PM

Monitorização Habitacional com Uma Rede de Sensores Sem Fios

PROJETO DE MESTRADO

João Francisco Azevedo Baptista

MESTRADO EM ENGENHARIA ELETROTÉCNICA-TELECOMUNICAÇÕES

UNIVERSIDADE da MADEIRA

A Nossa Universidade

www.uma.pt

setembro | 2017

Monitorização Habitacional com Uma Rede de Sensores Sem Fios

PROJETO DE MESTRADO

João Francisco Azevedo Baptista

MESTRADO EM ENGENHARIA ELETROTÉCNICA-TELECOMUNICAÇÕES

ORIENTADOR

Joaquim Amândio Rodrigues Azevedo

UNIVERSIDADE da MADEIRA

Faculdade de Ciências Exatas e da Engenharia

Monitorização habitacional com uma rede de sensores sem fios

Dissertação submetida à Universidade da Madeira para obtenção do Grau de Mestre em Engenharia Eletrotécnica - Telecomunicações

Orientador: Professor Doutor Joaquim Amândio Rodrigues Azevedo

João Francisco Azevedo Baptista

2017

Resumo

O objetivo deste trabalho foi projetar uma rede de sensores sem fios para a monitorização de diversas variáveis de interesse numa habitação, relacionadas com a qualidade do ar interior e com o consumo elétrico.

Inicialmente foi feita uma pesquisa sobre os parâmetros relevantes a serem medidos numa habitação e, posteriormente, pesquisaram-se produtos e trabalhos desenvolvidos com o mesmo intuito. Com base nas informações encontradas, os parâmetros selecionados foram a temperatura, a humidade, o dióxido de carbono, os compostos orgânicos voláteis, as partículas em suspensão e a corrente elétrica.

Projetou-se a rede de sensores sem fios, recorrendo a três tipos de nós, o nó terminal, o nó *router* e o nó coordenador. Foram desenvolvidos cinco protótipos, para medição dos parâmetros, e um outro, ligado ao coordenador, para a receção e armazenamento dos dados. Este módulo também permitiu visualizar os dados em tempo real através de uma aplicação *web*.

Com estas medições pretendeu-se, através de um sistema económico, relativamente pequeno e de baixo consumo, que o utilizador tenha uma melhor perceção da qualidade do ar interior e do consumo de energia, e que possa adotar medidas para melhoria do conforto interior e para o controlo da energia consumida.

Palavras-chave: Monitorização habitacional, Redes de Sensores Sem Fios (RSSF), ZigBee, qualidade do ar interior, consumo de energia.

Abstract

The objective of this work was to design a wireless sensor network for the monitoring of several variables of interest in a house, related to indoor air quality and electricity consumption.

Initially research was done on the relevant parameters to be measured in a house and was done on products and projects developed with the same intention. Based on the information found, the parameters selected were temperature, humidity, carbon dioxide, volatile organic compounds, suspended particles and electric current.

The wireless sensor network was designed using three types of nodes, the terminal node, the router node, and the coordinating node. Five prototypes were developed to measure the parameters and the coordinator was developed for data reception and storage. This module also allows you to view the data in real time, through a web application accessible in the local network.

With these measurements and their results, a relatively small and low-energy system is intended to provide the user with a better perception of indoor air quality and energy consumption, empowering the user with knowledge to improve those parameters.

Keywords: Housing Monitoring, Wireless Sensor Networks (WSN), ZigBee, indoor air quality, energy consumption.

Agradecimentos

Ao Professor Joaquim Amândio Rodrigues Azevedo, meu orientador, pela sua disponibilidade e conhecimento transmitido ao longo da execução deste projeto.

Ao Eng. Filipe Santos e ao Eng. Jorge Lopes por todo o auxílio e disponibilidade no desenvolvimento do trabalho.

À minha mãe, a toda a família, à Magda e aos meus amigos, por terem me acompanhado ao longo do meu percurso académico.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

API	*Application Programming Interface*
AT	*Transparent Mode*
AVAC	Aquecimento, ventilação e ar condicionado
COV	Compostos orgânicos voláteis
CO_2	Dióxido de Carbono
I2C	*Inter-integrated Circuit*
PM_{10}	Partículas inaláveis de diâmetro inferior a 10 micrómetros
$PM_{2,5}$	Partículas inaláveis de diâmetro inferior a 2,5 micrómetros
PPM	Partes por milhão
RSSF	Rede de Sensores Sem Fio
RSSI	*Received Signal Strength Indicator*
WSN	*Wireless Sensor Network*

1. Introdução

Este capítulo tem o principal objetivo de introduzir o tema das redes de sensores sem fios, e apresentar a motivação e os objetivos para a elaboração deste projeto.

1.1. Motivação

Uma rede de sensores sem fios (RSSF) é uma rede de telecomunicações formada por diversos nós, em que cada nó está equipado com um sensor capaz de detetar algum fenómeno físico, como o calor, a humidade e a luz, entre outros. As RSSF são consideradas uma forma revolucionária de recolher dados, resultando num sistema de informação e comunicação que permite avaliar um determinado ambiente, sendo possível, tomar decisões consoante os resultados obtidos. Comparando com as soluções com fio, estas redes apresentam uma implementação mais fácil e com maior flexibilidade e escalabilidade dos dispositivos.

Existem muitos desafios perante este tipo de redes, sendo que poder-se-á considerar o principal problema como a implementação de um sistema que tenha um baixo consumo de energia. Outros desafios são a própria gestão, a robustez e a comunicação. Pretende-se que um nó sensor seja autónomo, sem qualquer tipo de intervenção humana durante um tempo considerável e sem que isso prejudique a fiabilidade dos resultados a longo prazo, numa rede de comunicação projetada para que não aconteçam falhas de transmissão e de receção.

Existe uma enorme variedade de aplicações onde estas redes podem ser utilizadas. Exemplos como a implementação de segurança em áreas vastas, a monitorização ambiental, em habitats de animais, em desastres naturais e em fogos florestais, a monitorização de construções, a monitorização de tráfego automóvel, a monitorização da qualidade do ar e a monitorização de pacientes são apenas algumas aplicações possíveis com RSSF.

A monitorização habitacional através de nós sensores sem fios está diretamente relacionada com a definição de domótica, ou o conceito de "casa inteligente". Poder-se-á definir um sistema domótico como um número de subsistemas com inúmeros dispositivos eletrónicos, interligados entre si através de uma rede de comunicação, com o principal propósito de providenciar segurança, comunicação com o utilizador, gestão do consumo de energia e conforto [1]. Com a gestão do consumo de energia pretende-se reduzir o excesso de consumo, sendo realizada a monitorização e análise de determinados resultados de aparelhos de uso corrente nas habitações. Uma adequada monitorização levará à descoberta de eventuais excessos ou anomalias que podem passar despercebidos.

Ter uma boa qualidade do ar em espaços fechados é crucial pois passa-se cerca de 90% do tempo em locais fechados, tais como em casa, no trabalho ou na escola. Sem quaisquer tipos de dispositivos que permitam a monitorização ambiental interior torna-se difícil determinar a qualidade do ar e quais os principais poluentes. Uma má qualidade do ar poderá causar sintomas como dores de cabeça, cansaço, tosse, irritação nos olhos, na garganta ou no nariz, entre outros [2]. Em 2012, a Organização Mundial de Saúde

relacionou cerca de 4,3 milhões de mortes à falta de qualidade do ar em habitações em países em desenvolvimento, nomeadamente no sudeste Asiático e no Pacífico Ocidental [3]. É importante referir que a poluição do ar interior está classificada, em 9.º lugar, no ranking de carga global do risco de doenças [4].

1.2. Objetivo

Este trabalho teve como principal objetivo o desenvolvimento de uma rede de nós sensores sem fios para monitorização habitacional. Para isso, foi necessário efetuar uma pesquisa sobre as problemáticas que se pretendem identificar neste contexto, mais concretamente, a qualidade do ar interior e o consumo energético. A pesquisa consistiu em verificar quais os trabalhos que já foram desenvolvidos neste âmbito, como foram executados, e quais os produtos que existem atualmente no mercado.

Depois, desenvolveu-se uma rede de nós sensores sem fios para uma habitação, sendo implementados cinco protótipos, com objetivos diferentes, e devidamente testados. Os dados foram enviados, guardados e estavam acessíveis em tempo real através de uma plataforma *web* adaptada a este projeto.

Por fim, os resultados destes testes foram analisados e comparados com a informação adquirida na pesquisa inicial.

1.3. Estrutura do Relatório

Capítulo 1 - Introdução - neste capítulo é apresentada a motivação e os objetivos para a elaboração deste projeto.

Capítulo 2 - Estado da Arte - neste capítulo são apresentados os trabalhos que já foram desenvolvidos neste âmbito, como foram executados, e quais os produtos que existem atualmente no mercado.

Capítulo 3 - Desenvolvimento do sistema - neste capítulo são apresentados os requisitos do sistema, a diferenciação entre os tipos de módulos existentes e a explicação como cada um dos módulos foi desenvolvido até ao seu funcionamento.

Capítulo 4 - Testes ao sistema e análise dos resultados - neste capítulo serão apresentados os resultados dos testes efetuados ao sistema.

Capítulo 5 - Conclusões finais - neste capítulo são apresentadas as conclusões finais e as propostas para trabalhos futuros.

2. Estado da arte

Neste capítulo são apresentados os trabalhos que já foram desenvolvidos neste âmbito, como foram executados, e quais os produtos que atualmente existem no mercado.

2.1. Parâmetros a medir numa habitação

Segundo o Guia Técnico da Qualidade do Ar em Espaços Interiores, elaborado em 2009, da autoria da Agência Portuguesa do Ambiente [5], os fatores que afetam a qualidade do ar interior são os que estão na Tabela 2.1.

Tabela 2.1 - Fatores e fontes que afetam a qualidade do ar interior e o conforto [5] .

Fator	Fonte
Temperatura e valores extremos de humidade	Colocação imprópria dos dispositivos de medição (termostatos), deficiente controlo de humidade, incapacidade do edifício de compensar extremos climáticos, número de equipamentos instalados e densidade de ocupação.
Dióxido de carbono	Número de pessoas, queima de combustíveis fosseis, (gás, aquecedores, etc.).
Monóxido de carbono	Emissões de veículos (garagens, entradas de ar), combustão, fumo do tabaco.
Formaldeído	Madeira prensada, contraplacado não selado, isolamento de espuma de ureia, tecidos, cola, carpetes, mobiliário, papel químico.
Partículas	Fumo, entradas de ar, papel, isolamento de tubagens, resíduos de água, carpetes, filtros de aquecimento, ventilação e ar condicionado (AVAC) e limpezas.
Compostos Orgânicos Voláteis (COV)	Fotocopiadoras e impressoras, computadores, carpetes, mobiliário, produtos de limpeza, fumo tintas, adesivos, calafetagem, perfumes, laca e solventes.
Ventilação inadequada (ar exterior insuficiente, deficiente circulação)	Medidas de poupança de energia e manutenção, má conceção do projeto do sistema de AVAC, alteração do sistema de funcionamento do AVAC pelos ocupantes, conceção desajustada dos espaços em avaliação.
Matéria microbiana	Água estagnada em sistemas de AVAC, materiais molhados e húmidos, desumidificadores, condensadores das torres de arrefecimento e torres de refrigeração.

O dióxido de carbono (CO_2) é um gás inodoro, incolor e não inflamável e a sua principal origem resulta dos processos metabólicos, que variam conforme a quantidade de pessoas num espaço, das respetivas estruturas físicas e da atividade física das mesmas. A quantidade de CO_2 é um indicador da qualidade do ar interior, e em Portugal o limite máximo admissível de concentração desta substância é de 1250 ppm [6].

A humidade é um importante parâmetro a ser medido pois o seu excesso provocará o crescimento de microrganismos, tais como fungos e bactérias, de esporos, de células, de

fragmentos e de compostos orgânicos voláteis (COV), além de contribuir também para a degradação de diversos materiais. O excesso de humidade tem sido um dos principais fatores de risco para casos de asma, de tosse e de dificuldades respiratórias. A humidade e a temperatura são dois parâmetros importantes para o conforto térmico dentro de um espaço fechado [5].

As partículas em suspensão podem ter origem em fontes de combustão, de indústria ou mesmo em fontes naturais, tendo diferentes composições químicas e físicas e diferentes tamanhos. No entanto, as partículas em suspensão que representam um risco para a saúde são aquelas que são derivadas de atividades humanas, principalmente as partículas com diâmetro inferior a 10 µm (PM_{10}), pois têm a capacidade de penetrar nas vias respiratórias, contribuindo para a diminuição do nível de saúde humano, afetando com maior gravidade as crianças, os idosos e as pessoas asmáticas. As partículas com o diâmetro inferior a 2,5 µm ($PM_{2,5}$) conseguem penetrar nos brônquios e nos pulmões [7].

O monóxido de carbono (CO) é um gás perigoso, incolor, não irritante, inodoro, insípido e tóxico. Em espaços fechados, o monóxido de carbono é produzido através de fumo de cigarro ou por fontes de combustão, tais como no ato de cozinhar ou através do uso de equipamentos de aquecimento. Outra fonte provável de emissão de monóxido de carbono é a instalação incorreta ou a desapropriada manutenção de equipamentos de cozinha ou de aquecimento. A exposição prolongada leva a uma limitação da tolerância ao exercício e aumento de sintomas de doença cardíaca isquémica [8].

O formaldeído (metanal) (CH_2O) é um gás incolor. Fontes em espaços fechados estão presentes no fumo do tabaco, em aparelhos de aquecimento, na confeção de alimentos, em velas ou na queima de incenso. Também pode estar presente em produtos para o tratamento de madeiras (móveis, armários, etc.), pinturas, colas, produtos de limpeza de casa, equipamentos eletrónicos, entre outros. Exposição prolongada da substância poderá causar alguns efeitos como: tosse, irritação nos olhos e nariz e náuseas [8].

Os COV são gases emitidos por determinados sólidos e líquidos. São constituídos por uma variedade de químicos (formaldeído, benzeno, percloroetileno, etc.) que, a curto e a longo prazo, são prejudiciais para a saúde, sendo que as concentrações de COV em espaços fechados são muito maiores do que em espaços abertos [2]. Segundo o Guia Técnico [5], "O termo Composto Orgânico cobre todos os compostos químicos que contêm átomos de carbono e de hidrogénio. Os compostos orgânicos voláteis são os compostos orgânicos que têm pontos de ebulição aproximadamente na gama de 50-250°C" e existem vários milhares de químicos, sintéticos e naturais, que podem ser chamados de COV. Exposição prolongada aos COV poderá provocar os seguintes sintomas: irritação conjuntival, desconforto na garganta e nasal, dores de cabeça, fadiga e entre outras [2].

A matéria microbiana é um tipo de poluição originado em diferentes espécies de bactérias, fungos e/ou bolores, que normalmente cresce em ambientes fechados e não deve ser minimizada.

Todos estes fatores influenciam a qualidade do ar em espaços fechados, tais como ambientes públicos, zonas de estações de metro, fábricas, hospitais, parques de estacionamento, ou outros. No entanto, o que se pretende saber são quais os fatores

essenciais a medir numa habitação típica. Em [9], foi realizado um estudo para avaliar a qualidade do ar e do conforto nas habitações portuguesas. Neste estudo foram realizadas medições, em 557 habitações desde norte a sul do país, das seguintes concentrações: dióxido de carbono, monóxido de carbono, partículas em suspensão (PM_{10}), formaldeído, compostos orgânicos voláteis totais, ozono, dióxido de enxofre e dióxido de azoto. Para avaliar o conforto térmico foram efetuadas medições da temperatura do ar e da humidade relativa. Estas medições foram realizadas em dois espaços da habitação, nomeadamente na cozinha e no quarto. Analisando os resultados, destaca-se a importância das elevadas concentrações de dióxido de carbono, das partículas em suspensão, dos compostos orgânicos voláteis, e do conforto térmico. Os outros poluentes apresentaram "resultados inferiores ao limite de deteção dos equipamentos", à exceção do monóxido de carbono que em dois casos excedeu o limite nacional. As conclusões deste estudo indicam que para os parâmetros CO_2, PM_{10} e COV, é necessário um maior acompanhamento ou aprofundamento desta temática em edifícios residenciais.

Segundo [6], o limiar de proteção e a margem de tolerância para os poluentes físico-químicos, para efeitos de fiscalização da qualidade do ar interior em edifícios, está presente na Tabela 2.2. Os limiares de proteção indicados dizem respeito a uma média de 8 horas.

Tabela 2.2 - Limiar de proteção e margem de tolerância para os poluentes físico-químicos [6]. (nota: "A análise de radão é obrigatória em edifícios construídos em zonas graníticas, nomeadamente nos distritos de Braga, Vila Real, Porto, Guarda, Viseu e Castelo Branco")

Poluentes	Unidade	Limiar de Proteção	Margem de tolerância [%]
Partículas em suspensão (fração PM_{10})	$\mu g/m^3$	50	100
Partículas em suspensão (fração $PM_{2,5}$)	$\mu g/m^3$	25	100
Compostos Orgânicos Voláteis Totais (COV)	$\mu g/m^3$	600	100
Monóxido de carbono (CO)	mg/m^3	10	-
	ppmv	9	-
Formaldeído (CH_2O)	$\mu g/m^3$	100	-
	ppmv	0,08	-
Dióxido de carbono (CO_2)	mg/m^3	2250	30
	ppmv	1250	-
Radão	Bq/m^3	400	-

No que diz respeito à temperatura e à humidade, segundo o decreto-lei n.º 243/86 de 20 de agosto que define as condições de temperatura e de humidade de modo a proporcionar bem-estar e a defesa da saúde nos locais de trabalho, é indicado que a temperatura deve oscilar entre os 18 °C e os 22 °C (salvo determinadas condições climatéricas em que poderá atingir os 25 °C). Já a humidade relativa deverá oscilar entre 50% e 70% [11].

A monitorização do consumo energético permite quantificar o uso da energia elétrica, em casa ou num local de trabalho, e, dessa forma, controlar os gastos excessivos e reduzir os custos. Saber a quantidade de energia gasta num aparelho, ou quantos kWh são

consumidos no dia-a-dia, poderá ser muito útil e através da monitorização e do simples controlo do uso energético, poder-se-á reduzir o consumo da energia entre 5% a 15% [10].

