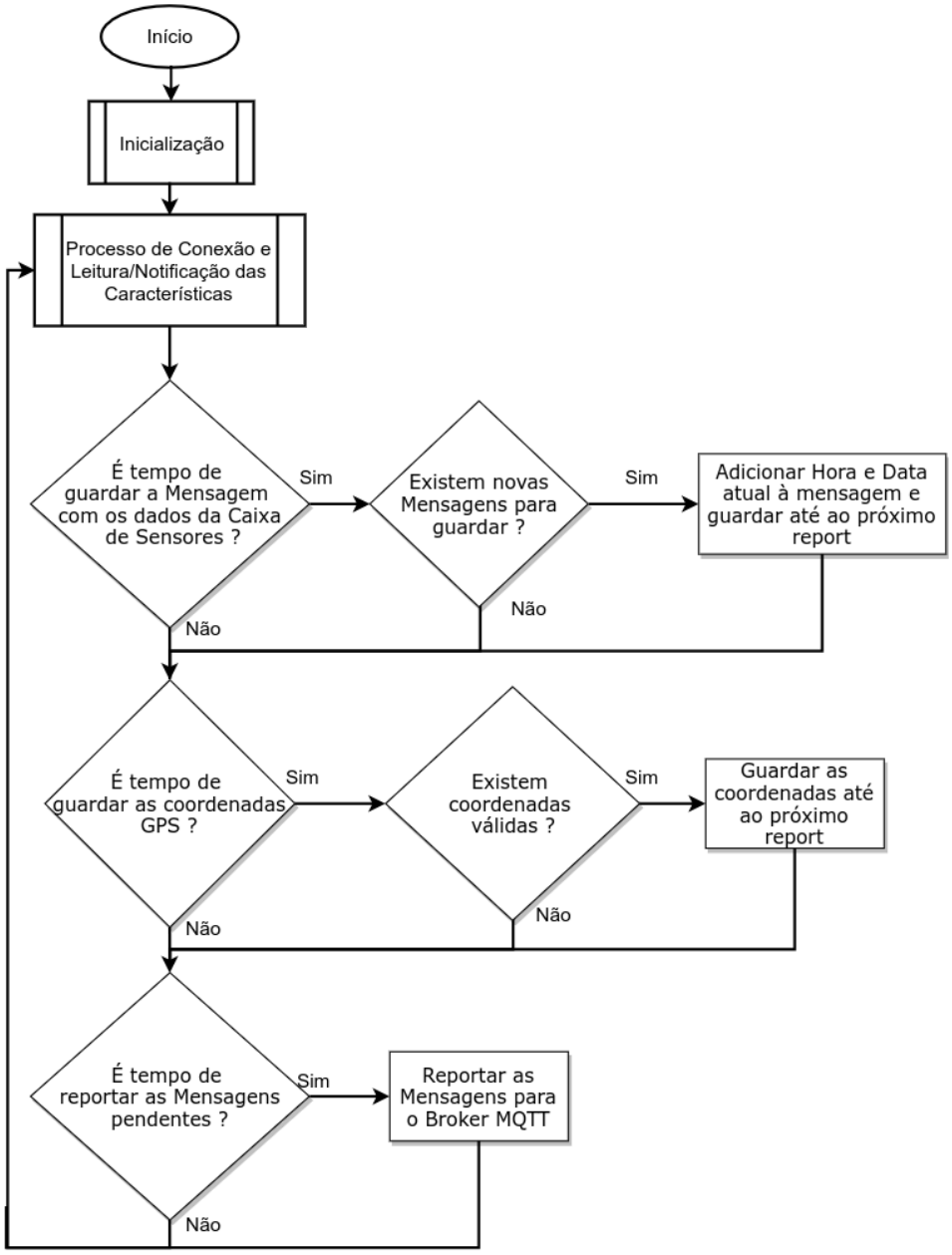


Figura 6.11: Fluxograma de *Software* do uMDC - Ciclo Principal

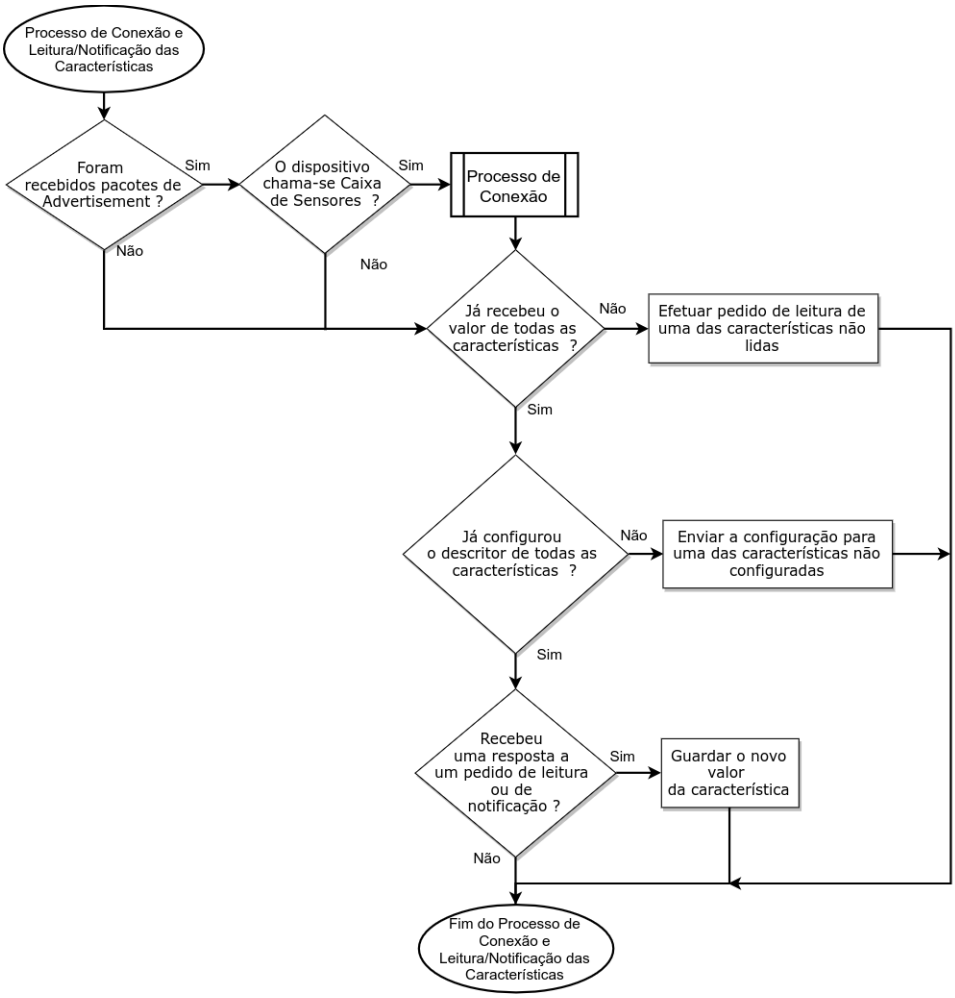


Figura 6.12: Fluxograma de *Software* do uMDC - Processo de Conexão e Leitura/Notificação das Características

6.3 *Smartphone - Software*

Nesta secção é apresentado o *software* desenvolvido para o *Smartphone*. Como na arquitetura global do sistema implementado o *Smartphone* tem o mesmo papel do uMDC seguiu-se o princípio de funcionamento implementado no uMDC para que tanto do lado da Caixa de Sensores como do lado do Servidor que está a analisar as mensagens enviadas pelos dois haja uma

implementação única independentemente do dispositivo que está conectado à Caixa de Sensores e vai reportar os dados.
Visto que os *Smartphones* são dispositivos com grande capacidade de armazenamento, ao contrário do uMDC, não precisam de estar constantemente conectados ao *Broker* MQTT para reportar quase instantaneamente os dados, podem armazenar os mesmos durante algum tempo e periodicamente verificar se existe uma ligação ao *Broker* MQTT, se não existir, estabelecê-la e reportar todos os dados armazenados. Para cumprir a compatibilidade com o uMDC as mensagens GPS são reportadas em separado das mensagens das Caixas de Sensores.

Na fase de inicialização descrita na figura 6.14 o *Smartphone* garante que tem permissões para interagir com os módulos BLE, GPS, Interface de Rede e inicializa os objetos para interagir com cada um deles. Também disponibiliza um botão para iniciar o processo da procura por Caixas de Sensores. Quando o botão de início de processo demonstrado no *layout* da aplicação desenvolvida (cf. figura 6.13) é pressionado, dá-se início a um *Timer* que de 10 em 10 segundos começa/para o *scan* por dispositivos nomeados de Caixa de Sensores.

Figura 6.13: Layout da Aplicação *Android* desenvolvida para o *Smartphone*

Quando o *Smartphone* encontra um dispositivo nomeado Caixa de Sensores inicia-se o processo de Conexão e Leitura/Notificação das Características representado na figura 6.15, o primeiro passo é a descoberta da tabela de atributos do Servidor GATT, reconhece através do UUID quais atributos pretendidos, insere a chave de segurança na característica para o efeito e efetua um pedido de leitura do primeiro atributo com permissões para tal.
Sempre que o *Smartphone* recebe um evento resultante de um pedido de leitura ou notificação, verifica se o novo valor recebido pertence a uma característica pretendida e guarda-o, se for o valor referente à última característica que estamos à espera cria-se um objeto com o conjunto de mensagens provenientes da Caixa de Sensores e adiciona-se data/hora e coordenadas GPS ao conjunto de dados.
Sempre que o uMDC estabelece uma nova conexão com uma Caixa de Sensores, tenta obter o valor de todas as características pretendidas, após este passo configura-se o descritor de cada uma dessas características para receber eventuais atualizações das mesmas. O procedimento de leitura do valor das características e configuração do descritor é efetuado numa lógica de fazer um pedido e esperar pelo evento de resposta. Como se configurou o descritor das características para receber notificações, sempre que a Caixa de Sensores atualiza o valor de uma das características pretendidas despoleta o processo de atualização do valor da mesma.
No ciclo principal, descrito na figura 6.14, o *Smartphone* além do processo de Leitura/Notificação das Características também atualiza as suas coordenadas GPS sempre que existir movimento, verifica se é tempo de se conectar ao Broker, se existirem mensagens pendentes para serem publicadas e não existir uma conexão válida a mesma é estabelecida, também inicia-se um *Timer* para de N em N segundos publicar todas as mensagens pendentes .

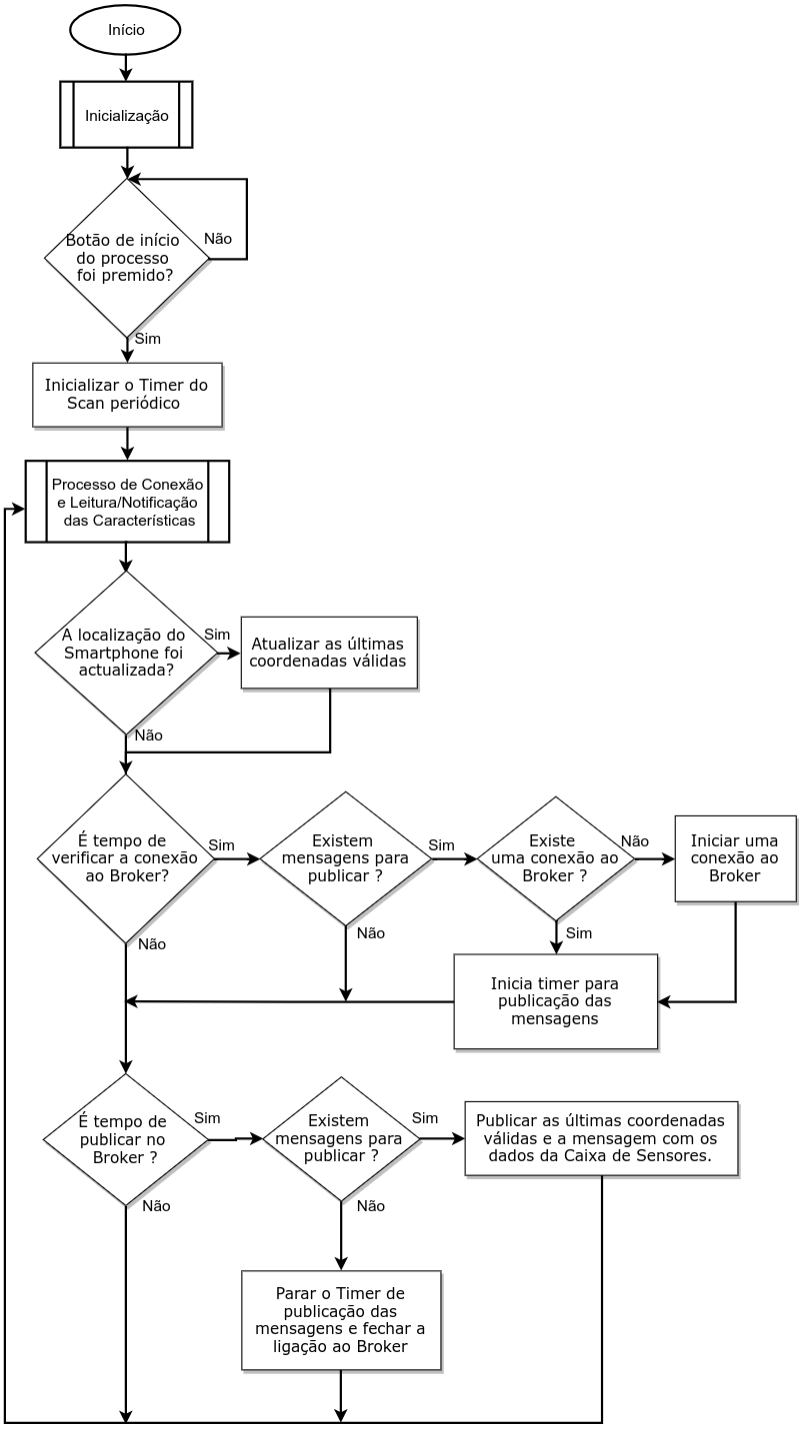


Figura 6.14: Fluxograma de *Software* do *Smartphone* - Ciclo Principal

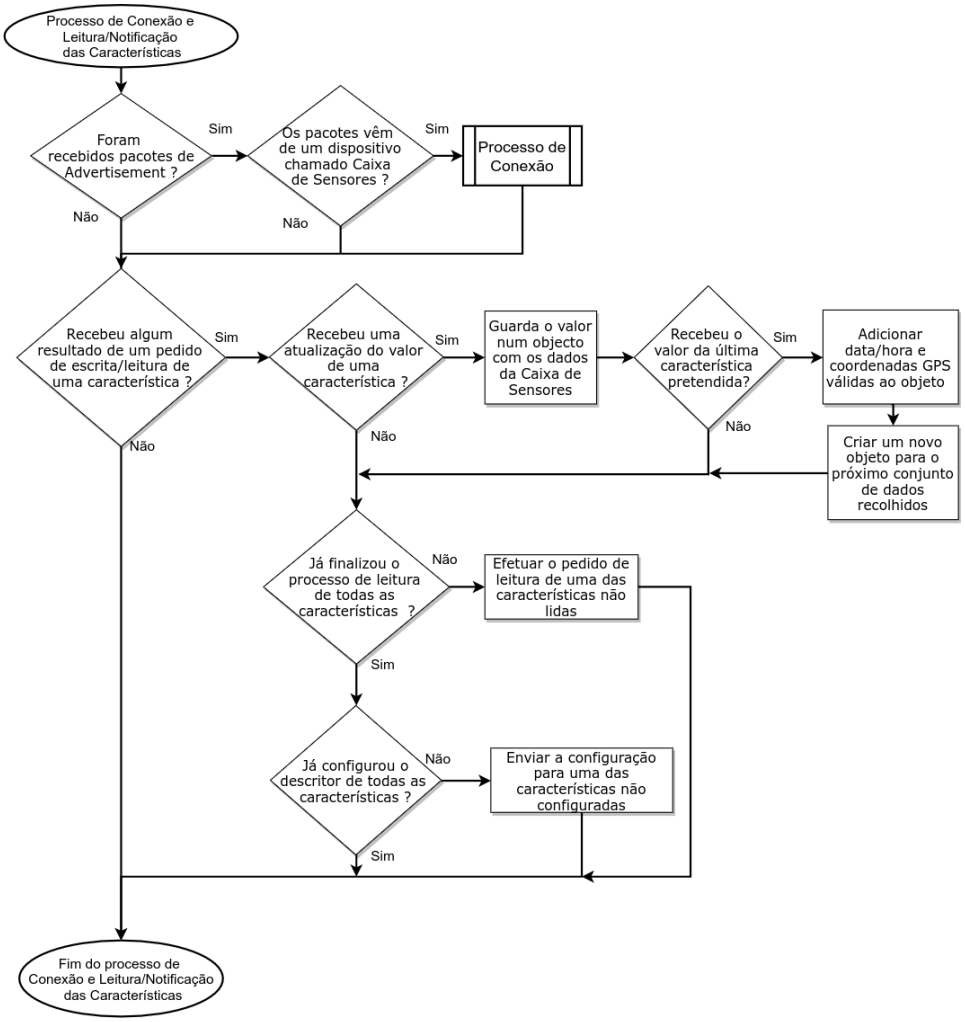


Figura 6.15: Fluxograma de *Software* do *Smartphone* - Processo de Conexão e Leitura/Notificação das Características

6.4 Servidor da Caixa de Sensores - *Software*

Nesta secção pretende-se descrever o *software* desenvolvido para o Servidor da Caixa de Sensores, responsável por analisar as mensagens reportadas pelo uMDC e pelo *Smartphone*. Na figura 6.16 está representado o fluxograma de funcionamento desta camada. Este Servidor permite testar e validar todo o funcionamento do sistema de forma independente do Servidor da empresa.

Na fase de inicialização é criando um Cliente MQTT e é estabelecida uma conexão ao Broker MQTT, subscreve-se aos tópicos pretendidos e inicia-se um Interface gráfico (Servidor HTTP) para visualizar os dados recebidos.
No ciclo principal o Cliente MQTT está constantemente a analisar o tipo de mensagens recebidas que pode ser relativas a coordenadas GPS, dados das Caixas de Sensores Fixas ou Móveis.
Se forem mensagens GPS, de acordo com o fluxograma da figura 6.17, é criado um *Marker* com a seguinte informação: ID incremental, número de série do dispositivo que reportou as coordenadas, as coordenadas reportadas e data/hora em que as coordenadas foram recolhidas. Este *Marker* é adicionado ao ficheiro de Caixas de Sensores Móveis, se o ficheiro não existir é criado.
Se forem recebidas mensagens com os dados das Caixa de Sensores é verificado se são provenientes de Caixas de Sensores Fixas ou Móveis de acordo com uma lista existente com as Caixas de Sensores Fixas e as coordenadas da sua instalação.
O *flow* das mensagens das Caixas de Sensores Fixas, descrito na figura 6.18, agrupa as mensagens por número de série da Caixa de Sensores que recolheu os dados, se o ficheiro das Caixas de Sensores Fixas ou o *Marker* da Caixa que recebemos a mensagem não existir, são criados. O *Marker* criado contém a seguinte informação: ID incremental, número de série e coordenadas da Caixa de Sensores.
Em relação às mensagens das Caixas de Sensores Móveis, o *flow* destas mensagens está representado na figura 6.19, analisa-se o ficheiro que numa fase inicial só contém coordenadas GPS e agrupa-se os dados recebidos às coordenadas GPS mais recentes reportadas pelo mesmo dispositivo (uMDC ou *Smartphone*).

Foi criado um Interface Gráfico onde o Utilizador pode filtrar os dados apresentados por Caixas de Sensores Fixas ou Móveis e por tipo de dados recolhi-

dos, neste caso, os dados recolhidos foram Temperatura, Pressão, Humidade, Concentração de CO2 e nível de ruído/som.
Na fase inicial o Interface Gráfico inicializa o Mapa e os botões para filtrar por Caixas de Sensores Fixas ou Móveis e por um tipo de dados (e.g. Temperatura, Pressão, Humidade, CO2 e Ruído). Por defeito é selecionada a configuração que permite visualizar os dados GPS das Caixas de Sensores Fixas.
Se o tipo de dados selecionado for GPS o interface deve desenhar os *Markers* nas respetivas coordenadas, atualizar a legenda do Mapa e adicionar o último conjunto de dados reportado naquela coordenada ao *Popup*.
Se o tipo de dados selecionado for Temperatura, Humidade, Pressão, CO2 ou Ruído filtra-se os dados pela média do tipo de dados em questão e enquadra-se nos *thresholds* definidos. Por fim, atualiza-se o interface como é feito para as coordenadas GPS.

Também foi desenvolvido um pequeno *script* em *Python* para apresentar um gráfico que demonstra a evolução ao longo do tempo de um tipo de dados de uma Caixa de Sensores a selecionar pelo Utilizador. Este *script* recebe como parâmetros se pretendemos ver os dados de Caixas de Sensores Fixas ou Móveis, o número de série da Caixa de Sensores Fixa ou o número de série do dispositivo móvel (uMDC/*Smartphone*) que reportou os dados e o tipo de dados a visualizar. Com estes dados o *script* simplesmente percorre o ficheiro pretendido, procura o número de série inserido e demonstra o tipo de dados pretendido ao longo do tempo.

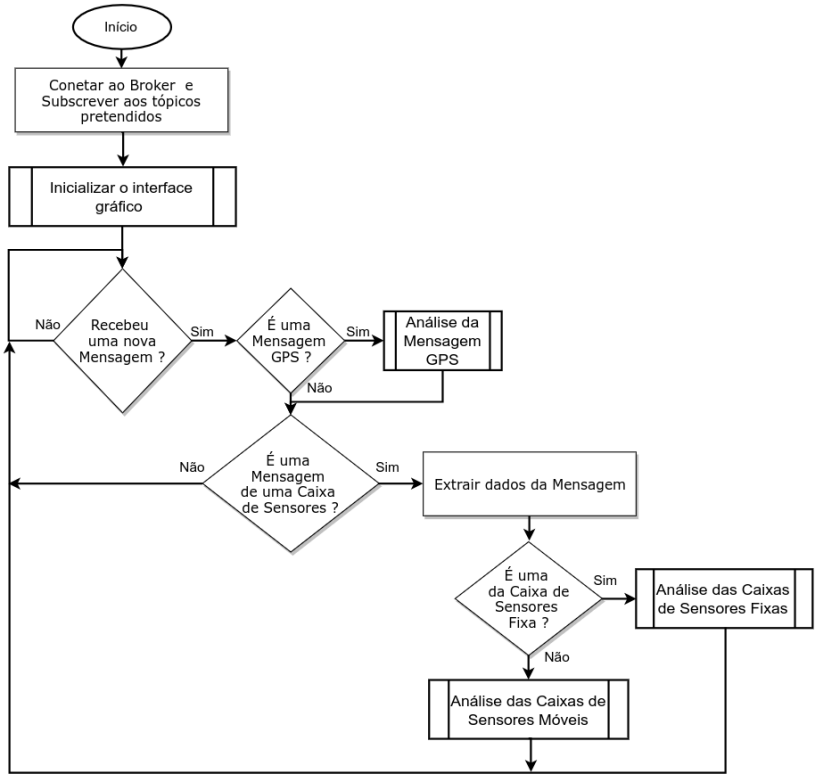


Figura 6.16: Fluxograma de *Software* do Servidor da Caixa de Sensores - Ciclo Principal

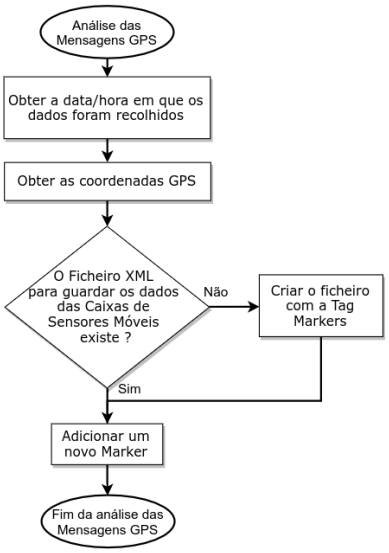


Figura 6.17: Fluxograma de *Software* do Servidor da Caixa de Sensores - Análise das mensagens GPS

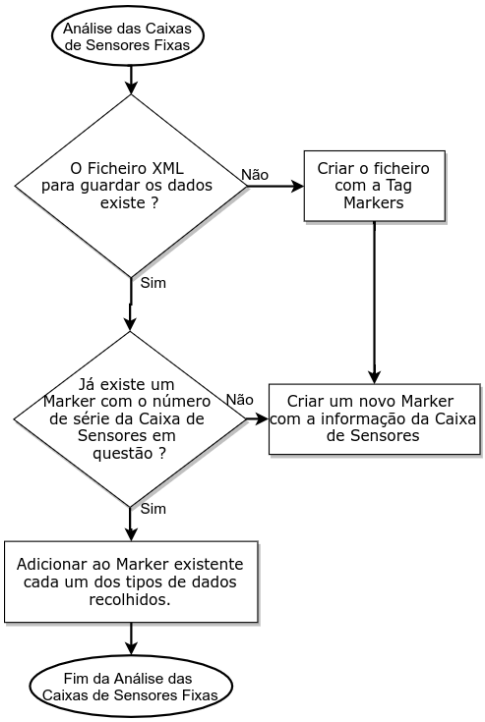


Figura 6.18: Fluxograma de *Software* do Servidor da Caixa de Sensores - Análise das mensagens das Caixas de Sensores Fixas

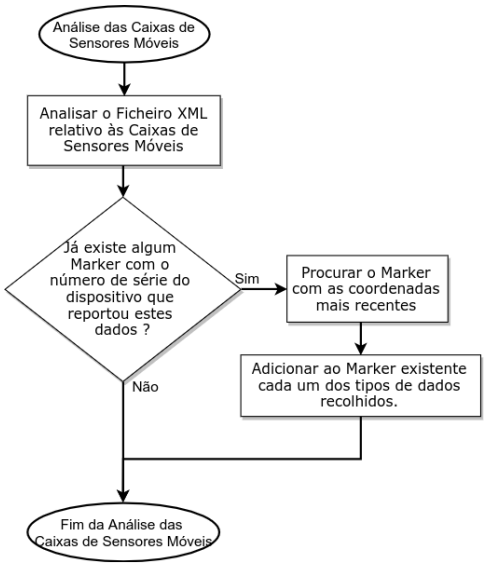


Figura 6.19: Fluxograma de *Software* do Servidor da Caixa de Sensores - Análise das mensagens das Caixas de Sensores Móveis

This page intentionally left blank.

Capítulo 7

Testes e Resultados

Neste capítulo pretende-se apresentar e analisar alguns dos resultados mais relevantes dos testes realizados durante o projeto, o que facilita a compreensão do sistema desenvolvido. Foram realizados testes para demonstrar o funcionamento global do sistema, tais como: a Caixa de Sensores a funcionar como dispositivo Móvel e Fixo, distância e velocidade de comunicação BLE entre a Caixa de Sensores e os *gateways*, produção de energia do painel solar e consumos da Caixa de sensores de modo a estudar o impacto do consumo dos sensores e da comunicação BLE.

7.1 Caixa de Sensores Móvel

Neste teste pretendeu-se estudar o comportamento da Caixa de Sensores a funcionar como um dispositivo móvel. Instalou-se uma Caixa de Sensores numa bicicleta elétrica que contém um uMDC representada na figura 7.1 e realizaram-se alguns percursos de maneira a analisar o comportamento da Caixa de Sensores em termos de comunicação com o uMDC e de recolha dos dados. Neste caso a Caixa de Sensores é alimentada pela bateria do veículo.

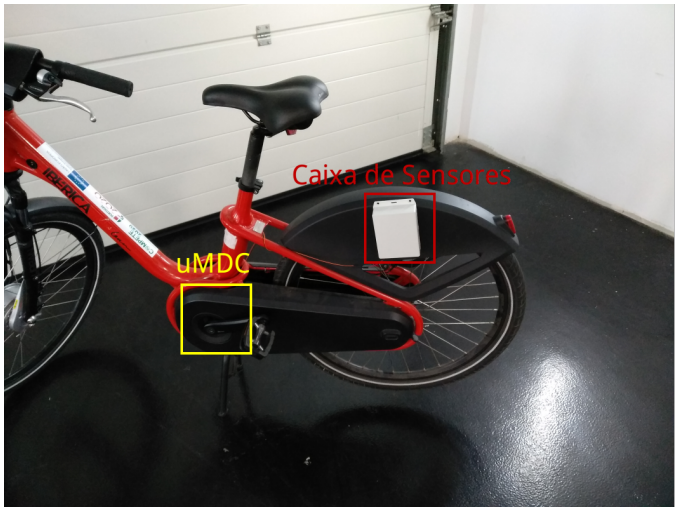


Figura 7.1: Caixa de Sensores instalada na bicicleta elétrica

Configurou-se a Caixa de Sensores da seguinte maneira:

- Reportar os dados de 30 em 30 segundos;
- Janela de *Advertising* de 15 segundos;
- Recolher em cada ciclo de *report*:
  - 15 amostras de ruído;
  - 1 amostra de Temperatura, Humidade e Pressão;
  - 1 amostra da Concentração de CO2.