2.2. Dispositivos existentes para monitorização da qualidade do ar interior e para monitorização do consumo de energia

Atualmente existem dispositivos no mercado que têm o intuito de monitorizar a qualidade do ar em espaços fechados. Diversos dispositivos começaram a surgir devido aos resultados de estudos que revelam dados preocupantes da qualidade do ar em espaços fechados, quer sejam através de empresas, quer através de projetos pessoais com o intuito de medir a qualidade do ar [12].

Um exemplo de um dispositivo é o *Foobot*, presente na Figura 2.1, com um custo a rondar os 200 $, destacando-se a capacidade de monitorizar em tempo real e mostrar os resultados através de uma aplicação num *smartphone*.

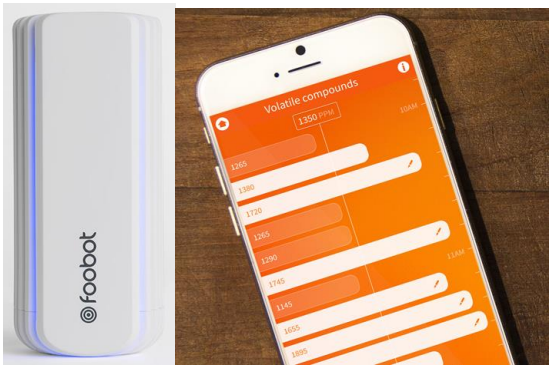


Figura 2.1 - O dispositivo *Foobot* e a aplicação no telemóvel.

O sistema monitoriza a temperatura, a humidade, o dióxido de carbono, as partículas em suspensão e os compostos orgânicos voláteis totais. Tem as dimensões de 17,2 cm por 7,1 cm sendo alimentado por USB através de um adaptador AC-DC de 5 V, 0,5 A. Os dados são enviados por *Wi-Fi* e, são armazenados a cada 5 minutos numa *cloud* e apresentados dados instantâneos a pedido [14].

O *The Birdi* tem sensores de monóxido de carbono, de dióxido de carbono, de partículas em suspensão ($PM_{2,5}$), de temperatura, de humidade e de luz ambiente. Os dados são enviados através de *Wi-Fi* ou *Bluetooth* e o sistema é alimentado através de bateria (duas pilhas AA) que pode ser carregada através de um adaptador para tomada elétrica. Tem um consumo energético inferior a 0,05kWh por mês. A visualização dos dados é feita através de uma aplicação para *smartphone*. O preço do dispositivo ronda os 120$ [15].

O *Netatmo Weather Station* possui dois dispositivos que providenciam parâmetros de qualidade do ar interior e exterior, mas que tem o principal objetivo de medir indicadores de meteorologia. Possui sensores de pressão atmosférica, de dióxido de carbono, de ruído, de temperatura e de humidade e os seus dados são gravados numa *cloud* através de *Wi-Fi*, podendo ser consultados através da aplicação para *smartphone*. Para o módulo para

medição do ar interior, a alimentação é feita através da ligação à tomada elétrica, não tendo qualquer bateria. Para o módulo de medição do ar exterior, este possui uma bateria (três pilhas AAA), tendo uma autonomia de 2 anos. O preço ronda os 170€. É possível adquirir mais módulos, nomeadamente, módulos para medição da pluviosidade e módulos para medição da velocidade do vento [16].

O *Awair* é um dispositivo que tem sensores de partículas em suspensão, de COV, de dióxido de carbono, de humidade e de temperatura. Os dados da qualidade do ar interior poderão ser acedidos através de uma aplicação para *smartphone* e *Bluetooth.* Este dispositivo não possui bateria, funcionando através de ligação à tomada elétrica. O *Awair* ronda os 190€ e a empresa que desenvolveu este sistema pretende fazer com que este se ligue a diversos dispositivos tais como termostatos, ventoinhas, humidificadores, *smart watches*, etc. [17].

O dispositivo *Speck* possui sensores de partículas em suspensão $PM_{2,5}$ (sensor DSM501A), temperatura e humidade. Os dados são obtidos por USB ou por *Wi-Fi* e são armazenados na memória interna (limite máximo de 2 anos com uma amostra por minuto) ou na *cloud* (sem qualquer limite). O intervalo entre amostras pode ser alterado entre 5 segundos e 4 minutos. A visualização dos dados pode ser feita através da aplicação para *smartphone* ou, localmente, através de um *Display Touch*. O dispositivo é alimentado através da ligação à tomada elétrica e não possui qualquer bateria. O *Speck* tem um custo que ronda os 190€ [18].

O *Cubesensors* consiste numa rede de sensores sem fios. Constituído por diversos módulos, cada um pode ser colocado em diferentes espaços de uma casa/escritório para que os mesmos parâmetros sejam medidos em diferentes espaços. Os parâmetros medidos são o dióxido de carbono, a temperatura, a humidade, os COV, o som, a luz e a pressão. Cada módulo comunica para um módulo central e este transmite para a Internet. Utiliza a tecnologia *Zigbee* para a comunicação entre o módulo central e os restantes módulos. A alimentação é de 5 V com um máximo de 1500 mA para o módulo central, e 5 V com um máximo de 500 mA para os restantes módulos (cada módulo tem baterias). Os dados são acedidos através de *smartphone*, utilizando a aplicação dedicada com ligação à internet (os dados são armazenados numa *cloud*). O conjunto de um módulo central e 6 módulos individuais tem o custo de 455€ [19].

O dispositivo *Elgato Eve Room* possui sensores de dióxido de carbono, de COV, de temperatura e de humidade. A ligação ao dispositivo é feita através de *Bluetooth* e os dados são visualizados através de uma aplicação para *Iphone*. Possui uma bateria (três pilhas AA) e o seu preço ronda os 80€. Podem ser adicionados mais módulos à mesma rede, incluindo diferentes dispositivos como o *Eve energy* (monitoriza corrente elétrica), *Eve weather* (para o exterior: temperatura, humidade e pressão), *Eve Thermo* (Regulador de temperatura do aquecedor), *Eve door & window* (monitoriza se as portas/janelas estão fechadas) [20].

Diversos dispositivos existem no mercado com o principal objetivo de monitorizar a energia consumida. Um sistema típico de monitorização de energia inclui sensores e armazenamento de dados. Os sensores mais utilizados são os transdutores de corrente que, através do campo magnético criado em volta do fio, geram uma tensão proporcional à corrente que flui nesse fio e essa tensão é medida, tal como ilustrado na Figura 2.2.

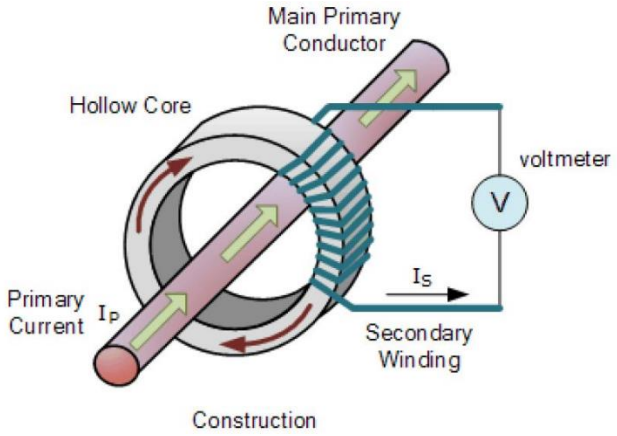


Figura 2.2 - Funcionamento de um transdutor de corrente [21].

Outro tipo de sensor utiliza o efeito de *Hall*. A Figura 2.3 ilustra o princípio básico do efeito de *Hall*.

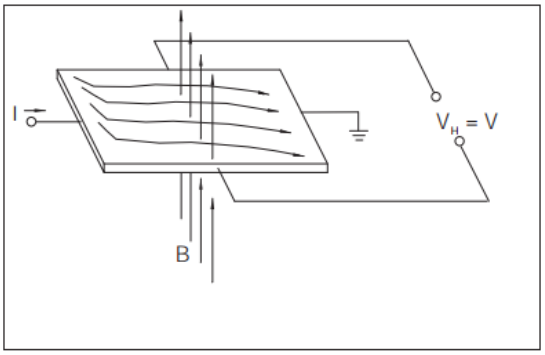


Figura 2.3 - Princípio básico do efeito de Hall [22] .

Na Figura 2.3 é possível ver uma zona fina de material semicondutor, na qual passa corrente e, perpendicularmente à corrente, está presente uma força magnética que exerce uma força na corrente (força de *Lorentz*). Esta força altera a distribuição de corrente, criando uma tensão, que será proporcional à corrente e ao campo magnético.

Ambos os tipos de sensores podem, através de um microcontrolador, comunicar os dados para, por exemplo, um *display*, ou para a Internet e a informação a ser mostrada ao utilizador baseia-se no consumo de corrente num determinando intervalo de tempo e o custo monetário a que esse consumo levou.

O *TED Pro Home*, apresentado na Figura 2.4, é um exemplo de um sistema de monitorização de consumo energético.

Figura 2.4 - *TED Pro Home*.

É constituído por diversos aparelhos de medição que funcionam como uma "pinça" e são ligados nos principais condutores do quadro elétrico. Mede a corrente, a tensão e o fator de potência. Consegue medir até 400 A. A informação das medições é enviada pela rede elétrica, os dados são recolhidos num recetor que estará numa tomada elétrica e, depois, são processados num *software* próprio que tem a capacidade de guardar até 10 anos de informação. Também dá a possibilidade de enviar a informação para aplicações para o *smartphone* ou enviar alertas SMS. O custo deste sistema ronda os 280€ [23].

O *El Gato Eve Energy*, na Figura 2.5, funciona através da colocação do dispositivo na tomada elétrica e, depois, os equipamentos são ligados através deste.

Figura 2.5 - El Gato Eve Energy.

Tem a funcionalidade de ligar/desligar através de uma aplicação para *IPhone,* e os dados são enviados através de *Bluetooth*. Tem a capacidade de medir até 11 A / 2500 W e o seu custo é de cerca de 50€ [24].

Outro exemplo é o dispositivo *Sense* que funciona de maneira diferente ao exemplo anterior. O dispositivo fica colocado no quadro elétrico e a monitorização do consumo é efetuada através de "pinças" (Figura 2.6 b)).

a) b)

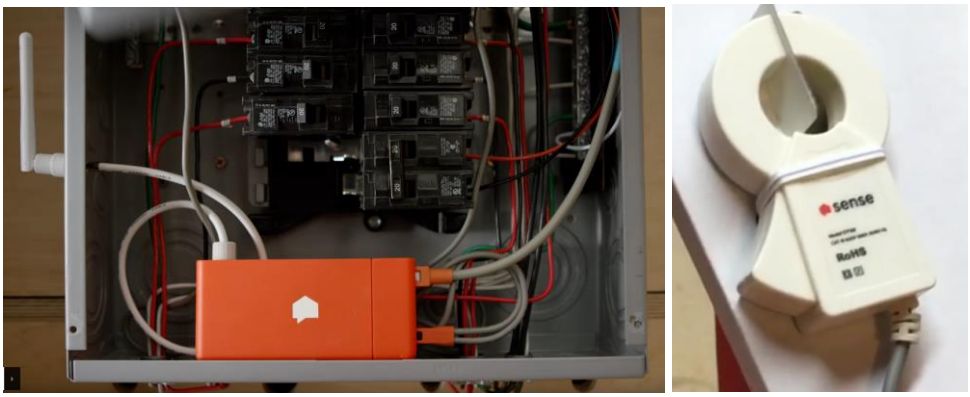


Figura 2.6 - Sense: a) dispositivo colocado no quadro elétrico; b) "pinça".

Os dados são enviados por *Wi-Fi* ou por *Bluetooth* e a visualização é feita através de uma aplicação para *smartphone. A* aplicação reconhece, após algum tempo, cada dispositivo através da "assinatura" do seu consumo. Tem um consumo inferior a 5 W e o preço ronda os 230€ [25].

O *Kill A Watt Meter* é um dispositivo que é colocado na tomada elétrica e os equipamentos são ligados a este dispositivo. Providencia o valor da tensão (85~125 V_{RMS}), da corrente (0~15 A_{RMS}), da frequência (47~63 Hz), do fator de potência (0,00~1,00), da potência ativa (0~1875 W) e da potência aparente (0~1875 VA). Os dados apenas são mostrados no *display* do dispositivo e o seu custo ronda os 25€ [26].

2.3. Protótipos e trabalhos desenvolvidos para monitorização da qualidade do ar interior e para monitorização do consumo energético

O desenvolvimento de uma rede de sensores sem fios é motivado pelo baixo custo em relação a uma rede de sensores com fios, que implica grandes custos de instalação e de cablagem. O conceito de sensores sem fios é de fácil compreensão, mas muitos desafios estarão perante o desenvolvimento destes sistemas.

O *Zigbee* é uma tecnologia de comunicação sem fios para distâncias curtas, de baixo consumo, de baixa taxa de transmissão e de baixo custo. A sua funcionalidade tem sido estudada em diversos trabalhos apresentados em artigos, como o exemplo apresentado em [27]. Nesse artigo é avaliado o desempenho do sinal numa rede de sensores *ZigBee* nas diversas divisões de uma casa, tendo-se chegado à conclusão de que existem diversos fatores que influenciam o sinal, tais como as distâncias entre os nós, as paredes, as portas abertas ou fechadas e as divisões de andares. Estas são variáveis que será necessário ter em conta na instalação de qualquer rede. Tendo em consideração que os sistemas de comunicação *Wi-Fi*, *ZigBee* e *Bluetooth* operam à mesma frequência, é necessário considerar interferências entre sistemas.

As redes de sensores sem fios têm estado envolvidas em diferentes aplicações, nas quais está incluída a monitorização de diversos fenómenos ambientais. A utilização de sensores de temperatura, de humidade, de gás, entre outros, juntamente com um microcontrolador, em que a comunicação é estabelecida através do módulo de transmissão *XBee*, é uma opção económica e viável, como mostram diversos trabalhos [28], [29].

Muitos trabalhos também têm surgido com o principal intuito de dar a conhecer ao utilizador o consumo a nível energético, sendo o grande desafio de um sistema, que meça a energia consumida, desenvolver um dispositivo fiável, económico e de baixo consumo.

2.3.1. Um sistema de rede de sensores sem fios económico, para aplicações de monitorização da qualidade do ar interior

Em [29] tem-se um exemplo de uma rede de sensores sem fios para monitorizar o ar em espaços fechados, em que se utilizam sensores de gás, módulos *XBee* e o Arduíno Uno. A transmissão entre a estação base e os nós foi feita com recurso a módulos *XBee* com antena incorporada. A alimentação passou por ligar uma tensão de 5 V ao microcontrolador que, posteriormente, alimentava todos os sensores e o módulo *XBee*. A arquitetura utilizada neste trabalho está ilustrada na Figura 2.7.

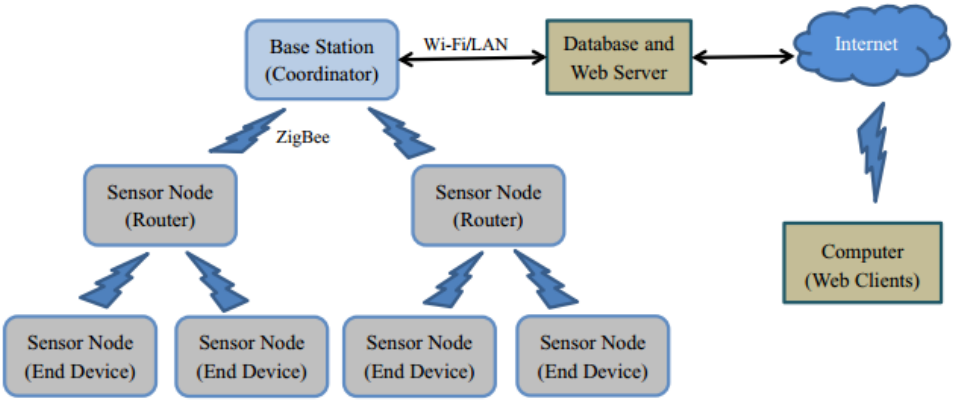


Figura 2.7 - Arquitetura do sistema [29].

Neste trabalho foram medidos os seguintes parâmetros: dióxido de carbono com o sensor MG-811, monóxido de carbono com o sensor MQ-7, temperatura e humidade com o sensor DHT-22, ozono com o sensor MQ131 e COV com o sensor TGS2602. Neste artigo foram executados alguns testes e a respetiva comparação com um sistema de medição da qualidade do ar de qualidade profissional. A Figura 2.8 apresenta uma comparação entre os resultados obtidos pelo sistema e os obtidos por um dispositivo comercial.

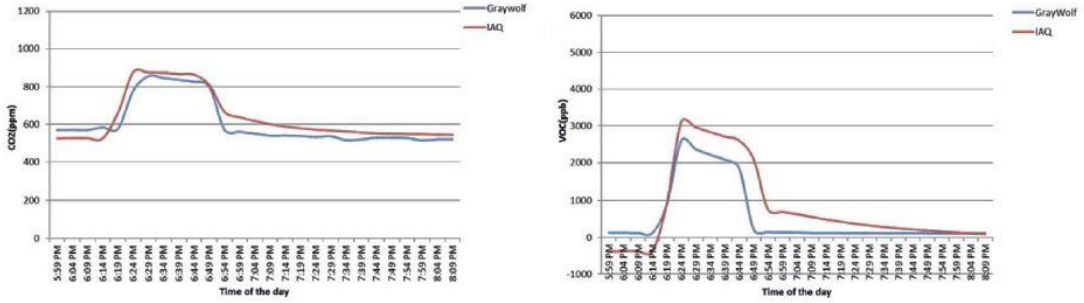


Figura 2.8 - Comparação de resultados entre o protótipo desenvolvido (vermelho) e o dispositivo comercial (azul) [29].