Nas figuras 7.2 e 7.3 estão representados os resultados, através do Interface Gráfico, de Temperatura e Ruído recolhidos ao longo de percurso realizado, com início na estrela e sentido assinalados na figura 7.2, os marcadores translúcidos representam as coordenadas GPS reportadas pelo uMDC ao longo do trajeto enquanto que os marcadores coloridos respeitam a legenda apresentada nas respetivas figuras. No Anexo C estão representados os restantes tipos de dados recolhidos durante o percurso.

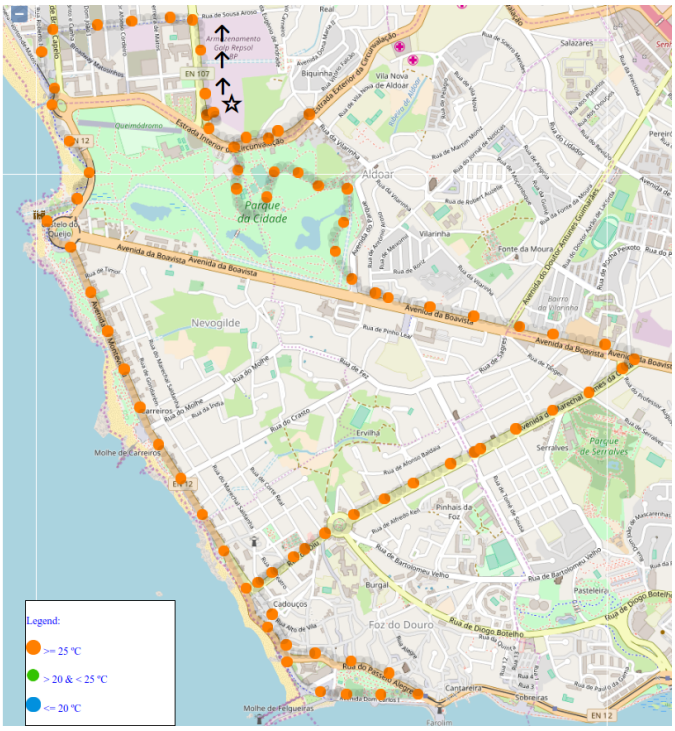


Figura 7.2: Caixa de Sensores Móvel: Dados de Temperatura - Cenário 1

Figura 7.3: Caixa de Sensores Móvel: Dados de Ruído - Cenário 1

Na figura 7.4 pretende-se demonstrar o efeito de clicar sobre alguns dos pontos apresentados no Mapa, o que permite a demonstração dos valores e quando esses dados foram recolhidos. Com estes dados é possível visualizar que os dados estão a ser recolhidos/enviados pela Caixa de Sensores sensivelmente de 30 em 30 segundos como era suposto e que foram reportados como era de esperar pelo uMDC. O erro de 1 segundo é bastante aceitável tendo em conta que a Caixa de Sensores não tem referência temporal e que os dados só são referenciados em relação ao tempo quando chegam ao uMDC. Este percurso é o mesmo realizado na figura 7.2, assim os *popups* representados estão cronologicamente de acordo com o percurso.

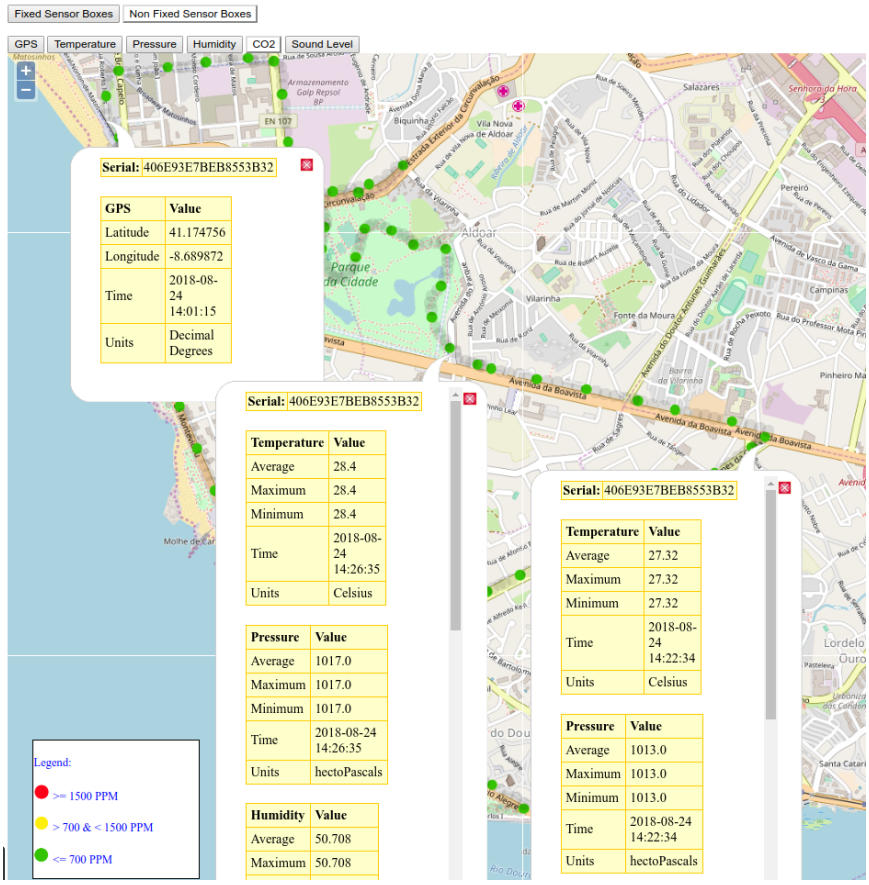


Figura 7.4: Caixa de Sensores Móvel: *Logs* de CO2 com *Popups* - Cenário 1

Na figura 7.5 está representado um extrato de como estão organizados os dados GPS que posteriormente permitem a demonstração realizada no Interface Gráfico.

O uMDC instalado na bicicleta vai reportando os pontos GPS e os dados da Caixa de Sensores referenciados temporalmente. O Cliente MQTT que analisa as mensagens adiciona um Marker por ponto GPS reportado.

```
  </marker>
- <marker id="758" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.164976" Longitude="-8.670537" Time="2018-08-24 14:25:54" Units="Decimal Degrees"/>
  </marker>
- <marker id="759" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.164984" Longitude="-8.670604" Time="2018-08-24 14:26:02" Units="Decimal Degrees"/>
  </marker>
- <marker id="760" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.16504" Longitude="-8.67088" Time="2018-08-24 14:26:08" Units="Decimal Degrees"/>
  </marker>
- <marker id="761" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.165108" Longitude="-8.67117" Time="2018-08-24 14:26:14" Units="Decimal Degrees"/>
  </marker>
- <marker id="762" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.165208" Longitude="-8.671677" Time="2018-08-24 14:26:20" Units="Decimal Degrees"/>
  </marker>
- <marker id="763" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.165484" Longitude="-8.671745" Time="2018-08-24 14:26:26" Units="Decimal Degrees"/>
  </marker>
- <marker id="764" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.16562" Longitude="-8.671998" Time="2018-08-24 14:26:32" Units="Decimal Degrees"/>
```

Figura 7.5: Estrutura dos dados das Caixas de sensores Móveis sem os dados da Caixa de Sensores

Ao receber as mensagens relativas às Caixas de Sensores o Servidor da mesma concatena os dados à coordenada mais próxima em termos temporais reportada pelo dispositivo em questão, ou seja, procura a coordenada com a menor diferença entre o tempo em que a coordenada e os dados da Caixa de Sensores foram recolhidos. Neste caso em concreto como todos as coordenadas GPS são do mesmo uMDC, podem-se ver na figura 7.6 que os dados da Caixa de Sensores foram associados à coordenada mais próxima, em termos temporais.

```
</marker>
- <marker id="758" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.164976" Longitude="-8.670537" Time="2018-08-24 14:25:54" Units="Decimal Degrees"/>
  </marker>
- <marker id="759" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.164984" Longitude="-8.670604" Time="2018-08-24 14:26:02" Units="Decimal Degrees"/>
    <Temperature Average="28.28" Maximum="28.28" Minimum="28.28" Time="2018-08-24 14:26:05" Units="Celsius"/>
    <Pressure Average="1017.0" Maximum="1017.0" Minimum="1017.0" Time="2018-08-24 14:26:05" Units="hectoPascals"/>
    <Humidity Average="51.746" Maximum="51.746" Minimum="51.746" Time="2018-08-24 14:26:05" Units="Percentage"/>
    <CO2 Average="400.0" Maximum="400" Minimum="400" Time="2018-08-24 14:26:05" Units="PPM"/>
    <Sound Average="Normal Noise" Maximum="High Noise" Minimum="Low Noise" Time="2018-08-24 14:26:05" Units="String"/>
  </marker>
- <marker id="760" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.16504" Longitude="-8.67088" Time="2018-08-24 14:26:08" Units="Decimal Degrees"/>
  </marker>
- <marker id="761" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.165108" Longitude="-8.67117" Time="2018-08-24 14:26:14" Units="Decimal Degrees"/>
  </marker>
- <marker id="762" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.165208" Longitude="-8.671677" Time="2018-08-24 14:26:20" Units="Decimal Degrees"/>
  </marker>
- <marker id="763" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.165484" Longitude="-8.671745" Time="2018-08-24 14:26:26" Units="Decimal Degrees"/>
  </marker>
- <marker id="764" serial="406E93E7BEB8553B32">
    <GPS Latitude="41.16562" Longitude="-8.671998" Time="2018-08-24 14:26:32" Units="Decimal Degrees"/>
    <Temperature Average="28.4" Maximum="28.4" Minimum="28.4" Time="2018-08-24 14:26:35" Units="Celsius"/>
    <Pressure Average="1017.0" Maximum="1017.0" Minimum="1017.0" Time="2018-08-24 14:26:35" Units="hectoPascals"/>
    <Humidity Average="50.708" Maximum="50.708" Minimum="50.708" Time="2018-08-24 14:26:35" Units="Percentage"/>
    <CO2 Average="400.0" Maximum="400" Minimum="400" Time="2018-08-24 14:26:35" Units="PPM"/>
    <Sound Average="Normal Noise" Maximum="High Noise" Minimum="Low Noise" Time="2018-08-24 14:26:35" Units="String"/>
```


Figura 7.6: Estrutura dos dados das Caixas de sensores Móveis com os dados da Caixa de Sensores

7.2 Caixa de Sensores Fixa

Neste teste pretendeu-se estudar o comportamento da Caixa de Sensores a funcionar como um dispositivo fixo. Na figura 7.7 pretende-se demonstrar a Caixa de Sensores instalada, neste caso, num posto de carregamento. Esta Caixa de Sensores é alimentada por um painel solar e bateria.

Figura 7.7: Caixa de Sensores Fixa instalada no posto de carregamento

A Caixa de Sensores foi configurada da seguinte maneira:

- Reportar os dados de 300 em 300 segundos;
- Janela de *Advertising* de 300 segundos;
- Recolher em cada ciclo de *report*:
  - 150 amostras de ruído;
  - 1 amostra de Temperatura, Humidade e Pressão;
  - 1 amostra da Concentração de CO2.

Os dados foram recolhidos através de passagens de um uMDC instalado numa bicicleta e passagens a pé com um *Smartphone*.

Na figura 7.8 é possível verificar os últimos dados reportados pela Caixa de Sensores através do Interface Gráfico.

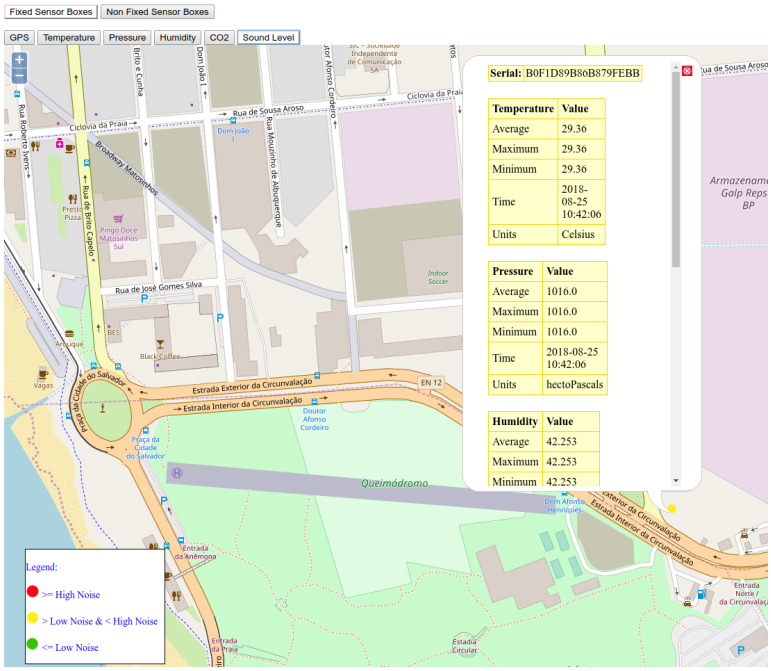


Figura 7.8: *Logs* de Som da Caixa de Sensores Fixa com *Popup*

Na figura 7.9 está representado um extrato de como estão organizado os dados que permitem a demonstração realizada no Interface Gráfico. Os dados são sempre adicionados ao marcador do número de série da Caixa de Sensores em questão. As coordenadas GPS apresentadas são as inseridas na lista de Caixas de Sensores Fixas. Como o *report* dos dados está de 300 em 300 segundos podemos verificar que os dados repetidos (conjuntos 2-3 e 4-5) significa que em menos de 300 segundos passou um uMDC e um *Smartphone* próximo da Caixa de Sensores, recolheram e reportaram os dados. Podemos também verificar que o último conjunto de dados (6) surge 600 segundos depois do anterior o que significa que existiu um intervalo de dados que não foi recolhido por nenhum uMDC nem *Smartphone*, ou seja, nenhum deles passou perto da Caixa de Sensores.

Figura 7.9: Estrutura dos dados das Caixas de sensores Fixas

7.3 Gráfico para Visualização dos dados ao longo do Tempo

Na figura 7.10 é apresentado um gráfico obtido através do *script* descrito na secção 6.4 em que é possível visualizar a variação de um tipo de dados de uma Caixa de Sensores à escolha. Neste caso em específico é possível visualizar a variação da Temperatura ao longo do Tempo de um conjunto de dados reportado por uma Caixa de Sensores Móvel em cenário de testes.

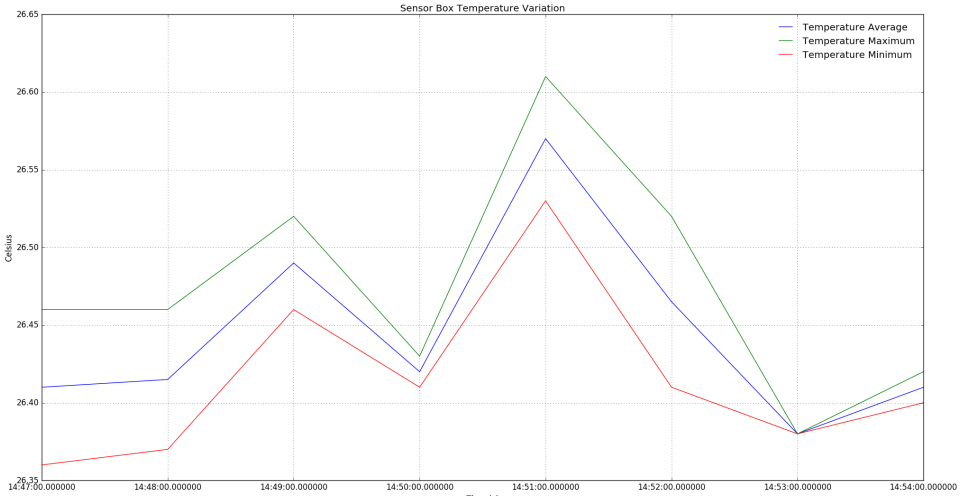


Figura 7.10: Gráfico dos *Logs* das Caixas de Sensores

7.4 Testes de Comunicação *Bluetooth Low Energy* - Distância e Velocidade

Foram feitos testes de comunicação BLE de modo a aferir se a solução da Caixa de Sensores Fixa é viável, ou seja, se é possível passarmos com um veículo e recolher os dados. Numa primeira fase fez-se testes de distância, ou seja, a que distância máxima seria possível conectarmos a uma Caixa de Sensores estando estático com um *Smartphone*. Na figura 7.11 está representado

o cenário em que o teste foi realizado em que a vermelho está representada a Caixa de Sensores. A distância máxima em que foi possível estabelecer uma conexão com a Caixa de Sensores instalada no posto de carregamento foi de aproximadamente 37 metros. O teste foi feito com a aplicação implementada neste projeto com o *Smartphone* ligado a um Computador para visualizar os *logs* de modo a testar que à medida que o *Smartphone* se afastava da Caixa de Sensores ainda era possível estabelecer uma conexão.

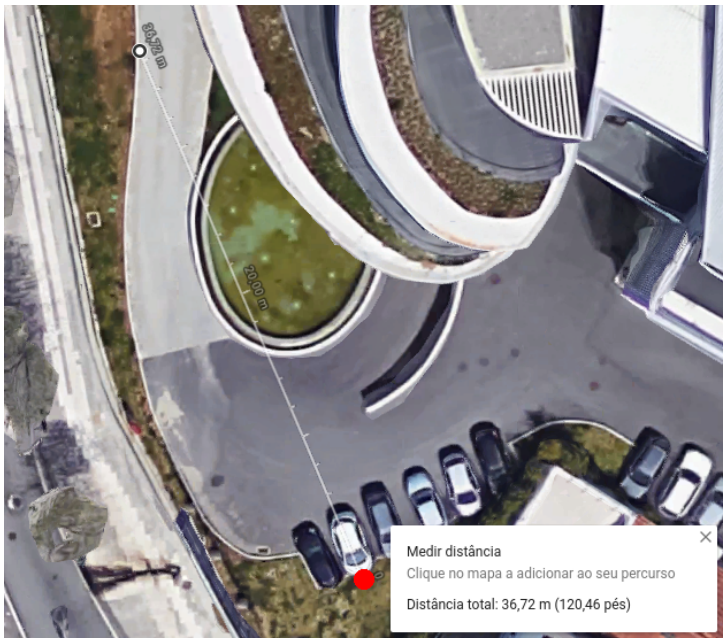


Figura 7.11: Teste de distância máxima para estabelecer conexão

Após verificar o limite de distância máxima definiu-se um cenário de testes para para fazer testes de velocidade representado na figura 7.12, foi instalada uma Caixa de Sensores no ponto assinalado a vermelho na figura 7.12 e foram feitas várias passagens (repetições) a velocidades diferentes até a comunicação entre a Caixa de Sensores e o *Smartphone*/uMDC começarem a falhar.

Devido a limitações de velocidade do primeiro cenário os testes a 50 km/h foram feitos noutro cenário representado na figura 7.13, devido ao meio mudar

entre os cenários os resultados não são comparáveis.

Figura 7.12: Velocidade vs Transmissão de dados - Cenário 1

Figura 7.13: Velocidade vs Transmissão de dados - Cenário 2

Nas tabela 7.1 e 7.2 é possível visualizar os resultados dos testes de velocidade realizados com um *Smartphone* num carro fazendo passagens a diferentes velocidades no cenário representado na figura 7.12 até 40 km/h e no cenário representado na figura 7.12 para testes a 50km/h. Os valores dos tempos foram obtidos através da aplicação implementada com o *Smartphone* ligado a um Computador para visualizar os *logs*.

Velocidade (km/h)	**Tempo de Transmissão de dados entre Caixa de sensores e Smartphone (milissegundos)**	**Resultado**
20	Repetição 1 - 2866 Repetição 2 - 2809 Repetição 3 - 2885 Repetição 4 - 3071 Repetição 5 - 3028	Sucesso
30	3134 2995 2517 2810	Sucesso
40	2927 2870 3200 2898 3166	Sucesso

Tabela 7.1: Velocidade vs Transmissão de dados entre Caixa de Sensores e *Smartphone* - Cenário 1

Velocidade (km/h)	**Tempo de Transmissão de dados entre Caixa de sensores e Smartphone (milissegundos)**	**Resultado**
50	2976 3608 - 9 Características enviadas/recebidas 3247 3404 - 16 Características enviadas/recebidas 2930 - 16 Características enviadas/recebidas	Sucesso em 2 das 5 Repetições

Tabela 7.2: Velocidade vs Transmissão de dados entre Caixa de Sensores e *Smartphone* - Cenário 2

Na tabela 7.3 pode-se visualizar os resultados dos testes de velocidade realizados com um uMDC instalado num carro a 15km/h. Os tempos foram obtidos com recurso a *prints* de *debug* no código do uMDC dos milissegundos atuais nos instantes de conexão estabelecida, e envio dos valores de cada característica.

Velocidade (km/h)	**Tempo entre inicio de conexão e fim do processo de descobrimento da tabela de atributos**	**Tempo de extração dos dados (milissegundos)**	**Tempo total de transmissão de dados entre Caixa de sensores e uMDC (milissegundos)**	**Resultado**
15	4126	3000	7126	Sucesso
15	5102	2274	7376 - 14 Características enviadas/recebidas	Insucesso
15	5102	3300	8402	Sucesso
15	4126	2850	6976	Sucesso
15	6002	2359	8361 - 7376 - 15 Características enviadas/recebidas	Insucesso

Tabela 7.3: Velocidade vs Transmissão de dados entre Caixa de Sensores e uMDC

Comparando os resultados das tabelas 7.1, 7.2 e 7.3 é de realçar que os resultados obtidos com o *Smartphone* são bastante melhores do que os resultados obtidos pelo uMDC. No caso do *Smartphone* verificou-se que a comunicação começou a ficar instalável no limite de velocidade das cidades o que é extremamente aceitável neste tipo de aplicações. Em relação ao uMDC apesar de em termos práticos ter as mesmas funções do *Smartphone* obteve-se resultados bastante piores. Visto que após estarmos conectados dá-se logo início ao processo de obter a informação acerca da tabela de atributos do Servidor GATT através de uma função disponibilizada pela *stack* da Nordic, concluí-se que a diferença de qualidade na troca de dados deve estar centrada em questões de *hardware*, como por exemplo, velocidade de processamento e antena Bluetooth.

7.5 Produção de energia do Painel Solar

Como em determinadas situações, nomeadamente nas Caixas de Sensores Fixas, podem existir severas restrições de energia, mediu-se a energia produzida pelo painel solar escolhido por dia, assim vai ser possível comparar a energia produzida com a consumida pelas Caixas de Sensores em diferentes cenários. Na figura 7.14 está representado o *setup* montado para fazer medições diárias de hora a hora da tensão e corrente fornecidas pelo painel de modo a estimar a potência e consequentemente a energia produzida por dia. Utilizou-se um potenciómetro para regular a tensão de saída do painel para valores acima do limite mínimo de funcionamento do IC responsável pelo carregamento da bateria (4.35 V) mas abaixo da tensão de circuito aberto do painel (6.4 V).

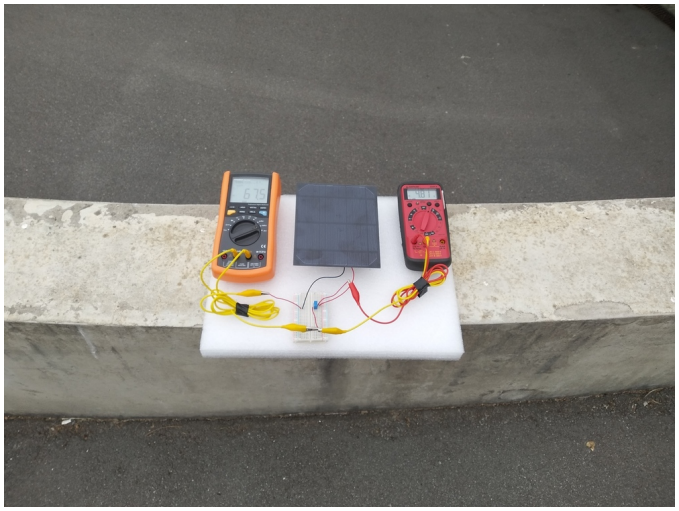


Figura 7.14: *Setup* para a medição de consumos da Caixa de Sensores

Nos testes com o painel solar também se verificou que o IC responsável pelo carregamento da bateria ajusta a corrente pedida ao painel de maneira a não requisitar ao painel uma corrente que o mesmo não possa fornecer, ou seja, se o IC detetar uma queda abrupta na tensão de entrada só pede a corrente

necessária para que o painel mantenha uma tensão acima do mínimo que o IC precisa para funcionar.

Na figura 7.15 está representada uma estimativa da produção de energia do painel solar num dia bastante nublado.

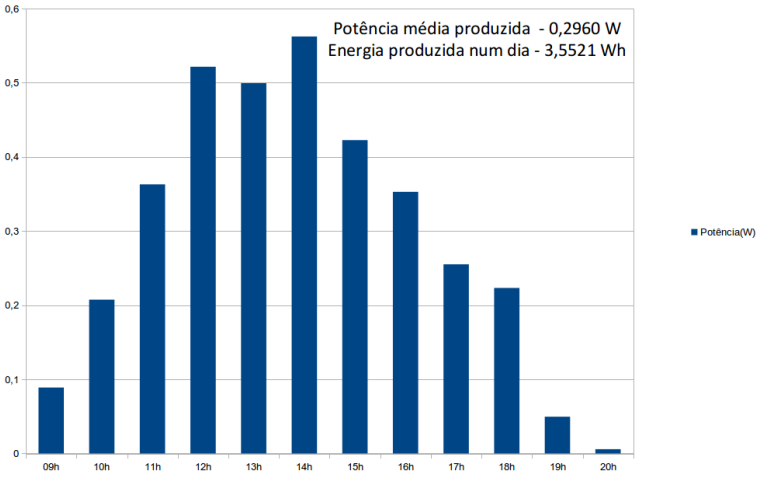


Figura 7.15: Produção do painel solar num dia nublado

Na figura 7.16 está representada uma estimativa da produção de energia do painel solar num dia normal em que ao início da manhã havia pouco sol e a radiação foi aumentando.

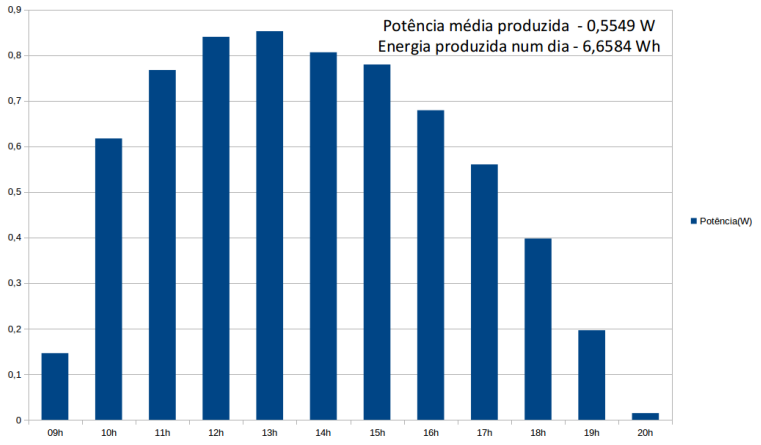


Figura 7.16: Produção do painel solar num dia com Sol

Os dados referentes ao consumo do painel solar em dias com exposição solar diferentes foram recolhidos para existir um termo de comparação com o que a Caixa de Sensores consome num possível cenário em que esteja Fixa com restrições energéticas.

7.6 Consumos da Caixa de Sensores

Os dados referentes ao consumo da Caixa de sensores em vários cenários foram recolhidos com recurso a uma fonte de alimentação com voltímetro e amperímetro integrados, representada na figura 7.17, que permite guardar gráficos com a tensão e corrente consumida em cada instante de tempo. Foram realizados testes para cobrir os casos de uso em que a Caixa de Sensores funciona como um dispositivo Móvel ou Fixo.