2.3.2. Dispositivo eletrónico para monitorização interior

Em [30] foi desenvolvida uma rede *Zigbee* com um nó a transmitir para uma base de dados *MySQL*. A alimentação deste sensor foi feita através da ligação a uma tomada elétrica, que depois convertia para as tensões de 5 V e 3,3 V. Na Figura 2.9 vê-se o diagrama do nó sensor e do módulo, com a respetiva antena utilizada.

a) b)

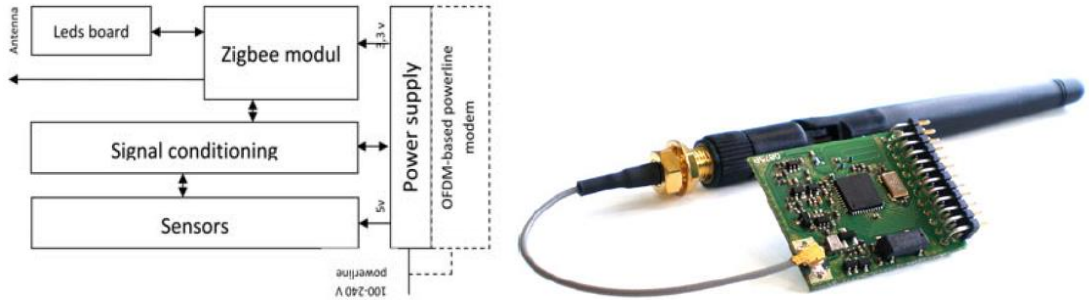


Figura 2.9 – Sistema: a) diagrama de blocos do nó sensor; b) módulo *Zigbee* XTR-ZB1-xHE da *Aurel* [30].

O módulo *Zigbee* utilizado foi o XTR-ZB1-xHE da *Aurel* que, segundo as especificações, tem um alcance de 300 metros em espaço aberto e um consumo de potência de 350 mW. Na Figura 2.10 apresentam-se os resultados dos testes. É importante destacar que os valores de CO_2 (a azul) e CO (a verde) são muito superiores no caso do *hall* de entrada, perto de uma estrada movimentada, em comparação com a janela de um escritório no quarto andar, comprovando que a qualidade do ar naquele espaço é melhor.

a) b)

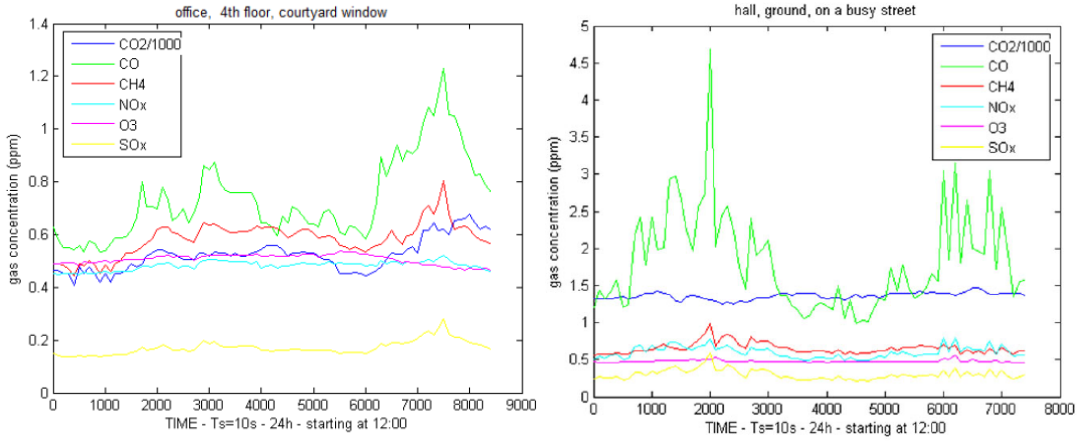


Figura 2.10 - Resultados dos testes. a): medição dos parâmetros da qualidade do ar numa janela de um escritório do 4.º andar; b): medição dos parâmetros no *hall* de entrada, em frente a uma estrada com muito movimento [30].

Os parâmetros de qualidade de ar interior medidos foram os mesmos parâmetros, com os mesmos sensores, que o trabalho anterior, à exceção da temperatura que foi medida com o sensor LM335, e a humidade que foi medida com o sensor HIH-4000.

2.3.3. Aplicação da tecnologia *Zigbee* para monitorização de variáveis ambientais em estufas

Em [31] foi descrita uma aplicação de redes sensores sem fios, baseada numa rede *Zigbee*, em que o principal objetivo passava por monitorizar variáveis ambientais. O sistema permitia ligar diversos dispositivos sem fios em que esses transmitiam dados de temperatura e de humidade relativa. Na Figura 2.11 apresenta-se a arquitetura do sistema e a constituição do módulo.

a) b)

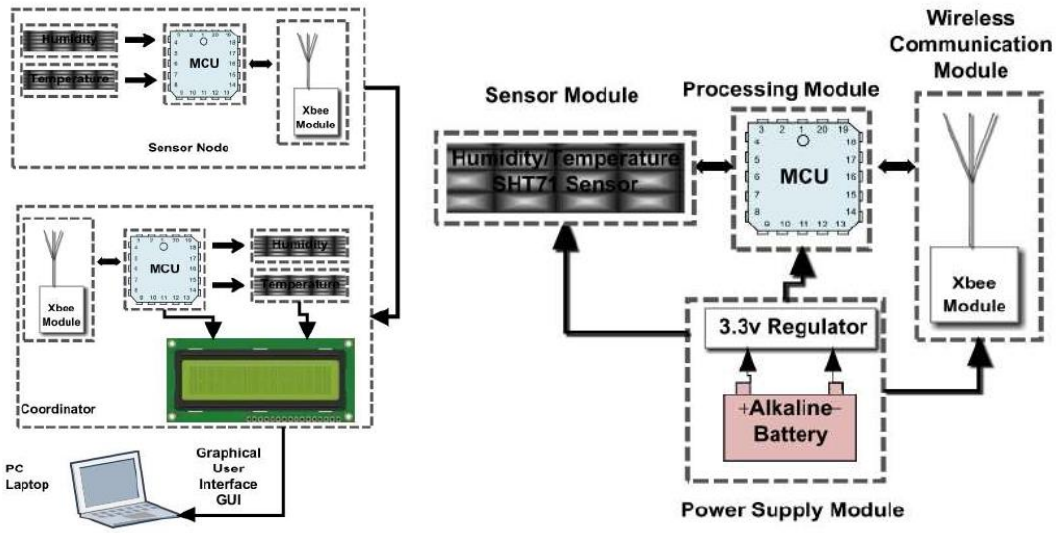


Figura 2.11 - a): Diagrama geral do sistema; b): Diagrama de um nó [31].

O sensor utilizado para medir a temperatura e a humidade em cada nó foi o SHT71. Esses dados eram transmitidos para um coordenador que guardava os dados num cartão e coloca-os, também, num *display*. Os nós sensores eram alimentados através de uma bateria de 9 V enquanto o coordenador era alimentado através da ligação de uma porta USB.

2.3.4. *AirSense*: dispositivo portátil de monitorização da qualidade do ar

Em [32] o principal objetivo foi conceber um dispositivo portátil que medisse a qualidade do ar. Este dispositivo não comunicava com outros dispositivos, mas guardava os dados num cartão SD. Na Figura 2.12 apresenta-se o diagrama de blocos do protótipo desenvolvido.

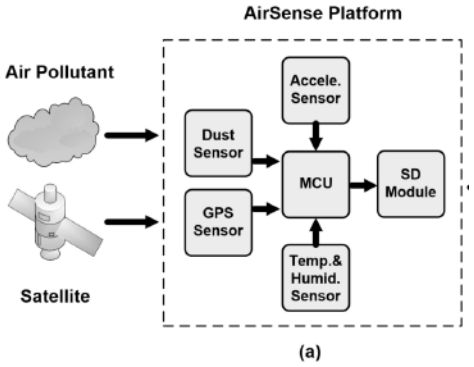


Figura 2.12 - Diagrama de blocos do *Air Sense* [32].

Foram utilizados sensores de GPS e de aceleração devido ao facto deste dispositivo ser portátil e interessar saber os locais das medições. Na Figura 2.13 veem-se os resultados de um teste realizado numa cozinha, antes, durante e depois da confeção de alimentos.

Observa-se, claramente, um aumento de partículas em suspensão ($PM_{2,5}$) quando algo é cozinhado.

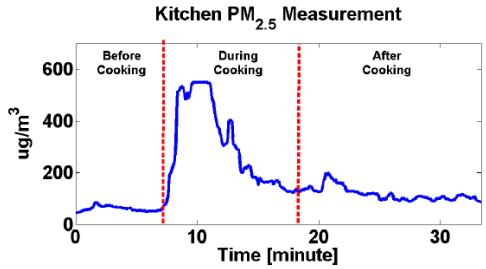


Figura 2.13 - Testes realizados numa cozinha com o protótipo desenvolvido [32].

Os parâmetros de temperatura e de humidade relativa foram medidos através do sensor SHT15 e a quantidade de partículas em suspensão, com o sensor Sharp GP2Y1010AU0F.

2.3.5. PANDAs: dispositivo portátil de baixo custo para aquisição de dados nefelométricos

No trabalho apresentado em [33], que consistiu no desenvolvimento de um sistema de monitorização de partículas em suspensão, foi utilizado um sensor de partículas em suspensão ($PM_{2,5}$), o *Shinyei* PPD42NS. Em [34], este sistema foi desenvolvido, testado e comparado com dispositivos comerciais, cujos resultados podem-se verificar na Figura 2.14.

Figura 2.14 - Testes realizados por 3 *Shinyei* PPD42NS (em baixo). Em cima, os dados são obtidos através de dispositivos comerciais [33].

2.3.6. Medidor de potência sem fios baseado em Arduíno

Em [35] foi utilizado um Arduíno que, através de um sensor de corrente, conseguia obter os valores da corrente e transmiti-los, por *Wi-Fi*, para uma estação-base. O sensor utilizado foi um transdutor de corrente CR3110 com um retificador. A saída desse retificador foi ligada ao Arduíno, a uma porta analógica, que fazia a leitura dos dados. Na Figura 2.15 apresenta-se a arquitetura desse sistema.

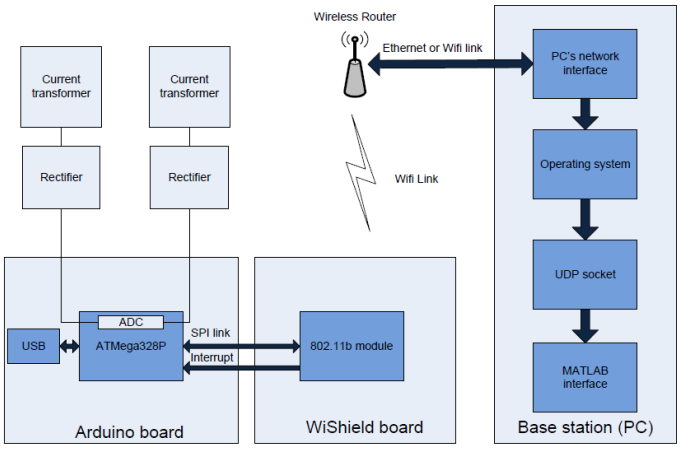


Figura 2.15 - Arquitetura do sistema [35].

Na Figura 2.16 apresenta-se um teste efetuado nesse trabalho, com os resultados da monitorização do consumo de um forno em comparação com os resultados obtidos de um dispositivo comercial de medição de corrente.

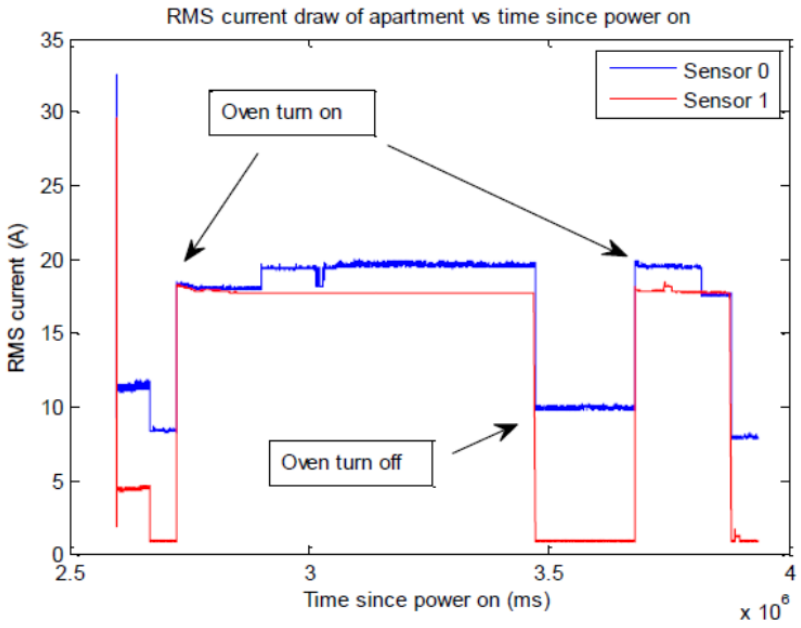


Figura 2.16 - Teste efetuado com a monitorização do consumo de um forno [35].

2.3.7. Projeto e construção de um sistema de monitorização de energia elétrica para uma habitação

Numa dissertação apresentada em 2010 [36], foi pretendido conceber um sistema que monitorizasse um sinal de corrente e enviasse os dados para uma base, neste caso um computador, através de um sistema sem fios. Na Figura 2.17 apresenta-se o diagrama desse trabalho.

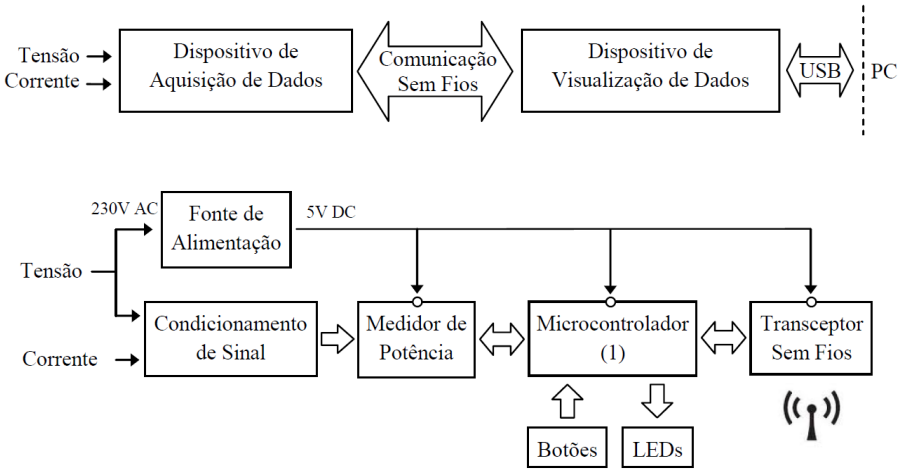


Figura 2.17 - Diagrama do sistema desenvolvido para o trabalho [36].

Os dados eram apresentados através de um visor LCD. O dispositivo media a corrente da mesma forma que o exemplo anterior (através do transdutor de corrente CR3110) mas também usou um medidor de potência (CS5463), para medir a potência e o fator de potência. O envio dos dados foi efetuado através do transcetor sem fios ER400TRS.

2.3.8. Energino

O *Energino* [37] é um trabalho que consistiu em criar um *shield* para o Arduíno e, através deste, monitorizar a corrente. Na Figura 2.18 apresenta-se o diagrama deste dispositivo.

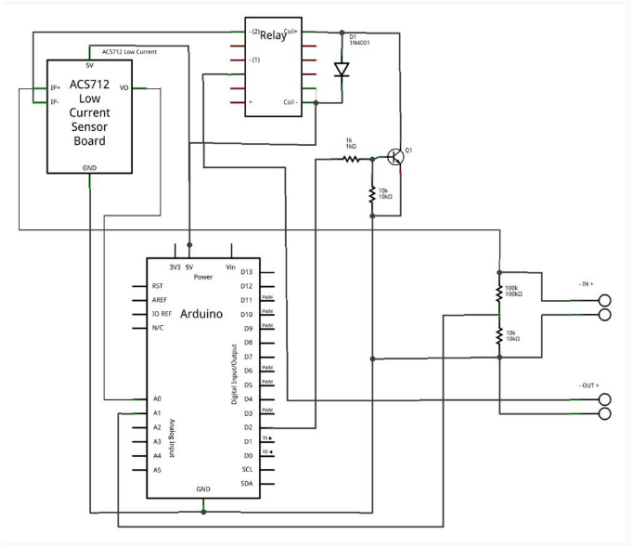


Figura 2.18 - Diagrama do Energino [37].

A combinação deste *shield* com um outro, de *Ethernet*, dá a possibilidade de disponibilizar os dados medidos numa página *web*. A leitura foi feita pelo sensor de corrente ACS712, sendo um sensor que, através do efeito de *Hall*, media a corrente. A grande diferença deste sensor em comparação com o anterior é que neste, há necessidade de interromper a ligação, o que pode não ser muito difícil quando se pretende medir o consumo de uma simples tomada elétrica, mas que se torna muito mais complicado

quando se pretender instalar um sensor num quadro elétrico. Ainda sobre o *Energino*, utiliza também um relé, para ligar e desligar as ligações, dando a possibilidade de executar essa função remotamente.

2.3.9. Desenvolvimento de um sistema de monitorização remota do consumo de energia elétrica e da presença humana em edifícios

Em [38] foi desenvolvido um sistema que tinha a finalidade de monitorizar remotamente o consumo de energia e a presença de pessoas num edifício. A estrutura do trabalho está ilustrada na Figura 2.19.

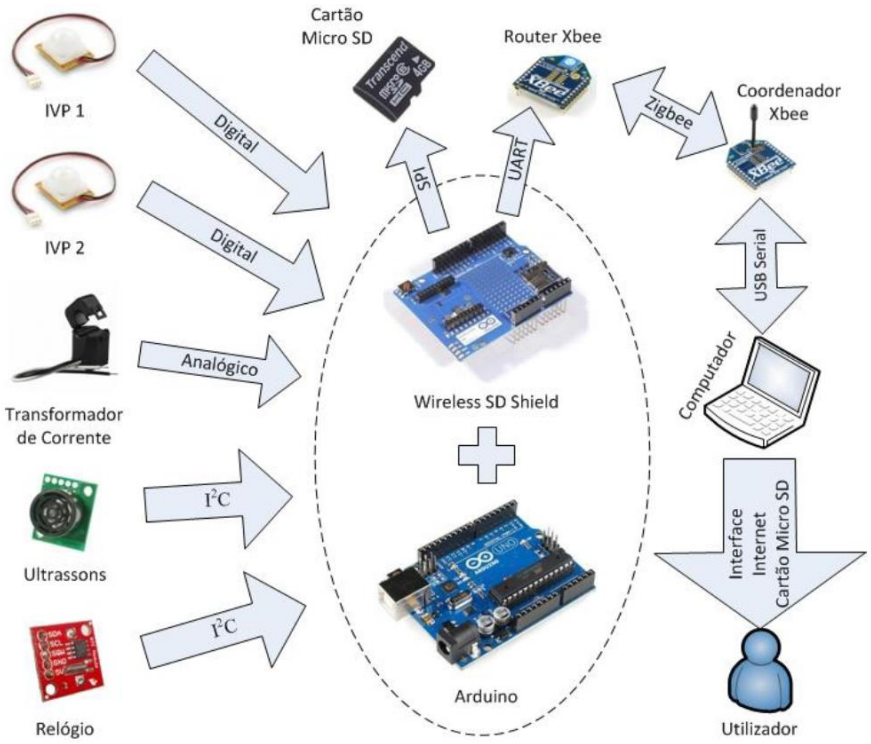


Figura 2.19 - Estrutura do sistema [38].