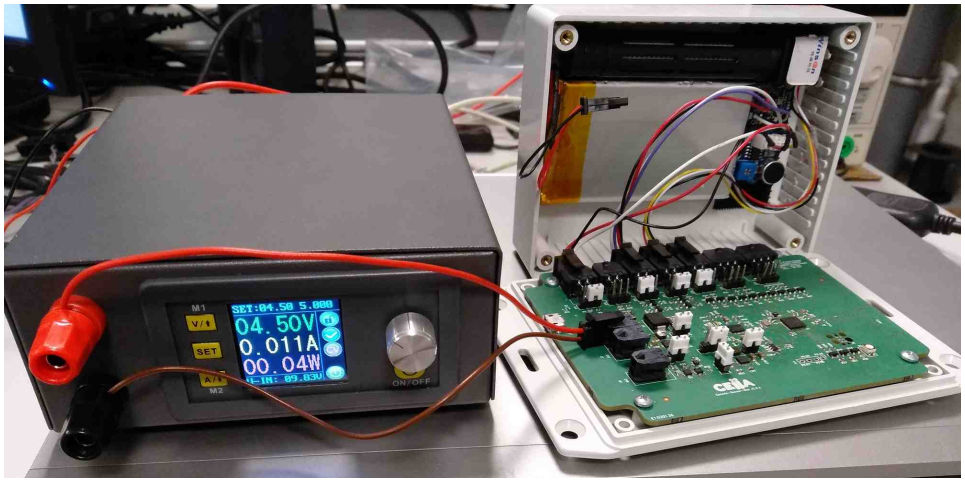


Figura 7.17: *Setup* para a medição de consumos da Caixa de Sensores

7.6.1 Casos de uso da Caixa de Sensores Móvel

Nesta sub-secção são apresentados os testes realizados com possíveis configurações de uma Caixa de Sensores Móvel onde o tempo de cada ciclo de leitura/*report* dos sensores deve ser curto. Como neste caso de uso a Caixa de Sensores vai ser alimentada pela bateria do veículo, não existem grandes restrições energéticas. A Caixa de Sensores nesta situação está instalada num veículo com um uMDC, assim após os dispositivos estabelecerem a primeira vez uma ligação Bluetooth vão permanecer conectados e no fim de cada ciclo de leitura da Caixa de Sensores vão trocar os dados recolhidos. Estes testes pretendem demonstrar o impacto no consumo de energia da Caixa de Sensores com a variação do número de ciclos de leituras/*report*, a variação do número de leituras de cada um dos sensores e da troca de dados entre os dois dispositivos. Para realizar estes testes com alguma precisão separou-se o ciclo de Leitura de dados provenientes dos sensores do ciclo de Comunicação, ou seja, normalmente quando a Caixa de Sensores acaba o ciclo de leitura de dados reporta esses mesmos dados para o uMDC e continua a fazer as medições necessárias para o próximo intervalo de leitura/*report*. Neste caso, em vez de continuar com todos os sensores alimentados sempre que necessário, desligou-se a alimentação dos sensores por *software* para ser possível analisar em separado o impacto de cada parcela.

7.6.1.1 Teste do Impacto do sensor de CO2

Na figura 7.18 está representado o consumo da Caixa de Sensores no Cenário 1 onde a mesma foi configurada da seguinte maneira:

- Reportar os dados de 30 em 30 segundos;
- Janela de *Advertising* de 15 segundos;

- Recolher em cada ciclo de *report*:
  - 15 amostras de ruído;
  - 1 amostra de Temperatura, Humidade e Pressão;
  - 1 amostra da Concentração de CO2.

Este caso representa a situação mais drástica em termos de consumo da Caixa de Sensores visto que o sensor de CO2 em questão precisa de estar ligado sensivelmente 33 segundos antes de cada leitura, como o ciclo de report é de 30 em 30 segundos a alimentação de todos os sensores nunca vai ser desligada. Os picos de consumo são representativos do sensor de CO2 estar ativo (pronto para efetuar leituras), visto que a alimentação dos sensores nunca é desligada. Na figura 7.18 também está representado a potência média consumida e a extrapolação para 24h dessa mesma potência para cada um dos ciclos. Em relação ao Ciclo de Comunicação conseguimos visualizar uma pequena ondulação em torno de um valor médio o que representa os consumos do envio dos dados.

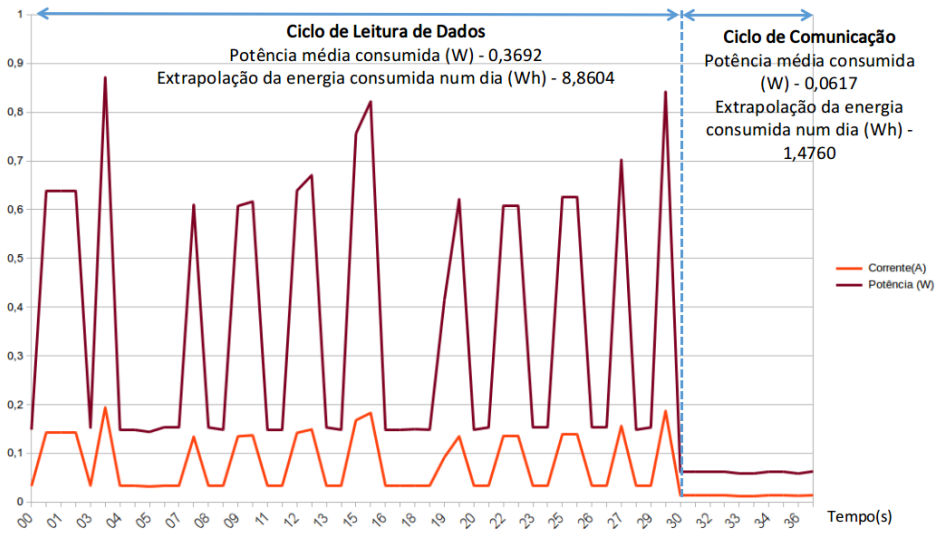


Figura 7.18: Consumo da Caixa de Sensores - Cenário 1

Na figura 7.19 está representado o consumo da Caixa de Sensores no Cenário 2 onde a mesma foi configurada da seguinte maneira:

- Reportar os dados de 60 em 60 segundos;
- Janela de *Advertising* de 15 segundos;
- Recolher em cada ciclo de *report*:
  - 30 amostras de ruído;
  - 2 amostra de Temperatura, Humidade e Pressão;
  - 1 amostra da Concentração de CO2.

Neste caso pretende-se analisar o impacto em termos de consumo do sensor de CO2 na Caixa de Sensores, para isso manteve-se a mesma proporção entre o tempo do ciclo de leitura e o número de leituras de cada sensor para todos os sensores menos para o sensor de CO2. Nesta situação a Caixa de Sensores já pode poupar alguma energia desligando a alimentação dos sensores quando não precisar de realizar leituras até ao momento em que liga o sensor de CO2 (33 segundos antes de efetuar a leitura). Os três picos representados na figura 7.19 antes de ligar os sensores aos 27s são provenientes do sensor de CO2 visto que a alimentação é conjunta. Dos 27 até aos 46 segundos o sensor de CO2 está em pré-aquecimento e os picos posteriores representam o sensor passar para o estado ativo. Neste cenário fica demonstrado que, como era expectável, o sensor de CO2 tem um grande impacto no consumo da Caixa de Sensores, visto que o mesmo diminuiu cerca de 43% pois em 27 dos 60 segundos do ciclo, ou seja, em 45% do tempo a Caixa de Sensores pôde desligar a alimentação quando não necessitou de fazer leituras do sensor de ruído e do Sensor de Temperatura/Humidade/Pressão. Como também era esperado o Ciclo de Comunicação mantém-se idêntico ao do caso anterior.

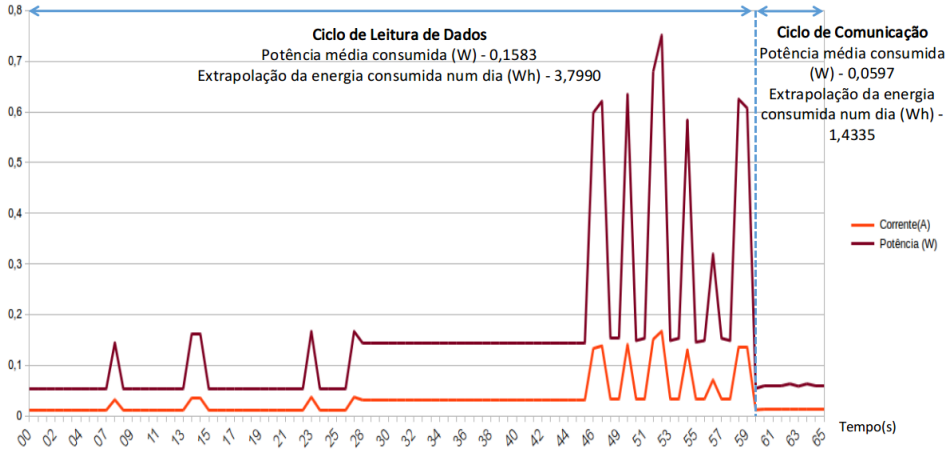


Figura 7.19: Consumo da Caixa de Sensores - Cenário 2

7.6.1.2 Teste do Impacto da variação do número de Ciclos de Leitura

Na figura 7.20 está representado o consumo da Caixa de Sensores no Cenário 3 onde a mesma foi configurada da seguinte maneira:

- Reportar os dados de 60 em 60 segundos;
- Janela de *Advertising* de 15 segundos;
- Recolher em cada ciclo de *report*:
  - 30 amostras de ruído;
  - 2 amostra de Temperatura, Humidade e Pressão;
  - 2 amostra da Concentração de CO2.

Nesta situação pretende-se analisar o impacto da variação do número de ciclos de Leitura na Caixa de Sensores para isso aumentou-se o tamanho do Ciclo de Leitura e consequentemente diminuiu-se o número de ciclos por dia. Manteve-se a proporção entre o tamanho ciclo de leituras e número

de leituras de sensor do Cenário 1, ou seja, o cenário mais drástico. Como expectável o aumento do tamanho do Ciclo de leitura mantendo a proporção relativamente ao número de leituras de cada sensor não tem grande impacto no consumo da Caixa visto que o mesmo diminuiu cerca de 5%. O intervalo de Comunicação manteve-se idêntico aos Cenários anteriores.

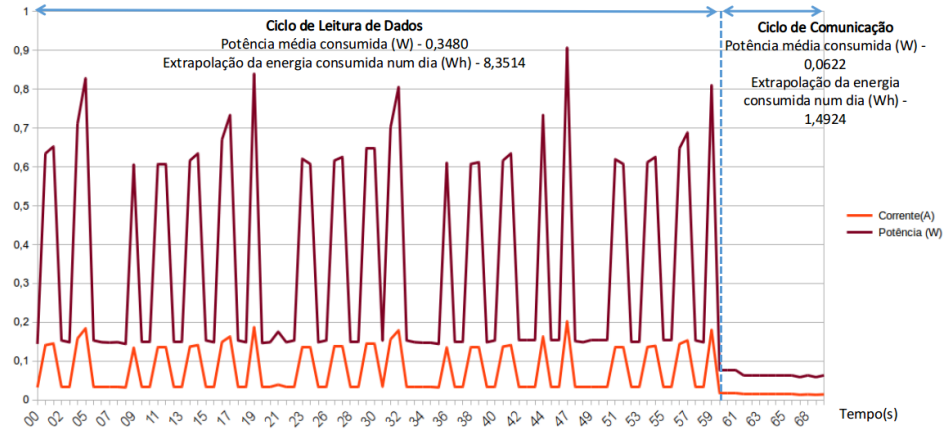


Figura 7.20: Consumo da Caixa de Sensores - Cenário 3

7.6.1.3 Teste do Impacto da variação do número leituras do sensor de Temperatura, Humidade e Pressão

Na figura 7.21 está representado o consumo da Caixa de Sensores no Cenário 4 onde a mesma foi configurada da seguinte maneira:

- Reportar os dados de 60 em 60 segundos;
- Janela de *Advertising* de 15 segundos;
- Recolher em cada ciclo de *report*:
  - 30 amostras de ruído;
  - 4 amostra de Temperatura, Humidade e Pressão;
  - 1 amostra da Concentração de CO2.

Neste caso fez-se um teste para analisar o impacto do sensor de Temperatura, Humidade e Pressão no consumo da Caixa de Sensores, para isso dobrou-se o número de leituras em relação ao Cenário 2 e manteve-se as restantes configurações iguais. Ao aumentar o número de leituras do sensor verificou-se que o consumo de energia manteve-se praticamente igual aumentando 2%. O intervalo de Comunicação manteve-se idêntico aos Cenários anteriores.

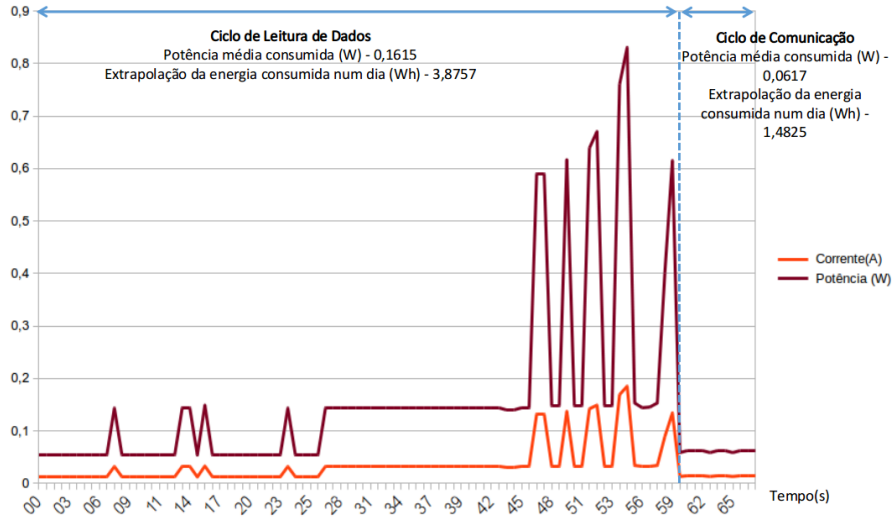


Figura 7.21: Consumo da Caixa de Sensores - Cenário 4

7.6.1.4 Teste do Impacto da variação do número leituras do sensor de Ruído

Na figura 7.22 está representado o consumo da Caixa de Sensores no Cenário 5 onde a mesma foi configurada da seguinte maneira:

- Reportar os dados de 60 em 60 segundos;
- Janela de *Advertising*;
- Recolher em cada ciclo de *report*:
  - 60 amostras de ruído;

– 2 amostra de Temperatura, Humidade e Pressão;

– 1 amostra da Concentração de CO2.