Foi utilizado um Arduíno com um *XBee*, em modo *router*, para enviar os dados para um outro módulo coordenador e, por sua vez, disponibilizava esses dados na internet e permitia a sua visualização através de uma interface. Esses dados foram também gravados num cartão micro SD no *XBee router*. O sensor de corrente utilizado foi o CR3110-3000. Neste nó sensor também foi utilizado um RTC (*Real Time Clock*) para um controlo preciso do tempo. Na Figura 2.20 apresenta-se uma comparação do sistema desenvolvido (a verde) com o sistema comercial CA 8332 da *Chauvin Arnoux* (a azul), durante 24 horas. Os resultados que foram obtidos neste trabalho demonstram uma grande proximidade, do protótipo desenvolvido, com o sistema profissional.

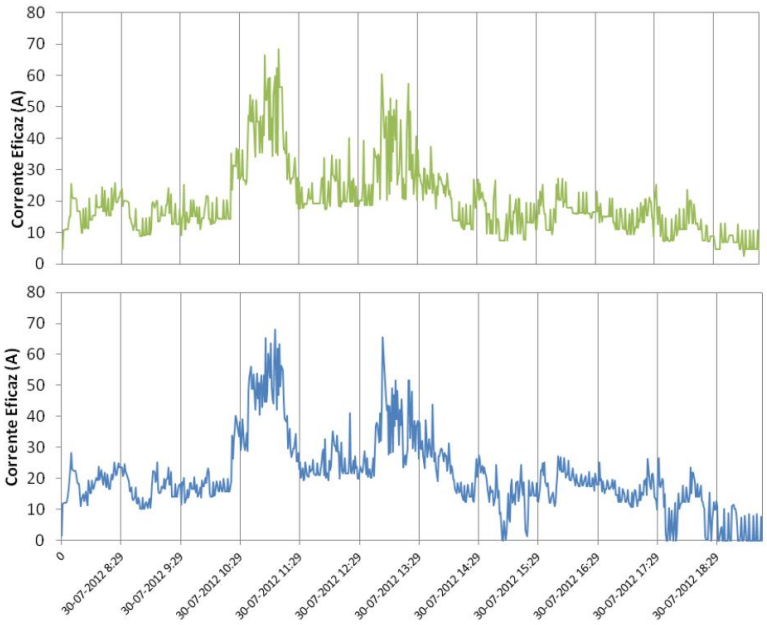


Figura 2.20 - Testes efetuados [38].

2.4. Sensores tipicamente utilizados para monitorização da qualidade do ar e para a monitorização do consumo de energia

Foi feita uma pesquisa de sensores, de baixo custo, para operarem com microcontroladores tais como o Arduíno. As tabelas seguintes apresentam exemplos de sensores que poderão ser utilizados para medir os vários parâmetros de interesse referidos anteriormente. Na Tabela 2.3 apresentam-se exemplos de sensores de temperatura e humidade.

Tabela 2.3 - Exemplos de sensores de temperatura e humidade.

	Gama de medição		Consumo de corrente (mA)	Tensão de funcionamento (V)	Preço (€)
	Temperatura (ºC)	Humidade (%)			
LM335 [39]	-40 a 100 (±6)		1 (máx: 5)	5	3
HIH-4000 [40]		0 a 100 (±3,5)	0,2 (máx: 0,5)	4 a 5,8	19
DHT-22 [41]	-40 a 80 (±0,5)	0 a 100 (±2)	1 (máx: 1,5)	3,3 a 6V	9
DHT-11 [42]	0 a 50 (±2)	20 a 90 (±5)	máx: 2,5	3-5,5 V	5
SHT71 [43]	-40 a + 123,8 (±0,4)	0 a 100 (±3)	0,55 (máx: 1)	2,4 a 5,5 V	30
SHT15 [44]	-40 a + 123,8 (±0,3)	0 a 100 (±2)	0,55 (máx: 1)	2,4 a 5,5 V	20
HTU21D-F [45]	-40 a +125 (±0,3)	0 a 100 (±2)	0,45 (máx: 0,5)	1,5 a 3,6 V	15

A Tabela 2.4 apresenta alguns exemplos de sensores de partículas. Os sensores de partículas obtêm o número de partículas. Segundo [46], para converter para concentração de partículas em µg/m^3, tem-se que

$$Concentração\left(\frac{\mu g}{m^3}\right) = \left(\frac{N.º\ de\ partículas}{0{,}01\ pés^3}\right) \times 3531{,}5 \times (massa\ da\ partícula) \qquad (1.1)$$

em que a constante 3531,5 é o fator multiplicador para converter de 0,01 pés cúbicos para metros cúbicos. A massa da partícula é calculada assumindo que a densidade de cada partícula é de 1,65x10^{12} µg/m^3, que o raio de uma partícula $PM_{2,5}$ é de 0,44 µm e que o raio de uma partícula PM_{10} é de 2,6 µm. Assim,

$$Massa\ da\ partícula\ PM_{10} = \frac{4}{3} \times \pi \times (2{,}6\ \mu m)^3 \times 1{,}65 \times 10^{12}\ \frac{\mu g}{m^3} \qquad (1.2)$$

$$Massa\ da\ partícula\ PM_{2,5} = \frac{4}{3} \times \pi \times (0{,}44\ \mu m)^3 \times 1{,}65 \times 10^{12}\ \frac{\mu g}{m^3} \qquad (1.3)$$

Tabela 2.4 - Exemplos de sensores de partículas.

	Gama de medição	Consumo de corrente (mA)	Tensão de funcionamento (V)	Preço (€)
Shinyei PPD42NS [47]	0~28000 partículas/litro (0~3401µg/m^3) Partículas acima dos 1µm	90	5	15
Sharp GP2Y1010AU0F [48]	0~4100 partículas/litro (0~500µg/m^3)	11 (máx: 20)	5	10
DSM501A [49]	0~11400 partículas/litro (0□1400 µg/m3) Partículas acima dos 1µm	90	5	10

Na Tabela 2.5 apresentam-se alguns exemplos de sensores de dióxido de carbono. É usual apresentar o dióxido de carbono em partes por milhão (ppm) ou por percentagem de concentração. Partes por milhão trata-se de um valor adimensional que representa parte de um número inteiro em unidades de 1/1000000. Por exemplo, 2000 ppm de dióxido de carbono significa que em um milhão de moléculas, 20 000 dessas seriam de dióxido de carbono e as restantes 980 000 seriam de outros gases. Quando as concentrações a medir ultrapassam as 10 000 ppm, os fabricantes utilizam a percentagem de concentração (exemplo: 20 000 ppm = 2%).

Tabela 2.5 - Exemplo de sensores de dióxido de carbono.

	Gama de medição	Consumo de corrente (mA)	Tensão de funcionamento (V)	Preço (€)
MG-811 [50]	0,035 a 1% (350 a 10000 ppm)	200	5	50
GSS C20 [51]	0,02 a 100% (200 a 1000000 ppm)	20	4,7 a 5,5	190
K-30 SE-0018 [52]	0 a 1% (0 - 10000 ppm)	40	5,5 a 14	80
COZIR GC-0016 [53]	0 a 100% (0 a 1000000 ppm)	1.5 (máx:33)	3,2 a 3,4	100

Na Tabela 2.6 apresentam-se alguns exemplos de sensores utilizados para medir compostos orgânicos voláteis.

Tabela 2.6 - Exemplos de sensores de compostos orgânicos voláteis.

	Gama de medição	Consumo de corrente (mA)	Tensão de funcionamento (V)	Preço (€)
Adafruit MiCS5524 [54]	Monóxido de carbono (1 to 1000 ppm), Amoníaco (~ 1 to 500 ppm), Etanol (~ 10 to 500 ppm), Hidrogénio (~ 1 - 1000 ppm), Metano/ Propano/ Isobutano (~ 1,000 ppm). (não diferencia os diferentes gases)	Máx: 32	4,9 a 5,1	15
TGS2602 [55]	1 ~ 30ppm de Etanol	Máx: 56	5	35
iAQ-2000 [56]	350- 2000ppm CO_2 equivalentes CO, CH_4, LPG Álcool	22 a 30	5	215

Na Tabela 2.7 estão alguns exemplos de sensores de corrente. Os sensores de corrente disponíveis no mercado podem ser de dois tipos: invasivos ou não invasivos. Essencialmente, a diferença entre estes tipos de sensores está na necessidade de interromper uma ligação para medir a corrente, sendo esse um sensor invasivo, enquanto no outro caso não é necessário interromper a ligação, sendo não invasivo.

Tabela 2.7- Sensores de corrente.

Sensor	Gama de medição (A)	Invasivo	Preço (€)
CR3110-3000 [57]	Até 75 (AC/DC)	Não	15
ACS712-30 [58]	0,07 a 30 (AC/DC)	Sim	7
CR3111 - 3000 [57]	Até 100 (AC/DC)	Não	15
AttoPilot Voltage and Current Sense Breakout – 45A [59]	Até 45 (DC)	Sim	30

2.5. Conclusão

A monitorização do ar interior é essencial para garantir uma boa qualidade do ar, e a monitorização do consumo de energia contribui para uma consciencialização dos utilizadores sobre o gasto em habitações. As diferentes variáveis a serem medidas podem ser reunidas num único sistema em que, acendendo a uma única plataforma, é possível aceder a todos os parâmetros importantes de uma habitação.

Os equipamentos existentes para monitorização do ar são, na sua maioria, dispositivos únicos que englobam os diversos sensores e os seus dados são transmitidos, por *Wi-Fi* ou *Bluetooth*. Esses dados são guardados numa *cloud* e são acedidos através de uma aplicação. O desenvolvimento de diferentes protótipos, não só com sensores diferentes, mas também com métodos de carregamento diferentes, é fundamental para uma análise mais completa. As variáveis mais importantes a serem medidas, no que diz respeito à qualidade do ar interior, são a temperatura, a humidade relativa do ar, o dióxido de carbono, os compostos orgânicos voláteis e as partículas em suspensão.

A monitorização do consumo elétrico deve abranger uma vasta gama de valores tendo em conta os vários equipamentos utilizados no dia-a-dia, como por exemplo um carregador de telemóvel (30 mA) ou uma chaleira (9 A). Esses valores devem ser transmitidos da mesma forma que os sensores acima referidos, englobando todas essas variáveis num único sistema.

A comunicação entre os diversos nós, à semelhança do sistema apresentado em [22], poderá ser feita recorrendo à tecnologia *ZigBee*. A utilização de *XBee* para uma rede de nós sensores sem fios poderá ser a mais indicada devido à facilidade de utilização.

Tal como verificado anteriormente, os preços dos sistemas de monitorização são elevados e um dos objetivos deste trabalhado também passa por desenvolver um sistema mais económico. Através de uma adequada pesquisa no mercado, é possível desenvolver um protótipo utilizando microcontroladores e sensores de baixo custo e com baixo consumo.

3. Desenvolvimento do sistema

Neste capítulo são apresentados os requisitos do sistema, a diferença entre os tipos de módulos necessários à monitorização e a explicação como cada um dos módulos foi desenvolvido até ao seu funcionamento.

3.1. Requisitos e componentes do sistema

O sistema desenvolvido possuía vários módulos em diferentes pontos de uma habitação. Os dados medidos por cada um dos sensores foram enviados, através de uma rede sem fios, para um módulo coordenador que recebe todos os dados e coloca-os disponíveis para visualização. Na Figura 3.1 apresenta-se o esquema da rede. A vermelho está representado o coordenador, a azul os *routers* e os restantes são os nós terminais

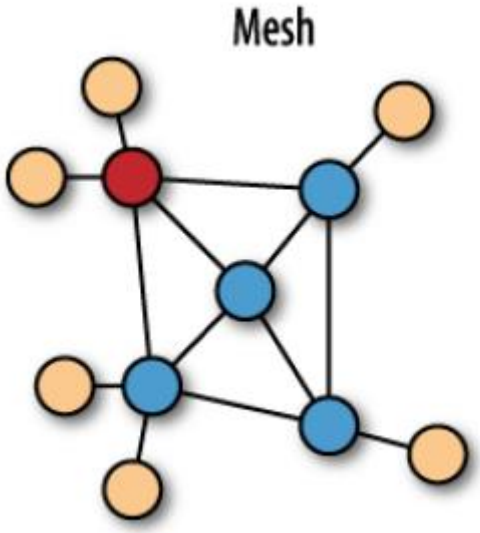


Figura 3.1 - Esquema de rede do tipo *mesh [60]*.

O esquema de rede apresenta uma configuração do tipo *mesh*. Os *routers* passam informação tanto aos nós terminais como a outros *routers* e ao coordenador. O coordenador gere o funcionamento da rede e também tem a função de encaminhar a informação [60]. Desta forma apresentou-se uma rede em que a informação poderá ter várias alternativas de encaminhamento, melhorando a ligação entre os módulos e evitando que os módulos deixem de comunicar por falta de ligação, causada por eventuais obstáculos presentes numa habitação.

O funcionamento de cada um dos módulos foi garantido por um microcontrolador que processava os dados adquiridos através de cada um dos sensores e que posteriormente foram sendo enviados em determinados intervalos de tempo. Os parâmetros a medir foram a temperatura, a humidade, o dióxido de carbono, as partículas em suspensão, os compostos orgânicos voláteis e a corrente elétrica. Os dados obtidos foram enviados para o módulo coordenador, sendo que essa funcionalidade foi garantida através de módulos de comunicação sem fios instalados em cada um dos módulos da rede.

O módulo coordenador foi o dispositivo responsável por receber os dados de todos os sensores, colocá-los numa base de dados e apresentá-los através de uma plataforma de visualização. Na Figura 3.2 está representada a arquitetura da rede com os diferentes módulos que foram executados.

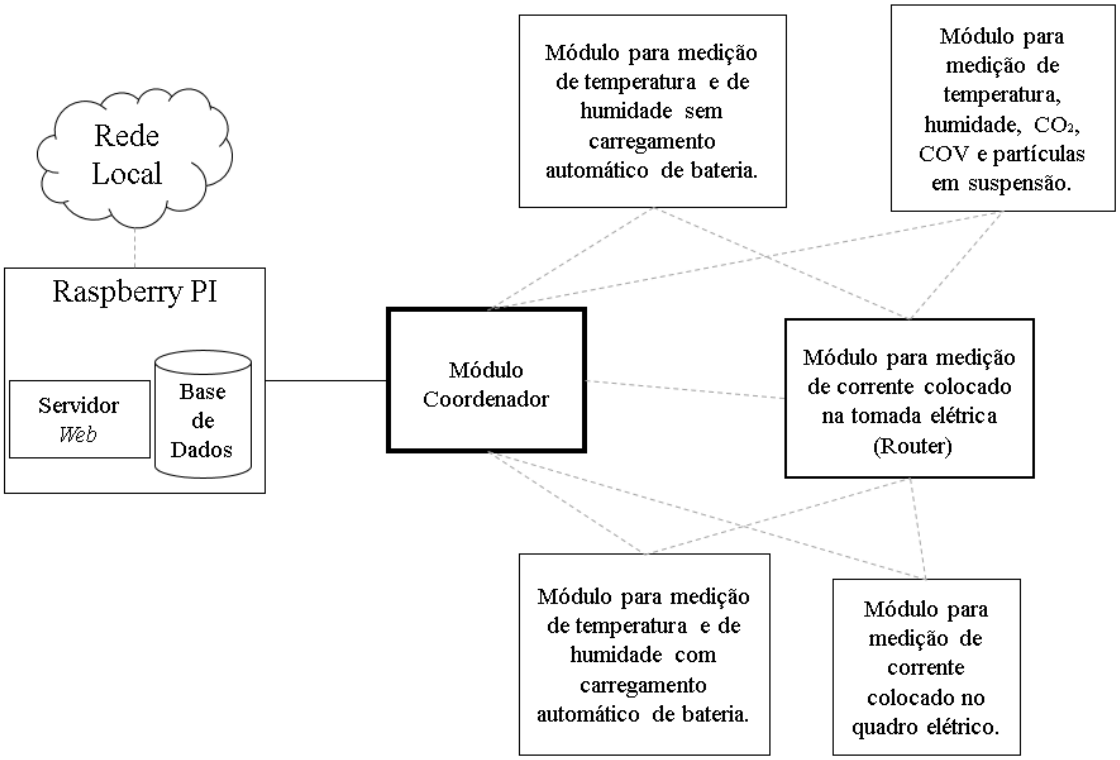


Figura 3.2 - Arquitetura da rede.

3.1.1. Módulos de medição (nós terminais)

Os módulos desenvolvidos para medição dos diversos parâmetros são os apresentados na Tabela 3.1.

Tabela 3.1 - Módulos desenvolvidos.

Dispositivo	Funções
Medição de temperatura e humidade 1	• Enviar dados com um intervalo de amostragem de 10 segundos • Monitorizar a tensão na bateria • Carregar automaticamente a bateria através da tomada elétrica • Monitorizar o RSSI (*Received Signal Strength Indicator* - Indicador de intensidade do sinal recebido)
Medição de temperatura e humidade 2	• Enviar dados com um intervalo de amostragem de 10 segundos • Monitorizar o RSSI • Monitorizar tensão na bateria
Medição de temperatura, humidade, CO_2, partículas em suspensão e COV;	• Enviar dados de temperatura e humidade com um intervalo de amostragem de 10 segundos • Enviar dados de CO_2 com um intervalo de amostragem de 10 segundos • Enviar dados de COV com um intervalo de amostragem de 10 segundos • Enviar dados de partículas em suspensão com um intervalo de amostragem de 30 segundos • Monitorizar o RSSI • Monitorizar a tensão na bateria
Medição da corrente numa tomada elétrica	• Enviar dados da corrente consumida numa tomada elétrica • Enviar dados com um intervalo de amostragem de 10 segundos • Monitorizar o RSSI
Medição da corrente num quadro elétrico.	• Enviar dados da corrente consumida num quadro elétrico • Enviar dados com um intervalo de amostragem de 10 segundos • Monitorizar o RSSI • Monitorizar a tensão na bateria

Cada um dos módulos de medição foi desenvolvido utilizando o Arduíno Fio, representado na Figura 3.3. Na imagem à direita é possível verificar o *socket* para a colocação do *XBee*.