Neste caso fez-se um teste para analisar o impacto do sensor de Ruído no consumo da Caixa de Sensores, para isso dobrou-se o número de leituras em relação ao Cenário 2 e manteve-se as restantes configurações iguais. Ao aumentar o número de leituras do sensor verificou-se que o consumo de energia manteve-se similar. O intervalo de Comunicação manteve-se idêntico aos Cenários anteriores.

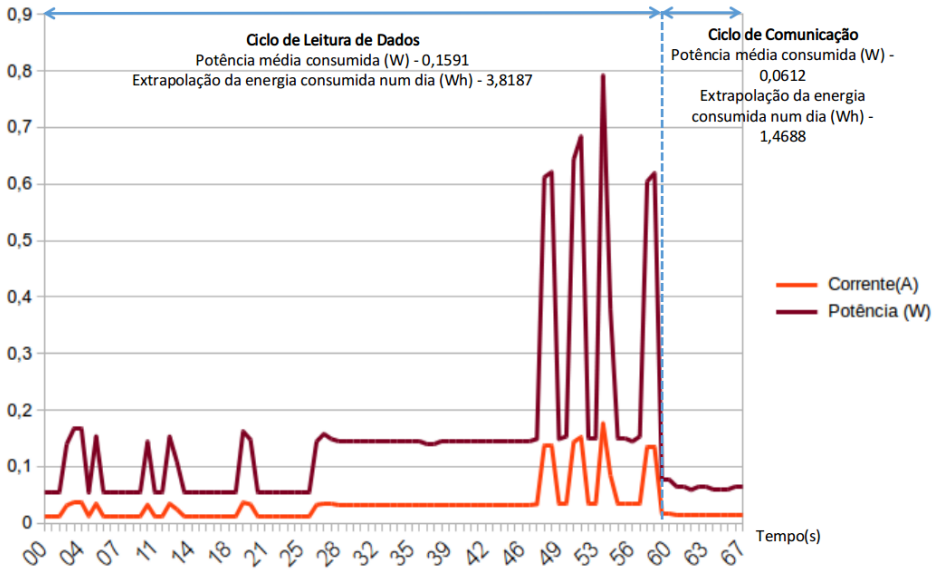


Figura 7.22: Consumo da Caixa de Sensores - Cenário 5

7.6.1.5 Teste do Impacto da troca de dados via *Bluetooth Low Energy*

Na figura 7.23 está representado o consumo da Caixa de Sensores no Cenário 6 onde a mesma foi configurada da seguinte maneira:

- Reportar os dados de 60 em 60 segundos;
- Sem Janela de *Advertising*;
- Recolher em cada ciclo de *report*:
  - 30 amostras de ruído;
  - 2 amostra de Temperatura, Humidade e Pressão;
  - 1 amostra da Concentração de CO2.

Neste caso fez-se um teste para analisar o impacto da troca de dados entre dispositivos no consumo da Caixa de Sensores, para isso utilizou-se as mesmas condições do Cenário 2 mas após recolher os dados colocou-se a Caixa de Sensores num modo *standby* sem fazer *Advertising*. Verificou-se que se a Caixa de Sensores estiver no modo *standby* extrapolando o consumo para 24h consome cerca 1.2960 Wh de Energia. Como a média extrapolada para 24h do consumo da Caixa de Sensores nos Ciclos de Comunicação do Cenário 1 ao 5 é cerca de 1.4702 Wh conclui-se que o impacto da troca de dados no Consumo da Caixa de Sensores é de 0.1743 Wh por dia se os dispositivos estivessem constantemente (24h por dia) a trocar dados.

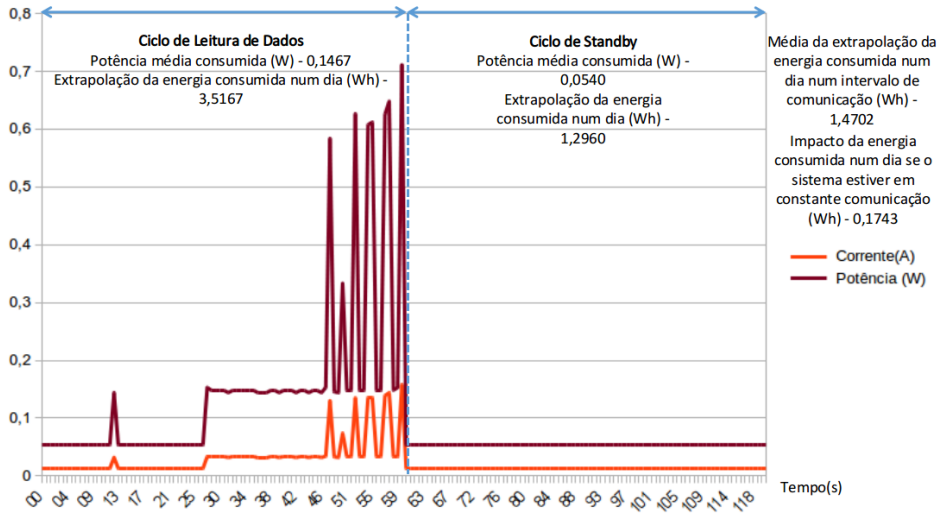


Figura 7.23: Consumo da Caixa de Sensores - Cenário 6

7.6.2 Casos de uso da Caixa de Sensores Fixa

Nesta sub-secção são apresentados os testes realizados com possíveis configurações de uma Caixa de Sensores Fixa onde o tempo de cada ciclo de leitura/*report* dos sensores deve ser maior do que na Caixa de Sensores Móvel. Neste caso de uso existem restrições energéticas visto que a Caixa de Sensores vai ser alimentada pelo painel solar, que num dia nublado produz cerca de 3,55 Wh e uma bateria de 7,4 Wh. A Caixa de Sensores nesta situação está instalada num ponto Fixo, assim após cada ciclo de leitura a Caixa de Sensores vai estar a fazer *Advertising* com o objetivo de estabelecer uma ligação com um uMDC ou *Smartphone* para reportar os dados recolhidos no ciclo anterior. Estes testes pretendem demonstrar que a Caixa de Sensores é autossustentável e o impacto no consumo de energia da Caixa de Sensores com a variação do tempo de *Advertising*. Para realizar estes testes utilizou-se o mesmo método descrito na sub-secção 7.6.1.

7.6.2.1 Teste do Impacto do *Advertising*

Na figura 7.24 está representado o consumo da Caixa de Sensores no Cenário 7 onde a mesma foi configurada da seguinte maneira:

- Reportar os dados de 300 em 300 segundos;
- Janela de *Advertising* de 30 segundos;
- Recolher em cada ciclo de *report*:
  - 150 amostras de ruído;
  - 1 amostra de Temperatura, Humidade e Pressão;
  - 1 amostra da Concentração de CO2.

Neste caso pretende-se analisar o consumo com uma possível configuração de uma Caixa de Sensores Fixa e o impacto da janela de *Advertising* no consumo da Caixa de Sensores. Ao estudar este impacto pretendemos analisar o custo energético de estar constantemente a tentar estabelecer uma conexão com um uMDC ou *Smartphone* para reportar os dados de leitura do ciclo anterior. Neste cenário a Caixa de Sensores após o ciclo de Leitura fez *Advertising* durante 30 segundos e esteve sem fazer *Advertising*, em modo *Standby*, durante 270 segundos. Conclui-se que a diferença de consumo entre o modo de *Advertising* e o de *Standby* extrapolando para um dia em constante funcionamento é de 0,0460 Wh o que representa um aumento de sensivelmente 3,5% no consumo da Caixa de Sensores. Em relação ao consumo do ciclo de leitura *versus* a produção do painel solar, o painel solar num dia nublado produz cerca de mais 45% de energia do que a Caixa de Sensores consome em 24h.

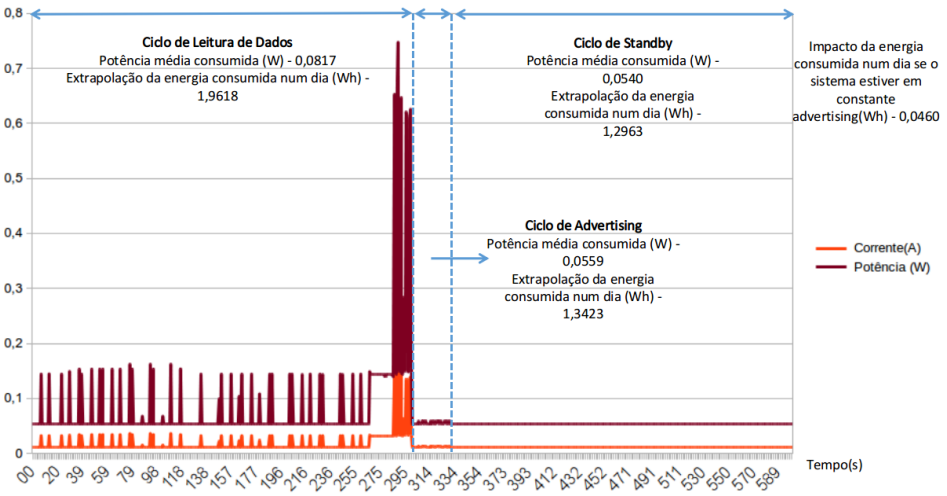


Figura 7.24: Consumo da Caixa de Sensores - Cenário 7

7.6.3 Análise Geral dos Resultados

Visto que a Caixa de Sensores foi projetada com o objetivo de ser modular e ser possível só com alterações de *software* alterar os sensores existentes na mesma, dependendo dos interesses de cada região, os testes de impacto de cada um dos sensores deixa de ser muito relevante visto que no futuro pode ser possível trocar, por exemplo, o sensor de CO2 que é o sensor que tem mais impacto no consumo da mesma por um sensor com consumos mais reduzidos.

Os testes mais relevantes foram os que provaram o conceito do funcionamento da Caixa de Sensores Fixa onde é possível, sem estar estático perto da Caixa de Sensores, recolher os dados enviados pela mesma.

Também foi importante verificar que a comunicação BLE em termos de consumo quer na troca de dados quer em termos de *Advertising* tem pouco impacto no consumo geral da Caixa de Sensores.

Os testes de produção de energia do painel solar serviram para ter uma noção de que um painel com dimensões reduzidas, próximas da Caixa de Sensores, produz energia mais do que suficiente para alimentar a Caixa de Sensores num possível cenário da Caixa estar instalada num ponto Fixo.

This page intentionally left blank.

Capítulo 8

Conclusões e Trabalho Futuro

A presente tese teve como objetivo criar um sistema para ser integrado na atual infraestrutura da rede Mobi.me que seja capaz de monitorizar parâmetros relevantes de cada cidade para os seus cidadãos e gestores, obtendo um mapa de dados de uma cidade ou região. Este capítulo resumirá o equilíbrio entre os requisitos iniciais e os resultados finais do projeto nos quatro grandes blocos desenvolvidos, bem como, sugestões serão feitas sobre o trabalho futuro a ser realizado pelo CEiiA.

Para cumprir com o principal requisito desta tese desenvolveu-se uma Caixa de Sensores modular para que seja possível alterar os sensores consoante a necessidade de cada região, bem como, a frequência de leitura de cada sensor por de *report* da Caixa de Sensores.

Em relação à Caixa de Sensores, os resultados foram bastante bons, alcançando todos os requisitos propostos, provou-se que as duas vertentes pretendidas para o dispositivo, Móvel ou Fixo são viáveis.

Em relação ao *software* desenvolvido tanto no uMDC como no *Smartphone* também cumpriu com os objetivos propostos de modo a fazer a ponte em termos de comunicação entre a Caixa de Sensores e o Servidor, no cenário

das Caixas de Sensores Fixas que é o mais critico em termos de comunicação ficou provado que é possível fazer passagens a diferentes velocidades com um uMDC ou *Smartphone* e recolher os dados das Caixas de Sensores.
O Servidor da Caixa de Sensores desenvolvido para a análise das mensagens enviadas pelo uMDC e *Smartphone* e a demonstração das mesmas num interface gráfico foi elaborado com o intuito de demonstrar a potencialidade deste sistema sem interferir com o sistema existente, o que foi alcançando com sucesso.

Em termos de trabalho futuro que se pretende continuar a desenvolver no CEiiA deve-se estudar a possibilidade do controlo da alimentação dos sensores ser feito individualmente de forma a amenizar o impacto de sensores que consumam mais energia como é o caso do sensor de CO2 utilizado. Também pretende-se definir um conjunto de sensores mais adequado em termos de consumo e fiabilidade de medições para este tipo de aplicações.
Pretende-se melhorar a performance na comunicação entre a Caixa de Sensores e o uMDC para resultados próximos dos obtidos com o *Smartphone*.
Para a inclusão da funcionalidade do Utilizador como *gateway* através do *Smartphone* ser completamente transparente para o utilizador final a aplicação desenvolvida nesta tese, para efeitos de prova de conceito, deve ser totalmente integrada numa aplicação já existente do CEiiA que permite a interação com o sistema Mobi.me. No futuro podem ser desenvolvidas métricas mais avançadas para cada um dos tipos de dados recolhidos.

Esta oportunidade de realizar a tese no CEiiA contribuiu para consolidar todo conhecimento adquirido durante o meu percurso académico. A realização deste projeto, a troca de ideias com outros colegas da empresa, teve um impacto direto no meu crescimento.

Em conclusão, gostaria de agradecer ao CEiiA por permitir o desenvolvimento desta tese e fornecer as ferramentas necessárias o efeito, visto que foi uma experiência muito enriquecedora e contribuiu para a aquisição de novas habilidades, o que certamente será útil ao longo de toda a meu carreira profissional.

This page intentionally left blank.

Bibliografia

[1] EMBRAER KC-390: 450.000 HORAS DE ENGENHARIA CEiiA. Acessado em Janeiro de 2018. Disponível em: https://www.ceiia.com/single-post/2016/07/04/EMBRAER-KC-390-450000-HORAS-DE-ENGENHARIA-CEiiA.

[2] MEDUSA DEEP SEA. Acessado em Janeiro de 2018. Disponível em: https://www.ceiia.com/offshore-deep-sea.

[3] Vito Albino, Umberto Berardi and Rosa Maria Dangelico "Smart Cities: Definitions, Dimensions, Performance, and Initiatives". The Society of Urban Technology, 2015.

[4] Andres Monzon "Smart Cities Concept and Challenges: Bases for the Assessment of Smart City Projects". Springer International Publishing Switzerland, 2015.

[5] Mircea Eremia, Lucian Toma and Mihai Sanduleac "The Smart City Concept in the 21st Century". 10th International Conference Interdisciplinarity in Engineering, 2016.

[6] Libelium: 50 Sensor Applications for a Smarter World. Acessado em Fevereiro de 2018. Disponível em: http://www.libelium.com/resources/top_50_iot_sensor_applications_ranking/.

[7] ESRI India: 'Sensors' for Smart 'Cities'. Acessado em Fevereiro de 2018. Disponível em: http://www.esri.in/esri-news/publication/vol9-issue1/articles/sensors-for-smart-cities.

[8] World Health Organization. Media centre - Ambient (outdoor) air quality and health. Acessado em Fevereiro de 2018. Disponível em: http://www.who.int/mediacentre/factsheets/fs313/en/.

[9] CityLab: How Portable Air Sensors Are Changing Pollution Detection. Acessado em Fevereiro de 2018. Disponível em: https://www.citylab.com/environment/2015/08/how-portable-air-sensors-are-changing-pollution-detection/401147/.

[10] Ministry for the environment: Why good air quality is important. Acessado em Fevereiro de 2018. Disponível em: http://www.mfe.govt.nz/more/environmental-reporting/air/air-domain-report-2014/why-good-air-quality-important.

[11] Siemens: Smart Cities - Prognose von Emissionen. Acessado em Fevereiro de 2018. Disponível em: https://www.siemens.com/innovation/de/home/pictures-of-the-future/infrastruktur-und-finanzierung/smart-cities-prognose-von-emissionen.html.

[12] CO2Meter: CO and CO2 – What's the difference?. Acessado em Fevereiro de 2018. Disponível em: https://www.co2meter.com/blogs/news/1209952-co-and-co2-what-s-the-difference.

[13] CLIMATE 101: AIR POLLUTION. Acessado em Fevereiro de 2018. Disponível em: https://www.nationalgeographic.com/environment/global-warming/pollution/.

[14] Comissão de Coordenação e Desenvolvimento Regional de Lisboa e Vale do Tejo. Acessado em Fevereiro de 2018. Disponível em: http://www.ccdr-lvt.pt/pt/o-ar-e-os-poluentes-atmosfericos/8082.htm.

[15] Comissão de Coordenação e Desenvolvimento Regional do Norte: Qualidade do Ar. Acessado em Fevereiro de 2018. Disponível em: http://www.ccdr-n.pt/servicos/ambiente/qualidade-ar.

[16] Agência Portuguesa do Ambiente "Evolução da qualidade do ar em Portugal entre 2001 e 2005 Relatório"2008

[17] NOMINET: LIST OF SMART CITY PROJECTS. Acessado em Fevereiro de 2018. Disponível em: https://www.nominet.uk/list-smart-city-projects/.

[18] Postcapes: Smart City Applications. Acessado em Fevereiro de 2018. Disponível em: https://www.postscapes.com/internet-of-things-award/smart-city-application/.

[19] Sharing Cities: BUILDING SMART CITIES TOGETHER. Acessado em Fevereiro de 2018. Disponível em: http://www.sharingcities.eu/.

[20] Smart Cities: SHARING CITIES: PROMOVER A PARTICIPAÇÃO NA CIDADE DE AMANHÃ. Acessado em Fevereiro de 2018. Disponível em: http://www.smart-cities.pt/pt/noticia/lisboa-sharingcities-cidadao2311/.

[21] Smart Cities: SHARING CITIES: SHARING CITIES: UM NOVO PARADIGMA DE GESTÃO URBANA SUSTENTÁVEL. Acessado em Fevereiro de 2018. Disponível em: http://www.smart-cities.pt/pt/noticia/sharing-cities-paradigma0811/.

[22] Smart Cities: SHARING CITIES: SHARING CITIES: ENERGIA, MOBILIDADE E INFRAESTRUTURAS. Acessado em Fevereiro de

2018. Disponível em: http://www.smart-cities.pt/pt/noticia/sharing-cities-eixos-1811/.

[23] Array of Things. Acessado em Fevereiro de 2018. Disponível em: https://arrayofthings.github.io/.

[24] Smart Citizen - Hardware. Acessado em Fevereiro de 2018. Disponível em: https://github.com/fablabbcn/smartcitizen-kit/tree/master/hardware.

[25] The Smart Citizen Kit: Crowdsourced Environmental Monitoring. Acessado em Fevereiro de 2018. Disponível em: https://www.kickstarter.com/projects/acrobotic/the-smart-citizen-kit-crow.

[26] Tomorrow's Cities: How Barcelona shushed noise-makers with sensors. Acessado em Fevereiro de 2018. Disponível em: https://www.bbc.com/news/technology-41015486.

[27] Telensa PLANet: Public Lighting Active Network. Acessado em Fevereiro de 2018. Disponível em: https://www.telensa.com/smart-lighting/.

[28] Bigbelly: Smart City Solutions. Acessado em Fevereiro de 2018. Disponível em: http://bigbelly.com/platform/#wheeled_interior_lift_bin.

[29] SmartSantander. Acessado em Fevereiro de 2018. Disponível em: http://www.smartsantander.eu/.

[30] Libelium: Waspmote Overview . Acessado em Fevereiro de 2018. Disponível em: http://www.libelium.com/products/waspmote/overview/.

[31] Libelium World: Smart City project in Serbia for environmental monitoring by Public Transportation . Acessado em Fevereiro de 2018. Dis-

ponível em: http://www.libelium.com/smart_city_environmental_parameters_public_transportation_waspmote/.

[32] Institute for Scientific Interchange. Acessado em Fevereiro de 2018. Disponível em: https://www.isi.it/en/home.

[33] Everyaware: Sensing Air Pollution. Acessado em Fevereiro de 2018. Disponível em: http://www.everyaware.eu/activities/case-studies/air-quality/.

[34] Internet of Things (IoT). Acessado em janeiro de 2018. Disponível em: http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT.

[35] A Simple Explanation Of 'The Internet Of Things'. Acessado em janeiro de 2018. Disponível em: https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#282787951d09.

[36] ZigBee. Acessado em Março de 2018. Disponível em: https://standards.ieee.org/findstds/standard/802.15.4v-2017.html.

[37] ZigBee. Acessado em Março de 2018. Disponível em: https://www.gta.ufrj.br/grad/10_1/zigbee/.

[38] IEEE 802.15 - IEEE 802.15 WPAN Task Group 4 (TG4). Acessado em Março de 2018. Disponível em: http://www.ieee802.org/15/pub/TG4.html.

[39] Yonghua Songa, Jin Lin, Ming Tangb, Shufeng Dong "An Internet of Energy Things Based on Wireless LPWAN", 2017.

[40] Paulo Sérgio Rangel Garcia, João Henrique Kleinschmidt, "Tecnologias Emergentes de Conectividade na IoT: Estudo de Redes LPWAN", 2017

[41] LPWAN (low-power wide area network). Acessado em Março de 2018. Disponível em: http://internetofthingsagenda.techtarget.com/definition/LPWAN-low-power-wide-area-network.

[42] LoRaWAN simply explained. Acessado em Março de 2018. Disponível em: http://jensd.be/category/lora.

[43] LoRa Alliance Wide Area Networks for IoT. Acessado em Março de 2018. Disponível em: https://www.lora-alliance.org/.

[44] Low Power Wide Area Network (LPWA) - LOW POWER WIDE AREA NETWORK TECHNOLOGY EXPLAINED. Acessado em Março de 2018. Disponível em: https://www.link-labs.com/blog/low-power-wide-area-network-lpwa.

[45] An Overview Of Narrowband IoT (NB-IoT). Acessado em Março de 2018. Disponível em: https://www.link-labs.com/blog/overview-of-narrowband-iot.

[46] Ericsson, "NB-IOT: A SUSTAINABLE TECHNOLOGY FOR CONNECTING BILLIONS OF DEVICES", 2016.

[47] Sigfox, "Sigfox Technical Overview", 2017.

[48] ABI research - ABI Research is a technology market intelligence company. Acessado em Abril de 2017. Disponível em: https://www.abiresearch.com/.

[49] Microchip - Introduction to Bluetooth Low Energy. Acessado em Abril de 2018. Disponível em: http://microchipdeveloper.com/wireless:ble-introduction.

[50] Generic Access Profile. Acessado em Abril de 2018. Disponível em: https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile.

[51] Tomas Rosa, "Bypassing Passkey Authentication in Bluetooth Low Energy", 2013.

[52] Jean Jacques Meneu, "Security for Bluetooth Low Energy", 2016.

[53] Bluetooth blog - Bluetooth Pairing Part 4: LE Secure Connections - Numeric Comparison. Acessado em Abril de 2018. Disponível em: https://blog.bluetooth.com/bluetooth-pairing-part-4.

[54] Carles Gomez, Joaquim Oller e Josep Paradells, "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology", 2012.

[55] Bluetooth Low Energy - Part 1: Introduction To BLE. Acessado em Fevereiro de 2018. Disponível em: https://www.mikroe.com/blog/bluetooth-low-energy-part-1-introduction-ble.

[56] Robert Davidson, Akiba, Carles Cufí, Kevin Townsend "Getting Started with Bluetooth Low Energy"O'Reilly Media Inc, 2014.

[57] Bluetooth low energy Characteristics, a beginner's tutorial. Acessado em Fevereiro de 2018. Disponível em: https://devzone.nordicsemi.com/tutorials/b/bluetooth-low-energy/posts/ble-characteristics-a-beginners-tutorial.

[58] MQTT 101 – How to Get Started with the lightweight IoT Protocol. Acessado em Abril de 2018. Disponível em: https://www.hivemq.com/blog/how-to-get-started-with-mqtt.

[59] Zhengzhou Winsen Electronics Technology CO.,LTD, "MH-Z16 Intelligent Infrared Gas Module Manual", Acessado em Abril de 2018.

[60] Bosch Sensortec, "BME280 Combined humidity and pressure sensor", 2016.

[61] Seed studio - Grow the difference, "Grove - Sound Sensor User Manual", 2015.

[62] Nordic Semiconductor, "nRF52832 Product Specification v1.4", 2017

[63] Texas Instruments, "bq2403x Single-Chip Charge and System Power-path Management IC", 2014

[64] Bluetooth Low Energy - Part 1: Introduction To BLE. Acessado em Fevereiro de 2018. Disponível em: `https://batteryuniversity.com/learn/article/charging_lithium_ion_batteries`.

[65] Texas Instruments, "bq297xx Cost-Effective Voltage and Current Protection Integrated Circuit for Single-Cell Li-Ion and Li-Polymer Batteries", 2018

[66] Texas Instruments, "TPS6300x High-Efficient Single Inductor Buck-Boost Converter With 1.8-A Switches", 2015

[67] Texas Instruments, "TPS27081A 1.2-V to 8-V, 3-A PFET High-Side Load Switch With Level Shift and Adjustable Slew Rate Control", 2015

[68] CYPRESS, "AN91445 – Antenna Design and RF Layout Guidelines", 2014

[69] Texas Instruments, "AN-1811 Bluetooth Antenna Design", 2013

[70] NXP, "BLE Antenna Design Guide", 2013

Anexo A. Planeamento

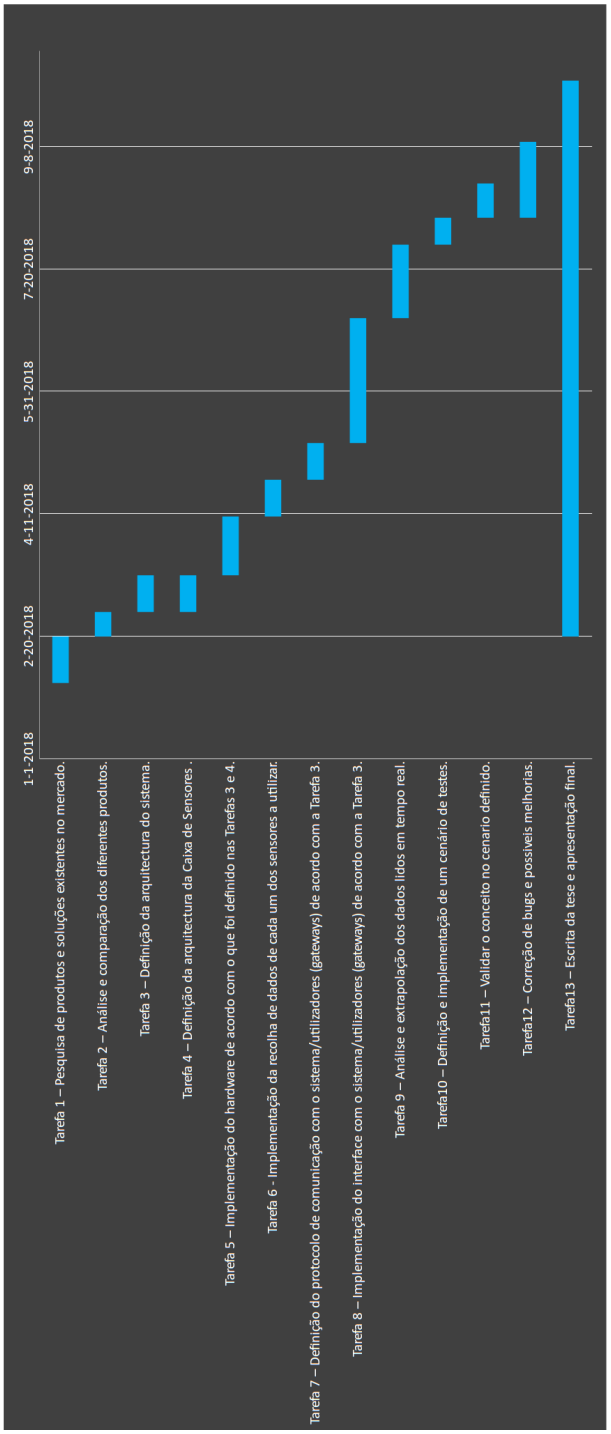


Figura 1: Proposta de planeamento

This page intentionally left blank.