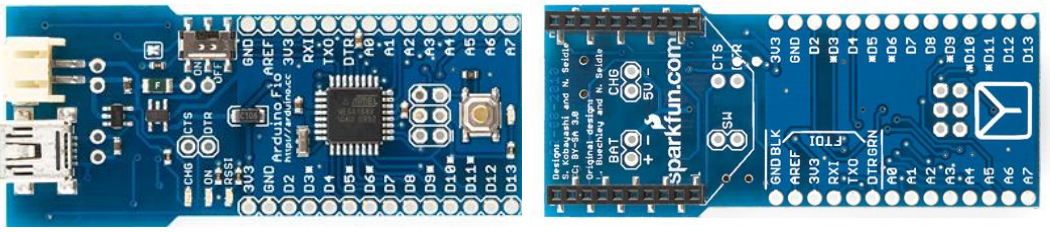


Figura 3.3 - Arduíno Fio.

Este dispositivo inclui um microcontrolador ATmega328P e opera a 3,3 V. Na Tabela 3.2 estão presentes as respetivas características.

Tabela 3.2 – Principais características do Arduíno Fio *[61]*.

Microcontrolador	ATmega328P
Tensão de operação	3,3 V
Tensão de entrada	3,35-12V
Tensão de entrada para carregamento	3,7-7V
Pinos digitais (I/O)	14 (6 providenciam saída PWM (*Pulse Width Modulation* - Modulação da largura de pulso)
Pinos Analógicos	8
Corrente DC nos pinos digitais	40 mA
Memória *Flash*	32 kB (2 kB são usados pelo *bootloader*)
SRAM	2 kB
EEPROM	1 kB
Velocidade de relógio	8 MHz
Largura	28 mm
Comprimento	65 mm
Peso	9 g

As grandes vantagens deste dispositivo são a sua pequena dimensão, conectores preparados para uma bateria de polímeros de lítio com um circuito de carregamento incorporado na placa e um *socket* para a colocação do *XBee*. O principal propósito deste dispositivo é para aplicações sem fios [61]. Para a comunicação serão utilizados os módulos *XBee* [62], Figura 3.4, que possibilitam uma conectividade sem fios de baixo custo entre dispositivos eletrónicos.

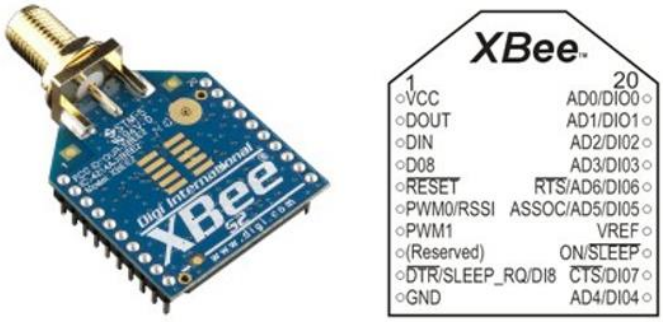


Figura 3.4 - *XBee*. Na imagem à direita apresenta-se os pinos do *XBee*.

Outras das grandes vantagens é o baixo consumo de energia, quando está em modo adormecido, e a possibilidade de estabelecer uma ligação em malha para transmissão dos dados. As suas principais características estão na Tabela 3.3.

Tabela 3.3 – Características do *XBee [62]*.

Tensão de operação	3,3 V
Consumo máximo de corrente	40 mA
Taxa máxima de informação	250 kbps
Pinos digitais (I/O)	8
Pinos ADC de 10 bits	6
Encriptação	128 bit

Para configurar estes módulos foi utilizado o *software* XCTU da *Digi*. O *XBee* tem dois modos de funcionamento: o modo AT e o modo API. O modo AT é o sinónimo para modo "Transparente" e, de forma simples, a comunicação entre nós é realizada de forma imediata, sendo a informação enviada pela porta TX e recebida, no nó destino, pela porta RX. No modo API a informação é enviada em tramas na qual está incluída a informação de destino, permitindo o acesso a mais parâmetros da rede, e sendo mais adequada para redes de maior dimensão. O modo API foi o modo escolhido para este trabalho. Na configuração de cada *XBee*, utilizando o *software* XCTU, foram tidos em atenção os principais parâmetros presentes na Tabela 3.4.

Tabela 3.4 - Principais parâmetros para a configuração do *XBee*.

Parâmetro	Descrição
ID	Identificador da rede. Todos os módulos da mesma rede deverão ter este parâmetro igual.
DH; DL	Endereço de destino. 2 campos de 32 bits de um endereço de 64 bits.
AP	Modo API. Liga o modo API.
AO	Modo de saída dos dados API.
SM	Modo de adormecimento. Caso o módulo seja um *router*, este campo deverá ter um valor de 0. Caso seja um nó terminal deverá ter um valor de 1 (Modo Pin *Hibernate*: o *XBee* é acordado através do acionamento de um pin).
SP	Definir/ler período de adormecimento para nós terminais. Este parâmetro deverá ser configurado nos módulos *router* e coordenador, e deverá corresponder ao maior tempo de amostragem dos nós terminais.
SN	Definir/ler o número de períodos de adormecimento cíclicos utilizados para calcular o tempo limite do nó terminal. Se um nó terminal não enviar uma solicitação de pesquisa para seu coordenador ou *router* dentro do tempo limite da pesquisa, o nó terminal é removido da tabela.

O nó terminal apenas tem a capacidade de enviar a informação para um *router* ou para um coordenador, podendo estar "adormecido" enquanto não envia informação, sendo isto muito útil para um baixo consumo de energia.

3.1.2. Módulos *router*

Os nós *router* têm a capacidade de receber os dados provenientes de outros nós sensores e encaminhá-los para o módulo coordenador. Este nó tem um maior consumo de energia, uma vez que deve estar sempre ligado. O seu funcionamento pode ser garantido com apenas o dispositivo de comunicação *XBee*, sem a necessidade do microcontrolador.

3.1.3. Módulo coordenador

O nó coordenador é composto por um *XBee*, configurado para essa função, e por um microcontrolador ou um computador. A ligação entre os dispositivos é efetuada através de um *XBee Explorer* [63] que é um dispositivo que tem a capacidade de se ligar ao computador ou ao microcontrolador. Para o processamento dessa informação, foi usada uma aplicação apresentada em [64]. Esta aplicação foi desenvolvida para a receção, armazenamento e visualização de toda a informação recebida pelo nó coordenador. Inicialmente tinha-se um programa desenvolvido, para *Windows*, em *Java*, que era responsável pela receção da informação e respetivo armazenamento numa base de dados *MySQL*, através do *software NetBeans* e *Xampp*. Também apresentava uma página Web, que utilizando vários *scripts* em PHP, que disponibilizava a informação ao utilizador. Todos estes programas foram instalados num computador com o ambiente *Windows*, que se trata de um dispositivo de grande consumo, pelo que foi mais conveniente optar por outra alternativa. Para isso, a utilização de um *Raspberry Pi* foi mais apropriada.

O *Raspberry Pi* é um pequeno computador de baixo consumo e de baixo custo, que surgiu para promover o ensino da informática nas escolas. Este dispositivo tem a possibilidade de ligar diversos dispositivos, como rato, teclado e ecrã, e também de executar projetos na área da eletrónica ou robótica. O principal objetivo passou por instalar os programas necessários e utilizar este sistema para receber os dados provenientes dos módulos, sendo neste que foi ligado o *XBee* coordenador. Foi utilizado o *Raspberry Pi 2 Model B* e as suas características estão na Tabela 3.5.

Tabela 3.5 - Características do *Raspberry Pi 2 Model B [65]*.

Processador	*Quad-core ARM Cortex-A7* - 900MHz
Memória RAM	1GB
Portas USB	4
Pinos GPIO	40
Interface Gráfica	Porta Full HDMI
Acesso à Internet	Porta *Ethernet*
Áudio	Porta Áudio *Jack* 3,5mm combinada com *Composite Video*
Memória externa	Suporte para Cartão Micro SD
Processador Gráfico	*VideoCore* IV 3D

Tendo em consideração que em [64], o trabalho foi preparado para utilização no sistema operativo Windows, foi feita uma adaptação para se proceder à instalação dos programas no *Raspberry Pi*. Para isso, foi necessária a instalação e atualização do sistema operativo num cartão SD apropriado para o efeito [66]. Neste trabalho o sistema operativo instalado foi o *Raspbian*. Foi necessário instalar o servidor de *web Apache2*, a base de dados MySQLServer, a linguagem de programação *PHP5*, o módulo MySQL para PHP5, a ferramenta de administração da base de dados *PHPMyAdmin* e ambiente integrado para desenvolvimento de *software NetBeans* (Anexo 1).

3.1.4. Formato dos Dados

Para que a informação seja recebida corretamente, esta deve ser enviada numa trama que o *software* esteja preparado para processar. A construção da trama para os nós sensores é feita no programa do microcontrolador Arduíno Fio e tinha a estrutura representada na Tabela 3.6.

Tabela 3.6 - Estrutura da trama.

trama[] = {0x7E,0x00,0x2E,0x10,0x00,0x00,0x13,0xA2,0x00,0x40,0x2D,0x25,0x36,0xFF,0xFE,0x00,0x00,'?','B','1',0x3D,0x00,0x00,0x00,0x00,0x26,'T','1',0x3D,0x00,0x00,0x00,0x00,0x26,'H','1',0x3D,0x00,0x00,0x00,0x00,0x26,'R','1',0x3D,0x00,0x00,0x00,0x00,0x00};	
0x7E	*Byte* que indica o início da trama;
0x00,0x2E	Comprimento da mensagem em *bytes*;
0x10	*Byte* que identifica que se trata de uma trama que enviará dados.
0x00	*Byte* que trata da numeração da mensagem.
0x00,0x13,0xA2,0x00, 0x40,0x2D,0x25,0x36	Endereço do destinatário (64 bits)
0xFF,0xFE	Endereço do destinatário (16 bits).
0x00	*Byte* que indica o número máximo de saltos de uma mensagem *broadcast*. 0 indica que é o número máximo de saltos.
0x00	Opções de transmissão. 0 indica *unicast*.
'?'	Identificador de início de informação
'B','1'	Trata-se da identificação de um parâmetro de um sensor (Exemplo: identificador para a tensão da bateria de um módulo)
0x3D	Identificador que assinala que os próximos bytes serão um valor correspondente ao sensor.
0x26	Identificador que separa informações de sensores.
0x00	Byte de *checksum*.

O programa em execução no *Raspberry Pi* ligado ao coordenador recebia as tramas que, por sua vez, extraía os dados e colocava-os na base de dados.

3.1.5. Visualização dos dados

Os dados estavam disponíveis para visualização numa página que era acessível a todos que estivessem na mesma rede doméstica que o *Raspberry Pi*. A página *web* apresentava os dados que eram colocados na base de dados. Na Figura 3.5 apresenta-se a página de internet com os dados disponíveis ao utilizador.

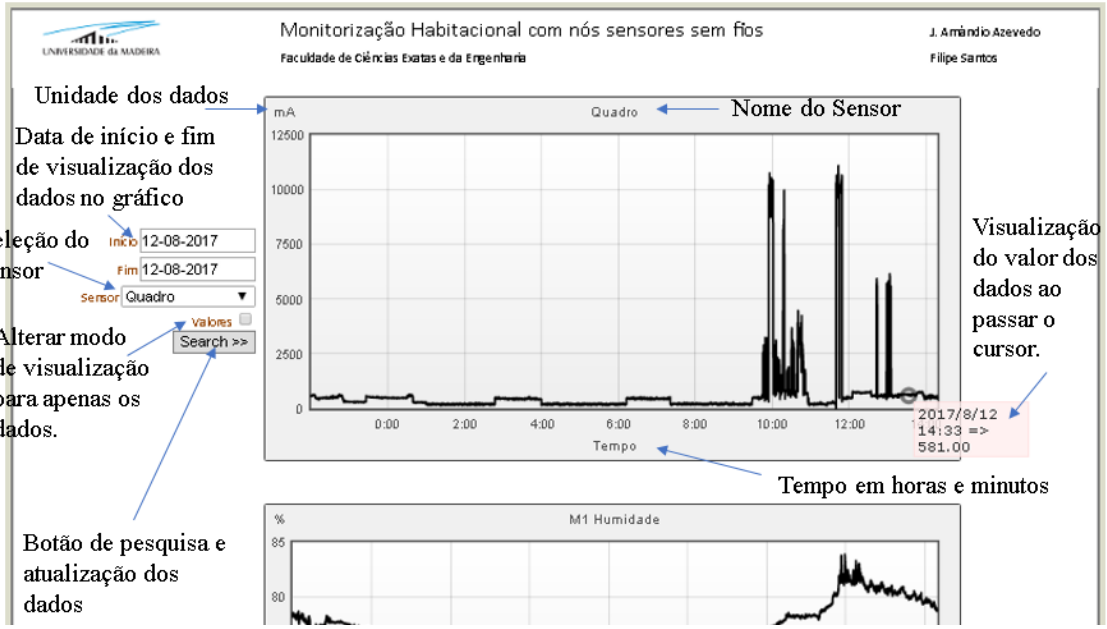


Figura 3.5 - Visualização dos dados na página *web*.

3.1.6. Antena

Para uma adequada comunicação entre os diversos módulos, é necessário utilizar uma antena apropriada. Em [67], foram estudadas diversas antenas, de diferentes tipos, destinadas para o uso em redes de sensores em fios, e escolheu-se a antena monopolo com manga, representada na Figura 3.6. As dimensões da antena manga 1 são: l2 = 34,8 mm; d1 = 0,86 mm; l1 = 19,2 mm; d1 = 0,91 mm; d2 = 2,95 mm; d3 = 5,3 mm e d4 = 6,3 mm.

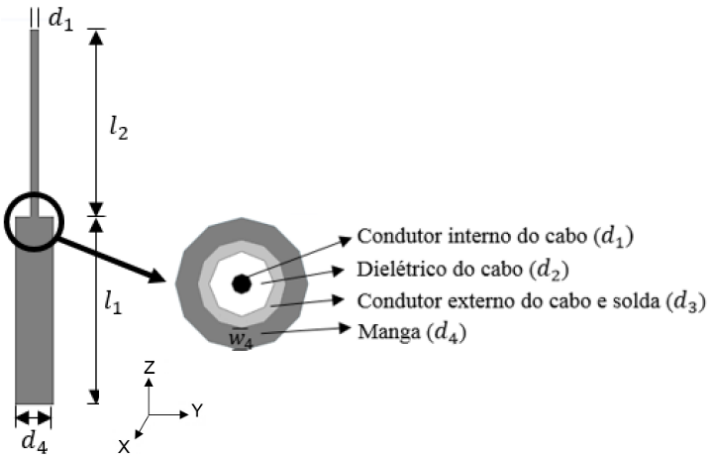


Figura 3.6 - Estrutura e parâmetros da antena monopolo com manga *[67]*.

A antena foi desenvolvida e o resultado está presente na Figura 3.7.

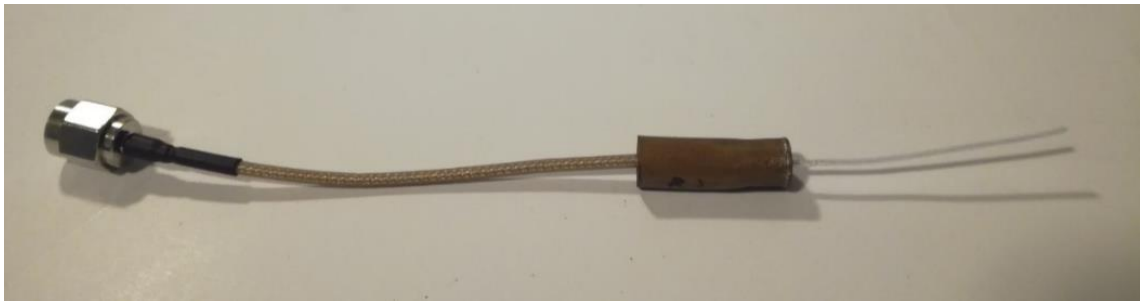


Figura 3.7 - A antena de manga curta desenvolvida para os módulos.

Optou-se por esta alternativa pois era de fácil conceção, tinha um desempenho satisfatório a nível de ganho e de tamanho relativamente pequeno. Mediu-se o seu ganho por comparação com uma antena dipolo de referência. O ganho obtido à frequência de 2,42 GHz foi de 1 dBi.

3.2. Medição de temperatura e humidade com carregamento automático de bateria

O objetivo principal foi construir um dispositivo que tivesse um sensor de temperatura e humidade, sendo capaz de enviar esses dados através do protocolo *ZigBee*. Este dispositivo dispunha de uma bateria que, quando o nível da bateria atingisse o mínimo, deveria automaticamente iniciar o carregamento, neste caso, através da rede elétrica. A tensão da bateria foi monitorizada pelo microcontrolador e, quando atingisse um certo limite, ligava um relé, que permitia a passagem da corrente elétrica para o carregamento da bateria. Esta bateria operava quando a tensão entre os terminais estivesse entre, aproximadamente, os 3,6 V e os 4,2 V, sendo estes os limites que se teve em conta na monitorização da tensão da bateria. A tensão era medida através de um simples divisor de tensão. No microcontrolador foi necessário monitorizar a porta analógica que tinha valores entre 0 e 1023, sendo que, nesta configuração, os 4,2 V na bateria correspondiam a um valor de, aproximadamente, 860 na leitura da porta analógica, enquanto os 3,6 V na bateria correspondiam a um valor de, aproximadamente, 738 na leitura da porta analógica.

Para esse processo de carregamento, utilizou-se a funcionalidade do Arduíno Fio que permite fazer o carregamento da bateria. Para o carregamento foram necessários 5V contínuos. Como referido, anteriormente, este nó sensor estava ligado à tomada elétrica sendo necessário um transformador de 220 V-AC para 5 V-DC. Tais dispositivos podiam ser comprados ou reutilizados, por exemplo, de um carregador de telemóvel. Entre o transformador e a tomada elétrica estava instalado um relé que foi ligado no momento do carregamento e desligado após a bateria estar carregada. Na Figura 3.8 apresenta-se o esquemático utilizado para a ativação do carregamento através do relé. O relé utilizado foi o OMRON G6BK-1114P-US [68].

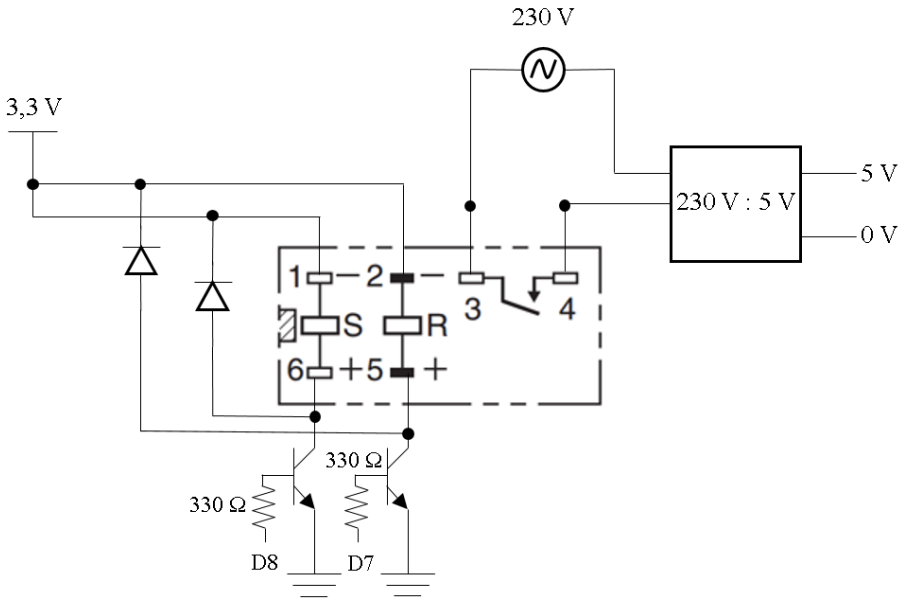


Figura 3.8 - Esquemático da ligação ao relé.

O relé era acionado através do programa em execução no microcontrolador, recorrendo às portas digitais que estavam ligadas aos pinos *SET* e *RESET* do relé, e conforme o nível de tensão da bateria referido anteriormente. O relé utilizado era do tipo *latching relay*. Este tipo de componente tem a particularidade de apenas consumir energia quando há transição de estado, ou seja, quando passa de ligado para desligado, ou vice-versa, mantendo esse estado sem consumir mais energia, sendo o mais adequado para o tipo de aplicação que se pretendia. O sensor de temperatura e humidade escolhido para este módulo foi o SHT15 [44], cujo esquema de ligação típico está presente na Figura 3.9.

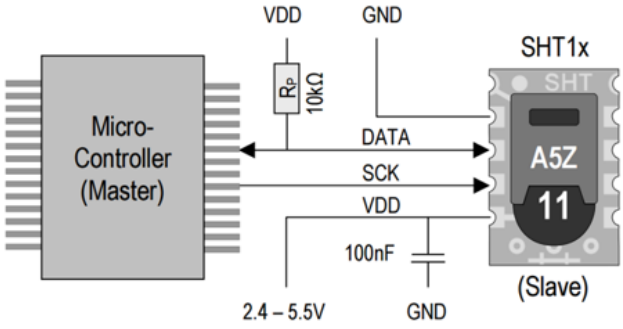


Figura 3.9 - Esquema de ligação entre o sensor SHT15 e o microcontrolador [44].

Baseou-se em [69] para obter os valores de temperatura e humidade e, depois, elaborou-se o programa principal deste módulo. Tal como explicado acima, foi adicionada a funcionalidade de "adormecer" o *XBee* entre medições. A Figura 3.10 representa o fluxograma deste programa.

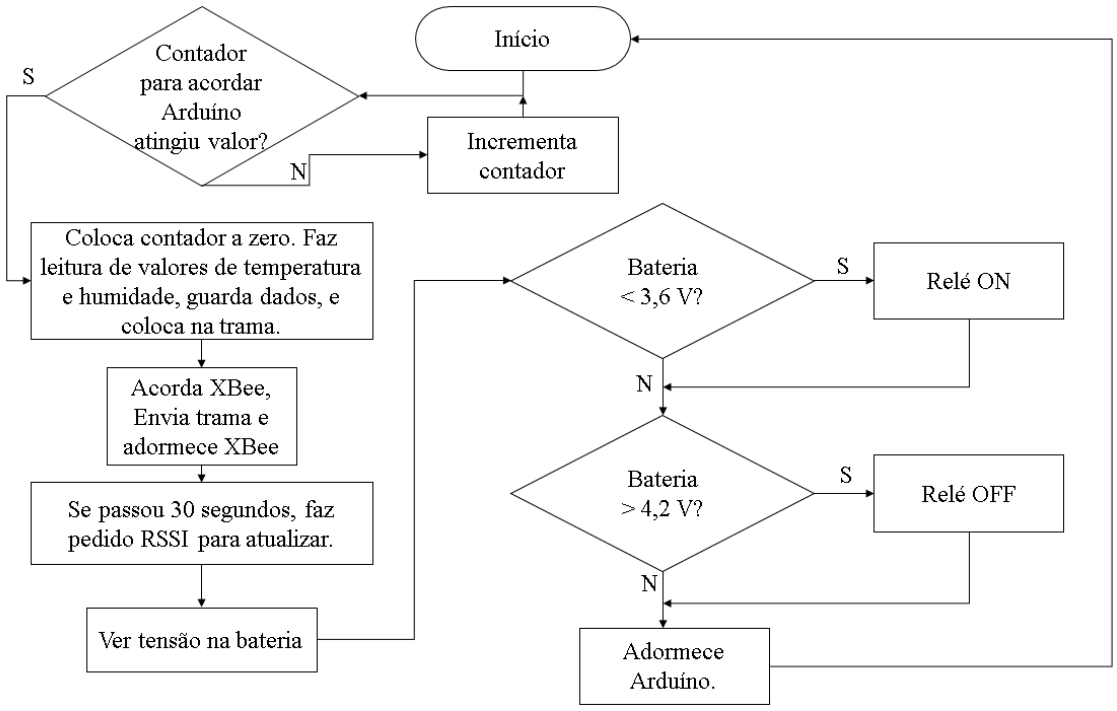


Figura 3.10 - Fluxograma do programa para o módulo de medição de temperatura e humidade com carregamento automático de bateria.

Procedeu-se à programação do Arduíno Fio tendo em conta a Figura 3.10, e implementou-se o circuito elétrico da Figura 3.11.

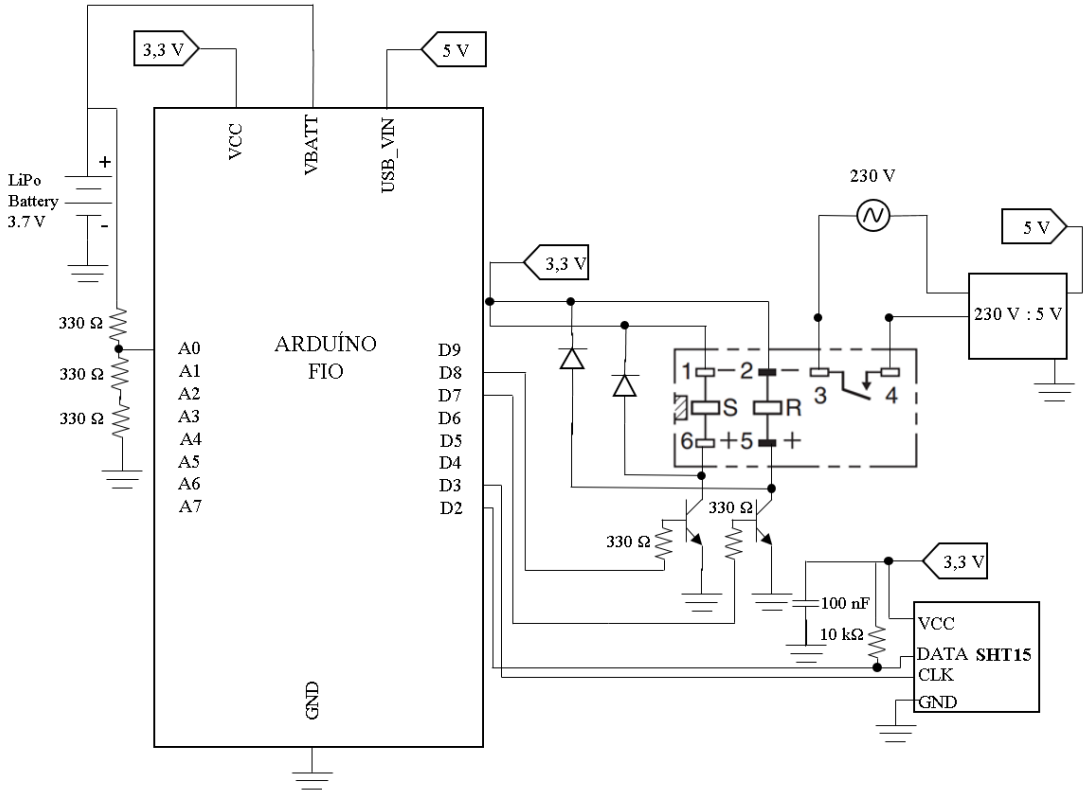


Figura 3.11 - Circuito para o módulo de medição de temperatura e humidade com carregamento automático.

O *XBee* não está representado no circuito, mas estava presente e foi ligado diretamente no *socket* do Arduíno Fio, como foi explicado anteriormente. A bateria selecionada para este módulo foi uma de 3,7 V com uma capacidade de 850 mAh, sendo que a principal razão para a escolha desta bateria foi a sua dimensão reduzida para que pudesse ser colocada dentro da caixa pretendida. No Anexo 2 são apresentadas fotos do módulo construído e o respetivo código do programa desenvolvido.

3.3. Medição de temperatura e humidade

Para o módulo de medição de temperatura e humidade pretendia-se que fosse um dispositivo simples e pequeno, com o único objetivo de fazer a leitura da temperatura e humidade, através do sensor SHT 15 [44]. O circuito está presente na Figura 3.12.

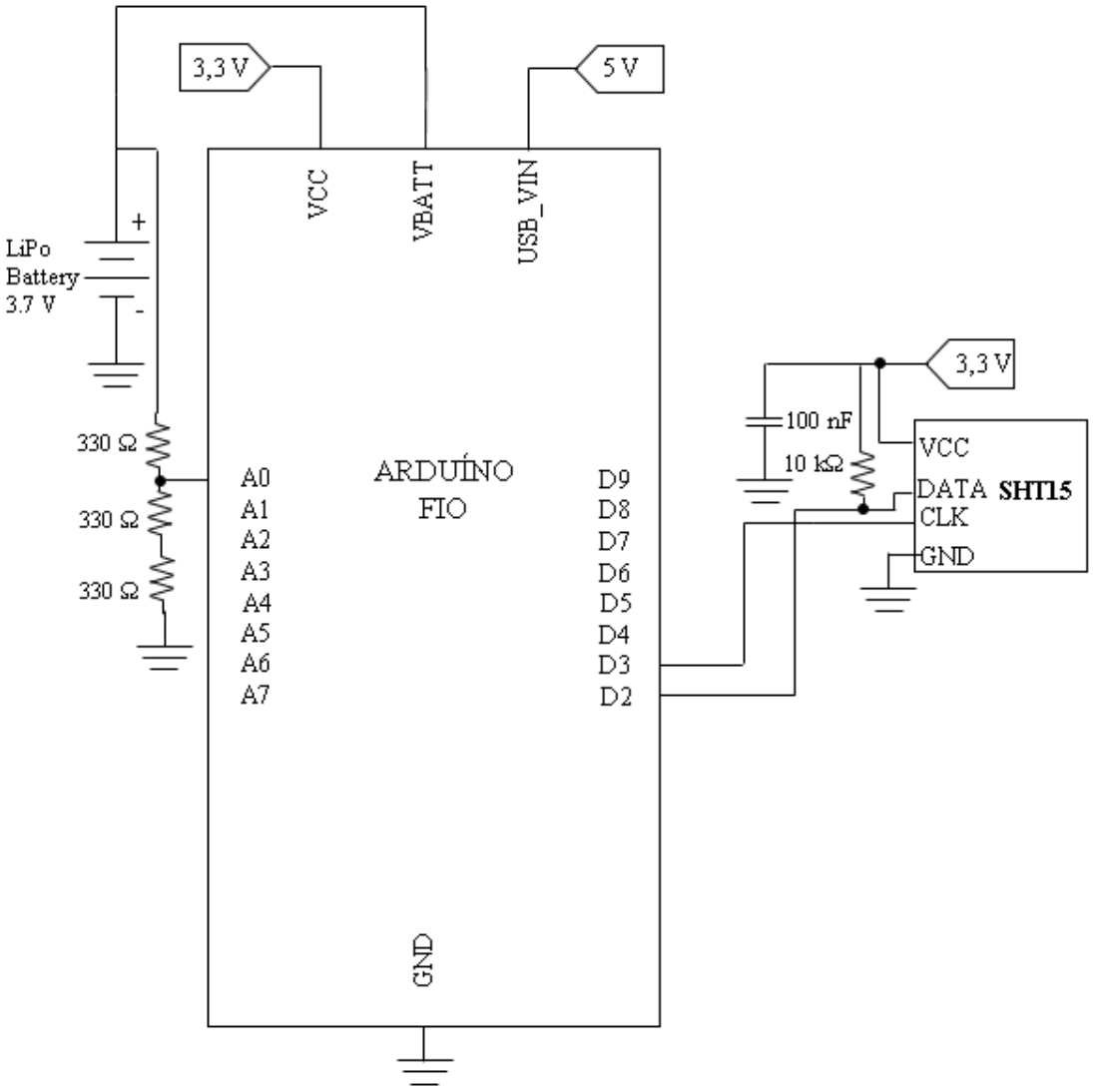


Figura 3.12 - Circuito para o módulo de medição de temperatura e humidade.

Foi utilizado um Arduíno Fio, com um *XBee*. A sua alimentação foi providenciada através da bateria. O Fluxograma para este módulo está representado na Figura *3.13*.

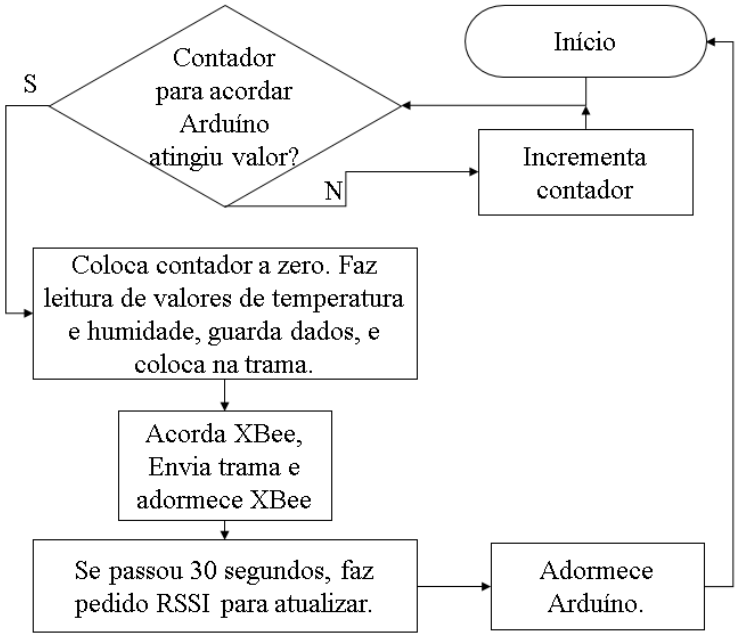


Figura 3.13 - Fluxograma para o módulo para medição de temperatura e humidade.

O programa e respetivo envio de dados é semelhante ao módulo apresentado na Secção 3.2. O carregamento da bateria era realizado aplicando 5 V na entrada de carregamento do Arduíno Fio, sendo esta tensão obtida a partir da rede elétrica com recurso a um transformador ou de outra fonte adequada. A bateria selecionada para este módulo foi de 3,7 V com uma capacidade de 1800 mAh. Em comparação com o módulo anterior, o espaço utilizado dentro da caixa era menor, pelo que se pôde optar por uma bateria de maior capacidade. Outra das razões foi não haver carregamento automático e, por isso, pretendia-se que houvesse uma maior autonomia. No Anexo 3 são apresentadas fotos do módulo construído e o respetivo código do programa desenvolvido.

3.4. Medição de temperatura, humidade, dióxido de carbono, partículas em suspensão e compostos orgânicos voláteis

Foi desenvolvido um módulo que media a temperatura, a humidade, o dióxido de carbono, os compostos orgânicos voláteis e as partículas em suspensão. O sensor para a medição da temperatura e humidade foi o DHT22, presente na Figura 3.14.

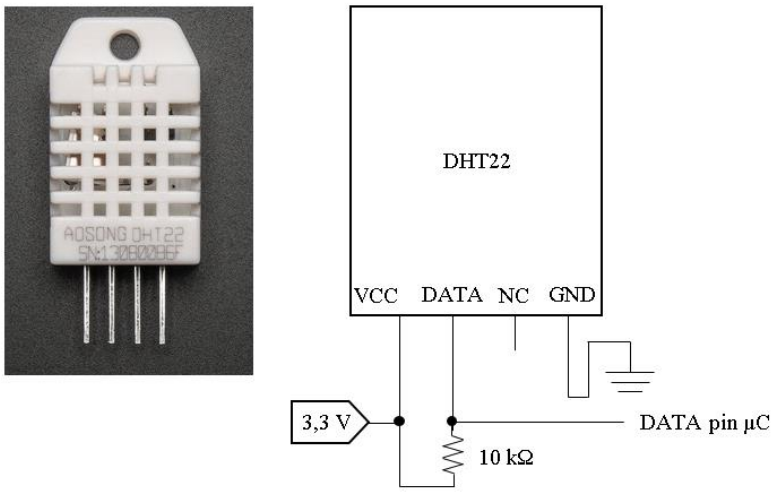


Figura 3.14 - Sensor de temperatura e humidade DHT22 e o esquema de ligação.

A monitorização do dióxido de carbono foi feita recorrendo ao sensor GSS C20 *Sensor*, presente na Figura 3.15.