Anexo B. Esquemáticos

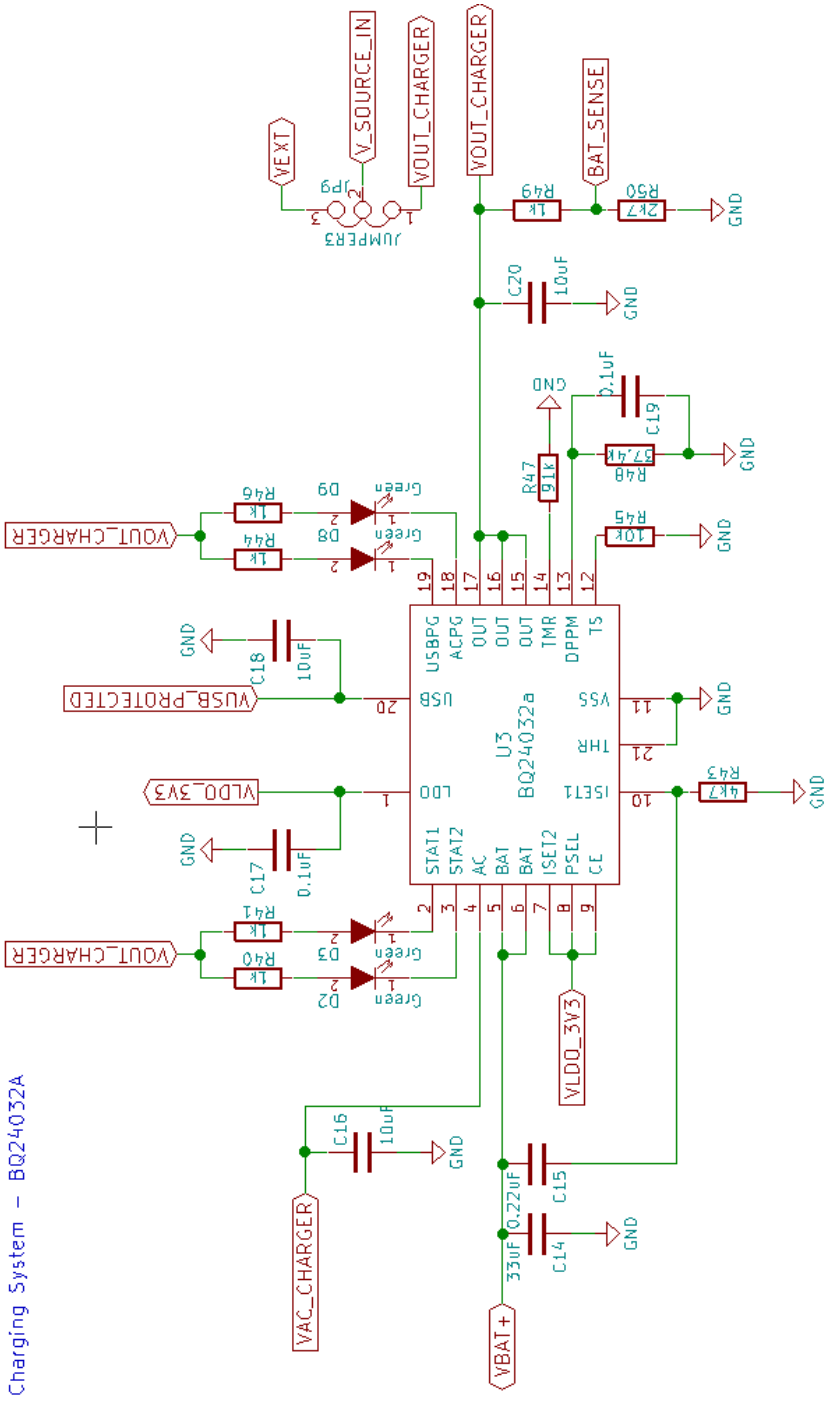


Figura 2: Esquemático completo do sistema de carregamento da Caixa de Sensores

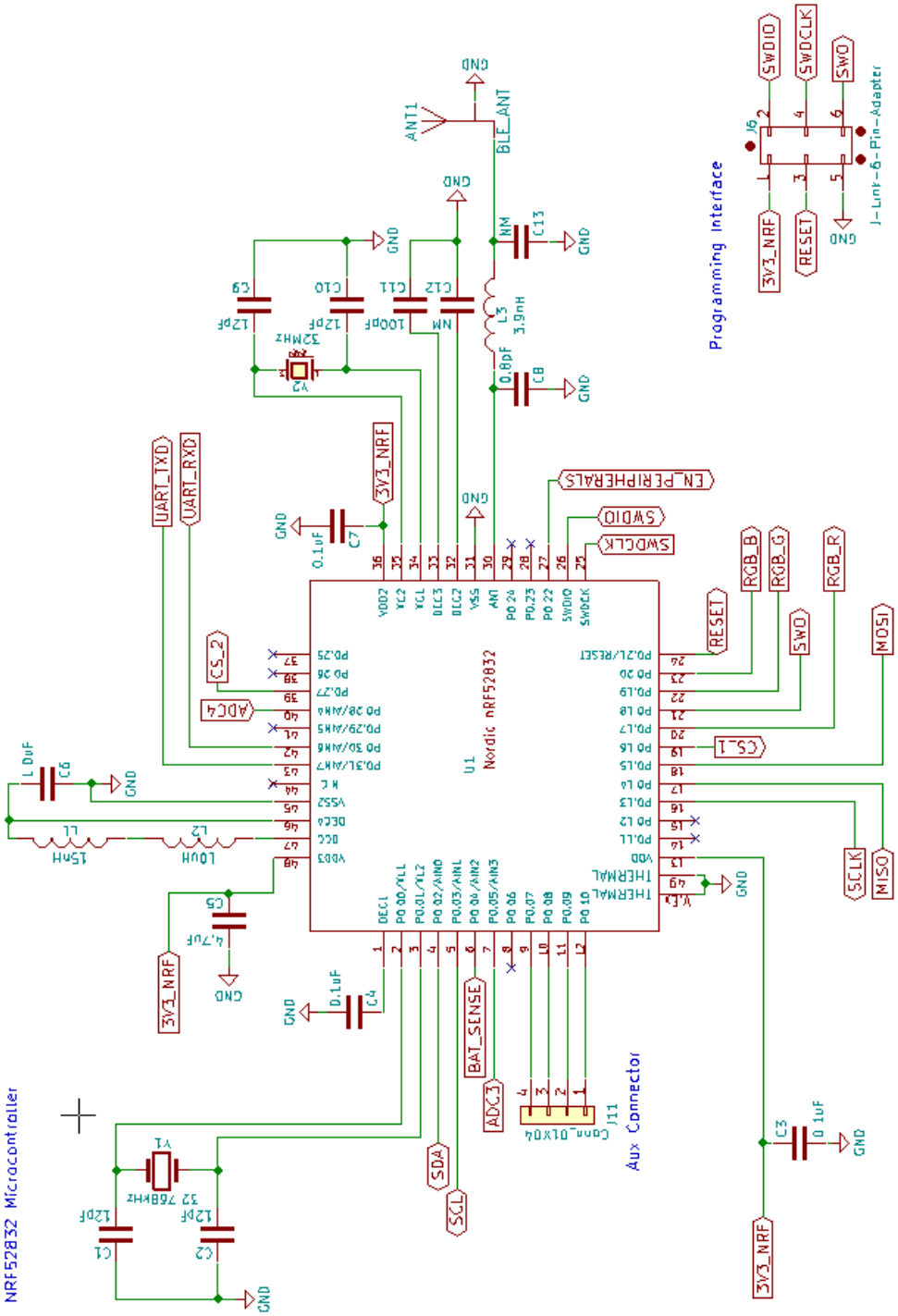


Figura 3: Esquemático do Microcontrolador

Anexo C. Resultados Caixa de Sensores Móvel Cenário 1

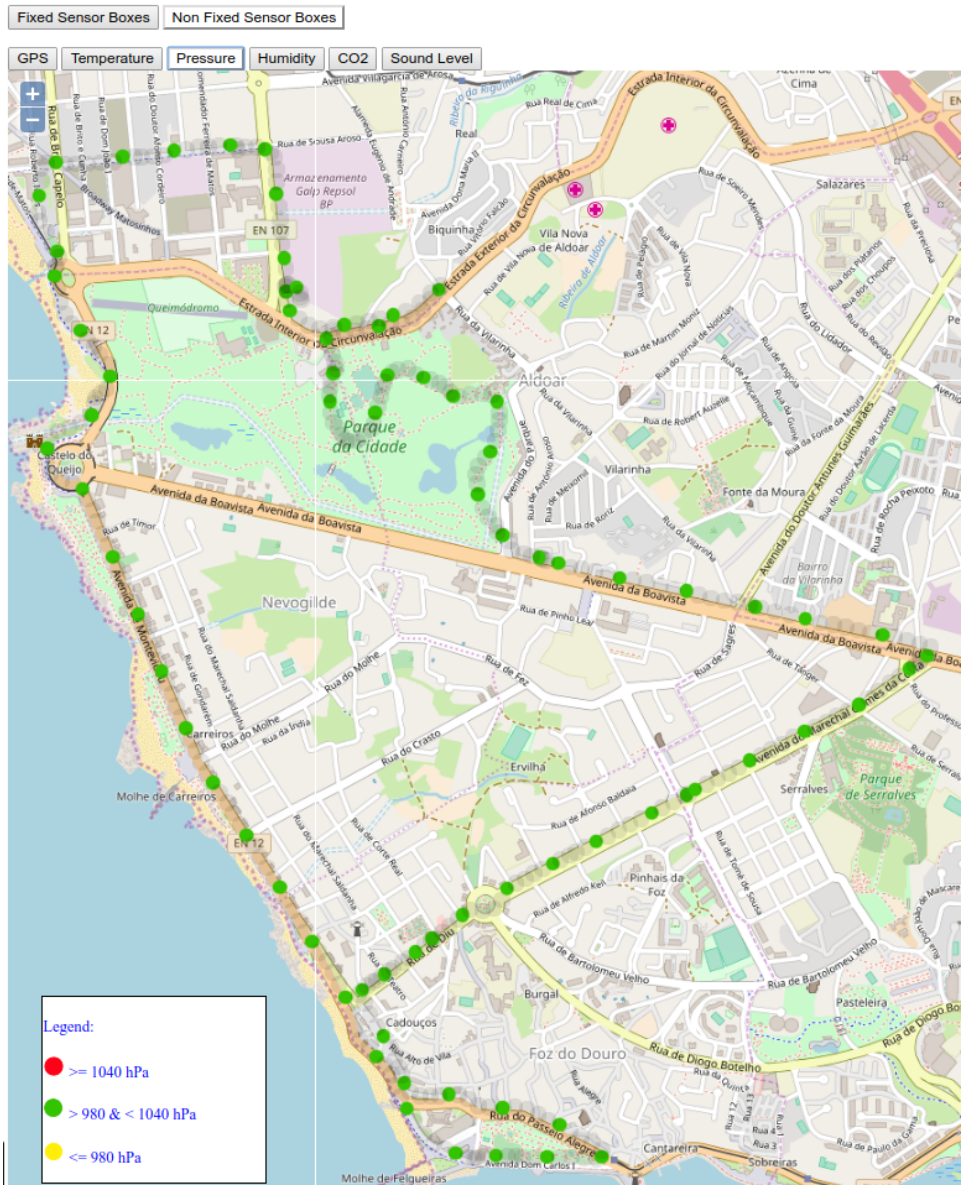


Figura 4: Caixa de Sensores Móvel: *Logs* de Pressão - Cenário 1

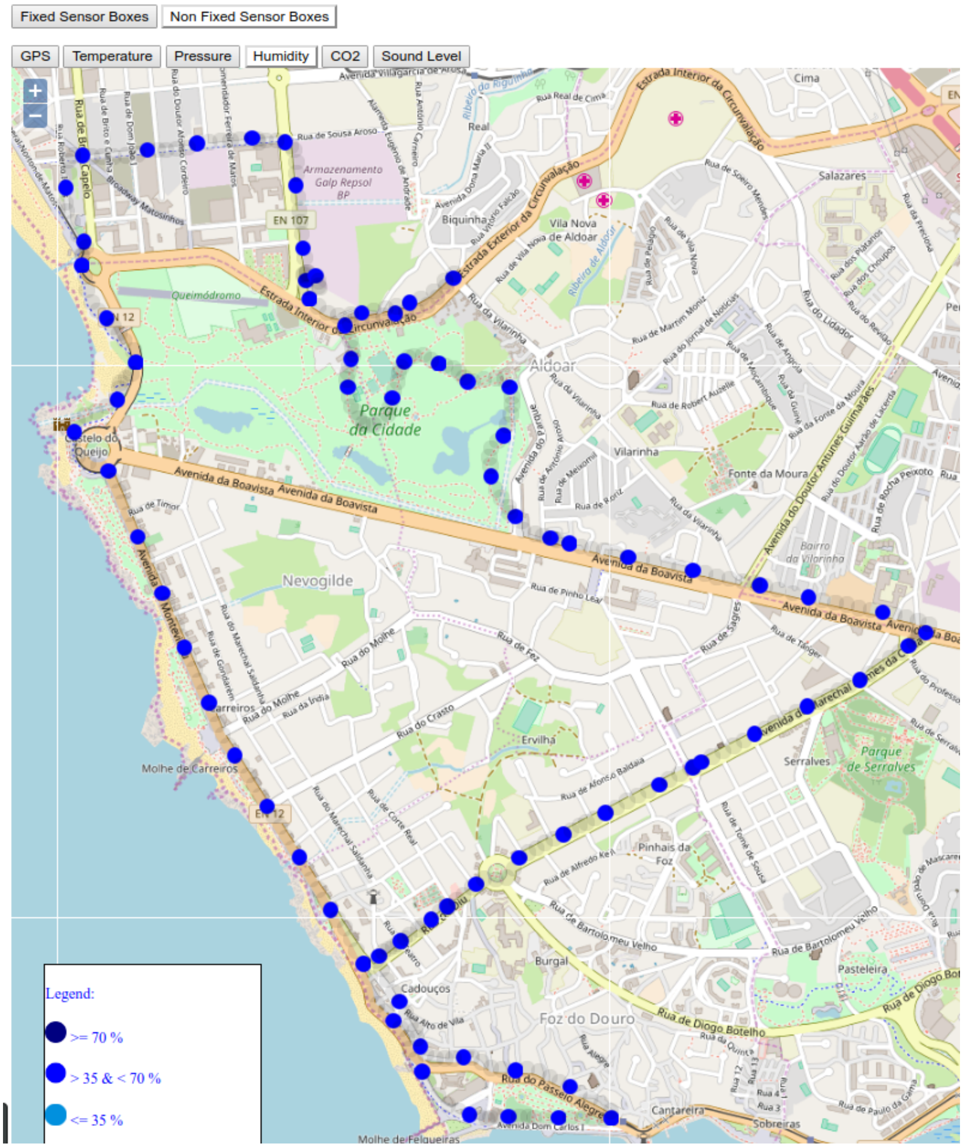


Figura 5: Caixa de Sensores Móvel: Dados de Humidade - Cenário 1

Figura 6: Caixa de Sensores Móvel: Dados de CO2 - Cenário 1

This page intentionally left blank.