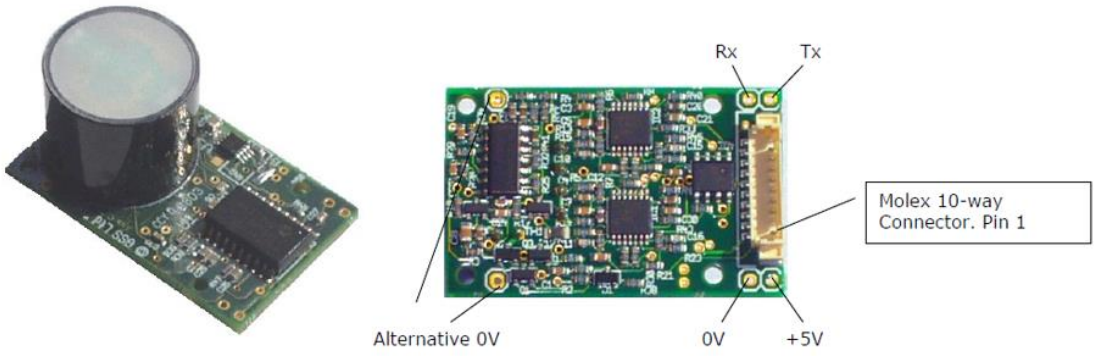


Figura 3.15 - *Sensor GSS* C20 e os pinos de ligação.

Este sensor trabalha através de comunicação por porta série (RX, TX) pelo que, inicialmente, bastaria configurar o microcontrolador para fazer a leitura da porta série.

A monitorização de partículas em suspensão foi realizada através da utilização do sensor PPD42NS, presente na Figura 3.16.

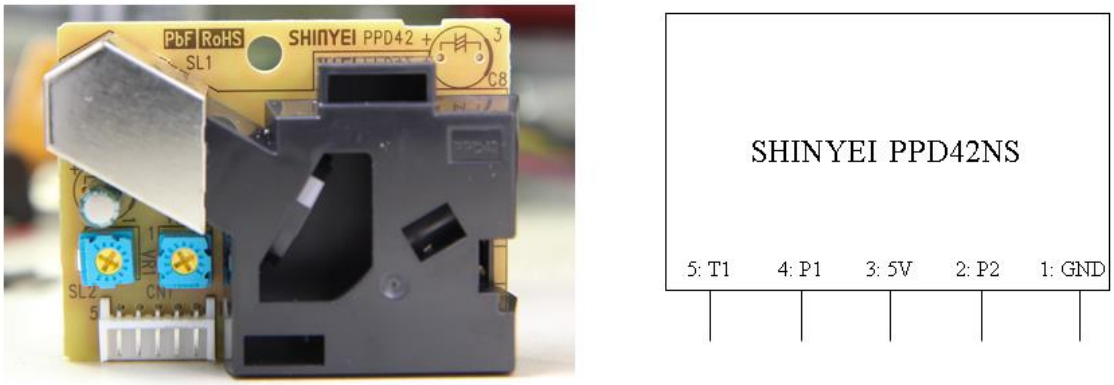


Figura 3.16 - Sensor *Shinyei* PPD42NS e os pinos de ligação.

Este sensor tem dois pinos de saída que ligavam a portas digitais do microcontrolador: P1 e o P2. A saída P1 consegue detetar partículas maiores ou iguais a 1 µm enquanto a saída P2 deteta partículas maiores ou iguais a 2,5 µm. De acordo com [70], é possível medir a concentração de partículas em suspensão de 1 µm a 2,5 µm ($PM_{2,5}$) e também a concentração de partículas maiores que 2,5 µm (PM_{10}).

Na Figura 3.17 apresenta-se o sensor MiCS5524, para medição de compostos orgânicos voláteis.

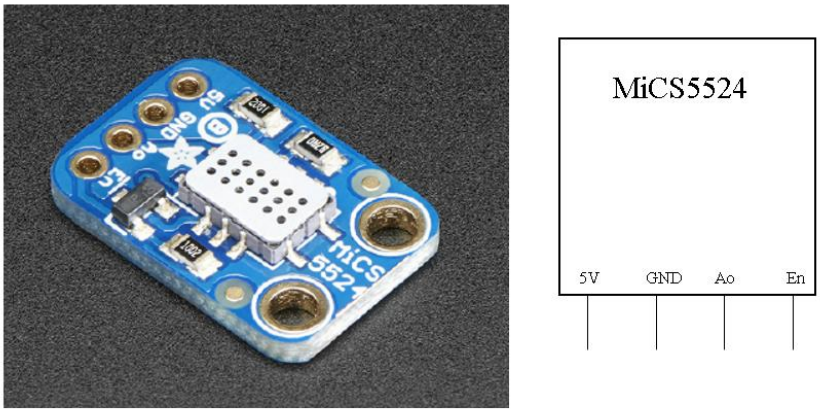


Figura 3.17 - Sensor MiCS5524 e os pinos de ligação.

O pino "Ao" liga a uma porta analógica do microcontrolador.

Para juntar todos os sensores num só módulo apresentaram-se três opções. A primeira hipótese seria fazer as medições de cada um dos sensores e enviá-las em conjunto, fazendo com que o mínimo intervalo de tempo de envio de dados fosse o tempo do sensor que demora mais tempo a fazer a sua medição. Desta forma, para este exemplo em concreto, o sensor de partículas está preparado para fornecer os dados a cada 30 segundos, fazendo com que este seja o tempo mínimo para envio dos dados, mesmo que os restantes sensores necessitem de menos tempo. Esta é a forma mais simples de enviar dados. No entanto, pretendia-se que os outros sensores transmitissem a informação em intervalos de tempo diferentes, conforme os objetivos de medição.

Para implementar o módulo que conseguisse enviar dados em intervalos de tempo diferentes podia-se configurar os *timers* que o Arduíno disponibiliza. Sabendo que o sensor de partículas demora 30 segundos a fazer a sua medição, um dos *timers* seria configurado para que, num dado período de amostragem, os outros sensores possam efetuar as suas medições e guardar os dados nas respetivas variáveis. De [58] sabe-se que se pode configurar três timers mas apenas seria possível usar dois desses, pois o *timer* 0 é essencial para a execução de funções como *millis()* e *micros(),* as quais foi necessário para o funcionamento deste programa. Assim, tinha-se dois *timers* disponíveis e a função principal, *loop(),* do Arduíno para utilizar. O outro *timer* será usado para verificar se o tempo, para cada uma das medições, já tivesse sido atingido e, quando fosse, deveria enviar os dados armazenados. A função *loop()* ocupar-se-ia de executar unicamente a função do sensor que demora mais tempo a obter os seus dados, o sensor de partículas, e quando tivesse esses dados, deveriam ser enviados.

Utilizando o método descrito, tem-se que configurar os timers de forma que estes funcionem em intervalos de tempo que permitam a execução das funções que possuem. Quando esse intervalo de tempo é atingido, o *timer* recomeça a executar as funções. Para configurar os *timers* utilizaram-se as explicações presentes em [58].

No entanto, é necessário analisar de que forma o microcontrolador funciona. Deve-se ter em conta que se estava perante um único microcontrolador com um processador e que a execução de certas funções acontece no mesmo intervalo de tempo que o funcionamento da medição de outros sensores, nomeadamente o sensor de partículas. Desta forma, durante a medição de partículas o microcontrolador terá que interromper a sua medição para executar outras funções, mesmo que estas demorem curtos intervalos de tempo. O número de funções que acontecem durante o tempo de medição de partículas é elevado, pelo que poder-se-á estar perante momentos em que o microcontrolador seja incapaz de medir partículas por estar ocupado a executar outra função. Experimentou-se este método, mas seria difícil de analisar se havia ou não perdas na contabilização das partículas, pelo que seria essencial conceber outra forma de executar todos estes processos.

O terceiro método consistiu na adição de um microcontrolador, ao já existente, em que esse seria responsável pela execução das funções para medição das partículas. O outro microcontrolador ficaria responsável pelas restantes medições e o envio dos dados. Os dois microcontroladores deveriam comunicar por I2C (*Inter-integrated Circuit*). O I2C é um protocolo que permite a comunicação entre os dispositivos. Através dos exemplos disponibilizados em [71], configurou-se o Arduíno Fio como *Master*, responsável pela medição da temperatura, da humidade, do CO_2, dos COV e pelo envio dos dados. O outro dispositivo foi o Arduíno Pro Mini, que foi configurado como *Slave*, teria de efetuar as medições das partículas e guardar os resultados nas variáveis respetivas. Quando o *Master* verificar que fosse necessário enviar os dados sobre a concentração de partículas em suspensão, pedia apenas ao *Slave* os resultados armazenados nas variáveis. O *Slave* enviava esses dados por I2C para o Master e este tratava de enviar os dados por *XBee*. A Figura 3.18 mostra o fluxograma para este método.

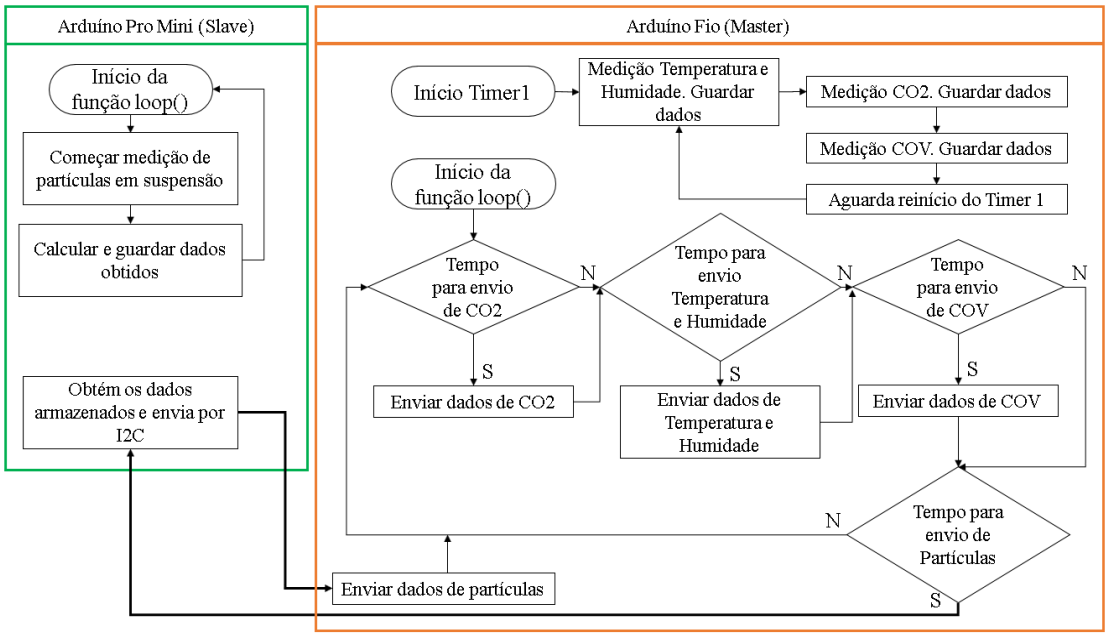


Figura 3.18 - Fluxograma do módulo.

Este método, apesar de se tornar mais complexo pela necessidade de ter outro microcontrolador, foi o mais indicado pois permitia o funcionamento mais adequado de todos os sensores, tendo a capacidade de definir tempos diferentes de medição para cada um. Optou-se por este método e concebeu-se o circuito apresentado na Figura 3.19.

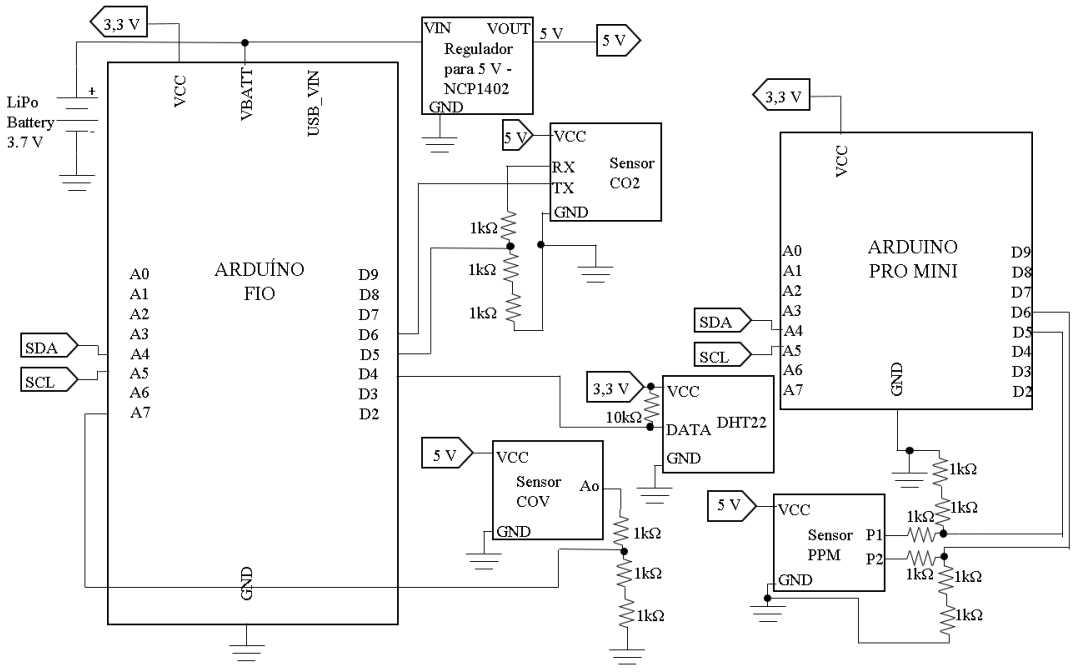


Figura 3.19 - Circuito para o módulo de medição de CO_2, de COV, de temperatura, de humidade e de partículas em suspensão.

Neste circuito foi necessário regular a tensão da bateria para 5 V pois os sensores de partículas, de CO_2 e dos compostos orgânicos voláteis funcionavam com essa tensão. Para

fazer a ligação dos sensores ao microcontrolador utilizou-se um divisor resistivo para baixar a tensão. Apenas o sensor DHT22 podia ser alimentado por 3,3 V. As ligações analógicas A4 e A5 de ambos os microcontroladores providenciaram a comunicação entre ambos, através de I2C, como referido acima. O *XBee* não está representado na Figura *3.19*, mas foi ligado diretamente no Arduíno Fio. A bateria selecionada para este módulo foi uma de 3,7 V com uma capacidade de 2500 mAh. O espaço utilizado dentro da caixa é suficiente para colocar uma bateria desta capacidade. Outra das razões foi não haver carregamento automático juntamente com o facto que este módulo tinha um consumo muito maior devido aos sensores implementados. No Anexo 4 são apresentadas fotos do módulo construído e o respetivo código do programa desenvolvido.

3.5. Medição de corrente numa tomada elétrica

Para o módulo de medição de corrente elétrica numa tomada utilizaram-se dois tipos de sensores de corrente: um invasivo e outro não invasivo. O sensor não invasivo utilizado foi o CR3110-3000 [57], representado na Figura 3.20, tendo a capacidade de medir correntes até 75 A.

Figura 3.20 - O sensor de corrente CR3110-3000.

O sensor invasivo foi o ACS712-30A [58], representado na Figura 3.21.

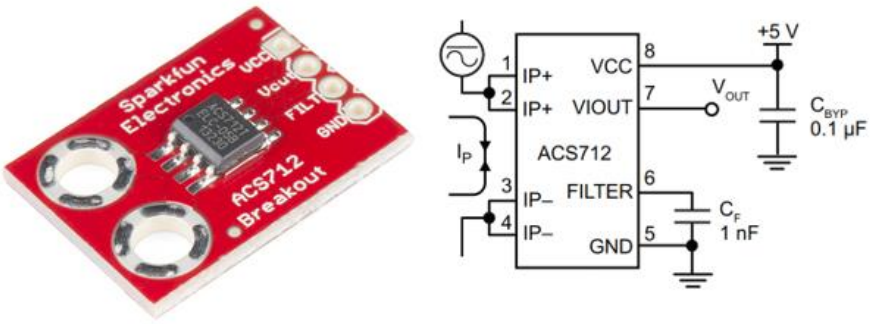


Figura 3.21 - O sensor de corrente ACS712-30A e o seu esquema de ligação.

A grande necessidade de ter dois tipos de sensor prende-se pelo facto de se querer abranger uma gama de corrente, de 2 mA até 20 A. O sensor ACS712 mede desde -30 A até +30 A, sendo que, a leitura mais baixa que este sensor conseguirá detetar será de,

aproximadamente, 60 mA, tendo em conta um ADC de 10 bits. Como se pretendia ler valores mais baixos de corrente, recorreu-se a outro tipo de sensor, neste caso o CR3110-3000. Este sensor gera uma tensão que é causada pelo campo magnético criado pela passagem da corrente de um determinado condutor, sendo essa tensão proporcional à corrente. Assim é possível estabelecer uma relação que torna possível calcular a corrente que passa nesse condutor. Para valores muito baixos de corrente, a tensão gerada é muito baixa pelo que pode ser aumentada utilizando um circuito de amplificação. Na Figura 3.22 apresenta-se o circuito de amplificação escolhido, com alimentação não simétrica.

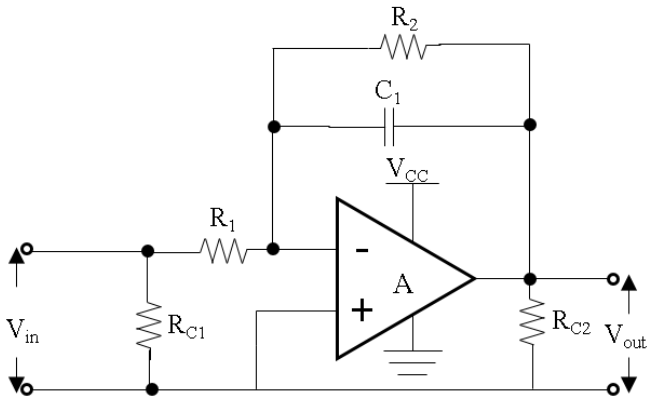


Figura 3.22 - Circuito de amplificação com montagem de ganho inversor [72].

Trata-se de um circuito de amplificação que utiliza um amplificador operacional com montagem de ganho inversor. O ganho do amplificador é dado por

$$A = -\frac{R2}{R1} \tag{3.1}$$

A entrada V_{in} corresponderá à tensão proveniente do sensor e a saída V_{out} será o sinal amplificado que será analisado. O condensador utilizado filtra as variações mais elevadas resultando num sinal, na saída, com menos ruído. Para que não houvesse qualquer afetação no sinal amplificado, calculou-se a frequência de corte

$$f_c = \frac{1}{2\pi(R2 \times C1)} = \frac{1}{2\pi \times (150 \times 10^3 \times 10 \times 10^{-9})} = 106{,}103\ Hz \tag{3.2}$$

Dado que se estava a trabalhar com a frequência da rede, 50 Hz, os valores escolhidos são adequados. Montou-se o circuito utilizando um amplificador operacional ISL28248FHZ [73] e dimensionando um ganho de A=100 (*R2*=150 kΩ; *R1*=1,5 kΩ; *C1*=10 nF; *RC1*=2,2 kΩ; *RC2*=10 kΩ). Através da utilização de um multímetro, controlou-se o valor eficaz da corrente e, através do osciloscópio, tinha-se a visualização dos sinais após o sensor e após o circuito de amplificação. Na Figura 3.23 apresenta-se a captura do ecrã do osciloscópio com dois sinais diferentes.

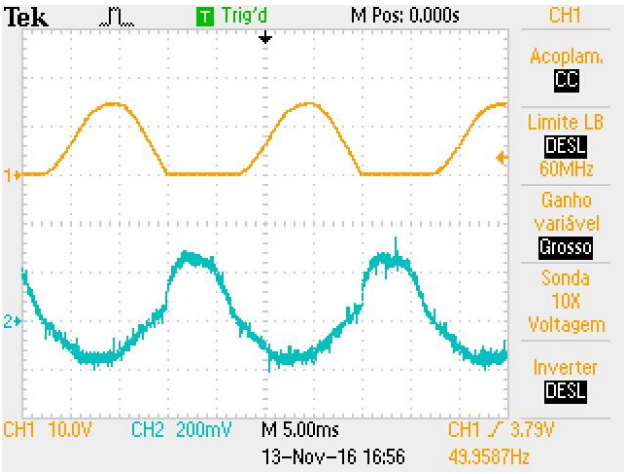


Figura 3.23 - Captura de ecrã do osciloscópio para uma corrente de 32 mA.

A azul apresenta-se o sinal antes do circuito de amplificação e a laranja apresenta-se o sinal amplificado. Para fazer a leitura destes sinais foi criado um programa no Arduíno. Com esse programa pretendia-se tirar o máximo de amostras, que o Arduíno conseguia, em 20 ms (1/50 Hz=20 ms) e guardar num vetor. Foram efetuadas várias medições desde 1 mA até aproximadamente 50 mA, sendo os resultados apresentados na Figura 3.24. As cargas utilizadas para estes testes foram resistências variáveis de alta potência.

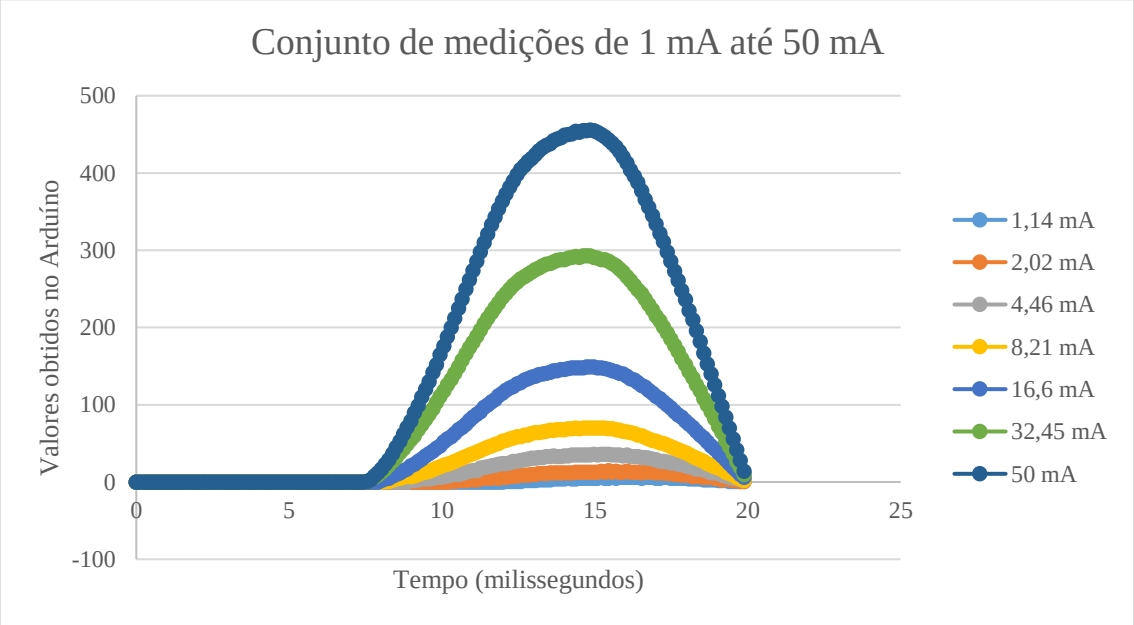


Figura 3.24 - Gráfico do conjunto de medições, obtidos através do Arduíno, de 1 mA até 50 mA.

Verifica-se que o aumento da corrente o sinal obtido no Arduíno também aumenta. De forma a verificar a expressão dos valores obtidos selecionou-se o máximo de cada conjunto de amostras, para cada um dos valores de corrente testado, e construiu-se o gráfico presente na Figura 3.25.

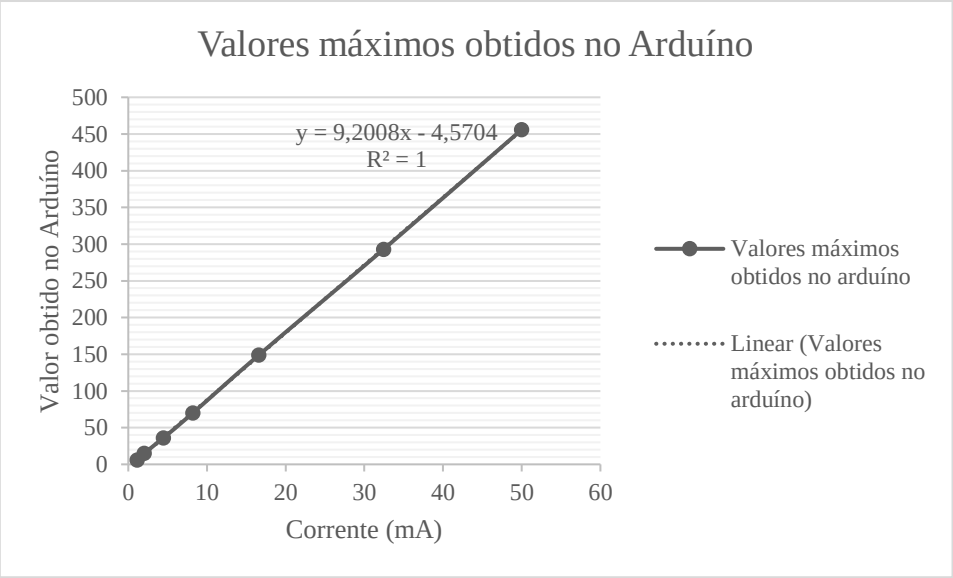


Figura 3.25 - Linearidade dos valores obtidos.

Analisando a Figura 3.25 verifica-se que os valores lidos no microcontrolador aumentam de forma linear com o aumento da corrente. Assim, foi possível encontrar uma expressão que, dado um determinado valor obtido no Arduíno, fosse possível encontrar o valor da corrente. Interessa, no entanto, que se saiba o valor eficaz desse conjunto de valores e, para isso, utiliza-se a expressão,

$$x_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} {x_i}^2} = \sqrt{\frac{{x_1}^2 + {x_2}^2 + \cdots + {x_N}^2}{N}} \tag{3.3}$$

em que x representa cada uma das amostras e N representa o número total de amostras.

Sabendo que este amplificador operacional tem uma tensão de saturação de 2,6 V, foi utilizado mais um amplificador operacional, com um ganho menor, para que fosse capaz de medir uma gama superior. Como cada circuito integrado do amplificador operacional acima mencionado possui, na prática, dois amplificadores, recorreu-se ao segundo. Assim, com estes dois amplificadores operacionais e de acordo com os testes efetuados, foi possível medir uma gama de 2 mA até 250 mA, em que o primeiro amplificador mediria de 2 mA até 120 mA e o segundo mediria de 120 mA até 250 mA. Para correntes superiores, o valor foi obtido através do sensor ACS712, referido anteriormente. A montagem de ambos os sensores foi feita conforme a Figura 3.26.

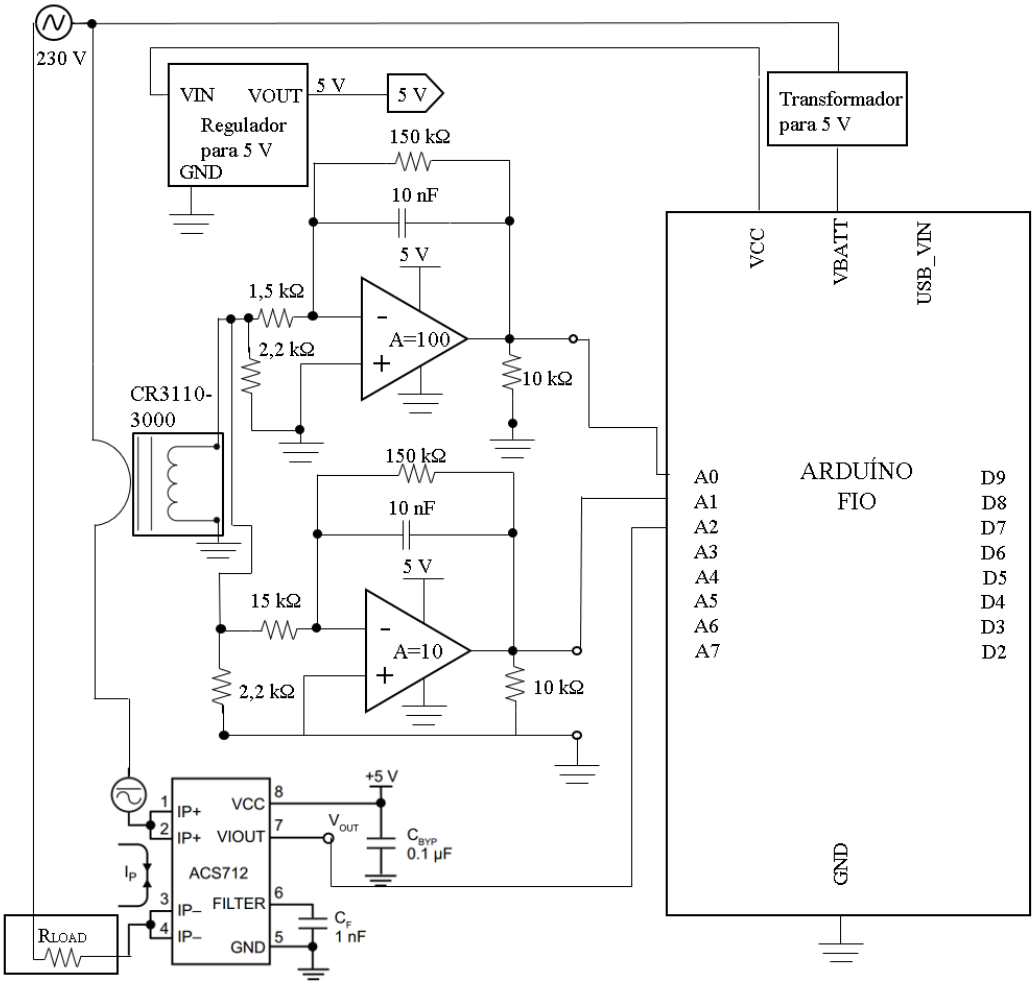


Figura 3.26 - Esquemático para o dispositivo de medição de corrente.

Na Figura *3.26* verifica-se 3 saídas V_{out} e cada uma delas estará ligada a uma entrada analógica do microcontrolador. Foi desenvolvido um programa capaz de fazer a leitura e respetivo cálculo da corrente, através das expressões obtidas, optar pelo valor de corrente mais adequado de acordo com as gamas projetadas para cada sistema e enviar esse valor por *XBee*. Na Figura 3.27 apresenta-se o fluxograma do programa.

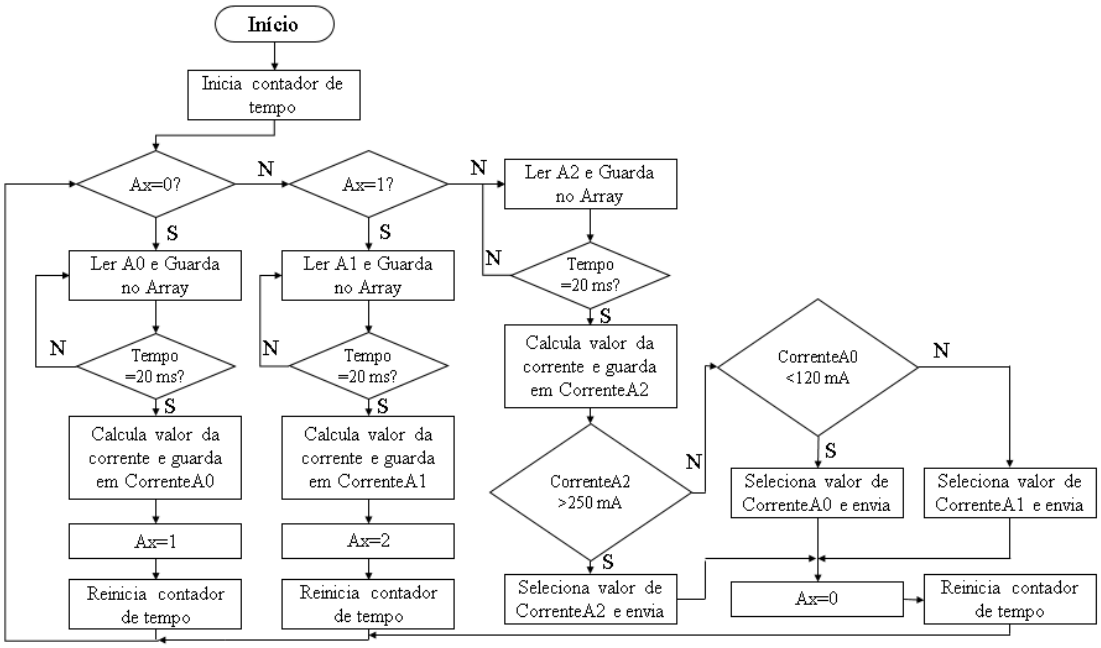


Figura 3.27 - Fluxograma para o programa do dispositivo para medição de corrente numa tomada elétrica.

O programa foi implementado no Arduíno Fio que, por sua vez, utiliza o *XBee* para o envio dos dados. No Anexo 6 estão apresentadas fotos do módulo construído e o respetivo código do programa desenvolvido.

3.6. Medição de corrente num quadro elétrico

O dispositivo de medição de corrente num quadro elétrico utilizou o sensor de corrente não invasivo e, desta forma, media-se a corrente sem interromper qualquer circuito, algo que é bastante útil quando se pretende fazer medições através do quadro elétrico de uma habitação. Sabe-se que o limite do sensor era o suficiente para uma aplicação doméstica o que bastou projetar um sistema que fizesse o tratamento do sinal. Como se apresentou na secção 3.5, necessitou-se de dois amplificadores operacionais para medir duas gamas de valores. Para este exemplo, acrescentou-se mais dois amplificadores operacionais para que pudesse medir, apenas com este sensor, um máximo de 20 A. Assim, na Figura 3.28, apresenta-se um fluxograma semelhante ao dispositivo desenvolvido para a medição da corrente numa tomada elétrica.

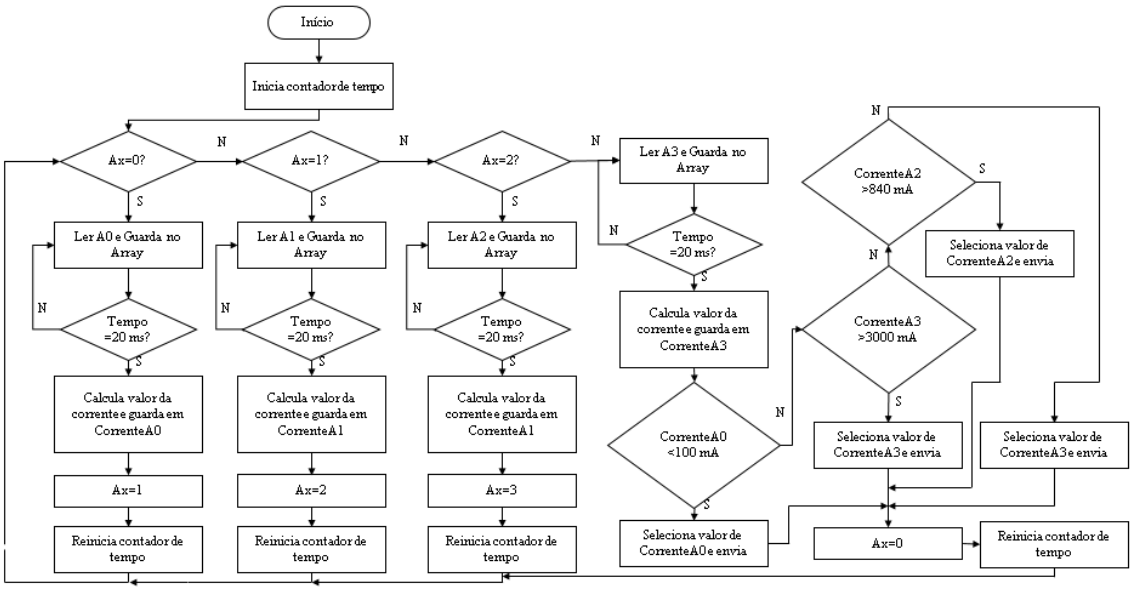


Figura 3.28 - Fluxograma para o programa do dispositivo que media a corrente no quadro elétrico.

O amplificador operacional utilizado para as duas gamas mais altas foi o LM-741 [74]. O programa foi implementado no Arduíno Fio que, por sua vez, tinha o *XBee* para o envio dos dados. Na Figura 3.29 apresenta-se o circuito para este módulo.

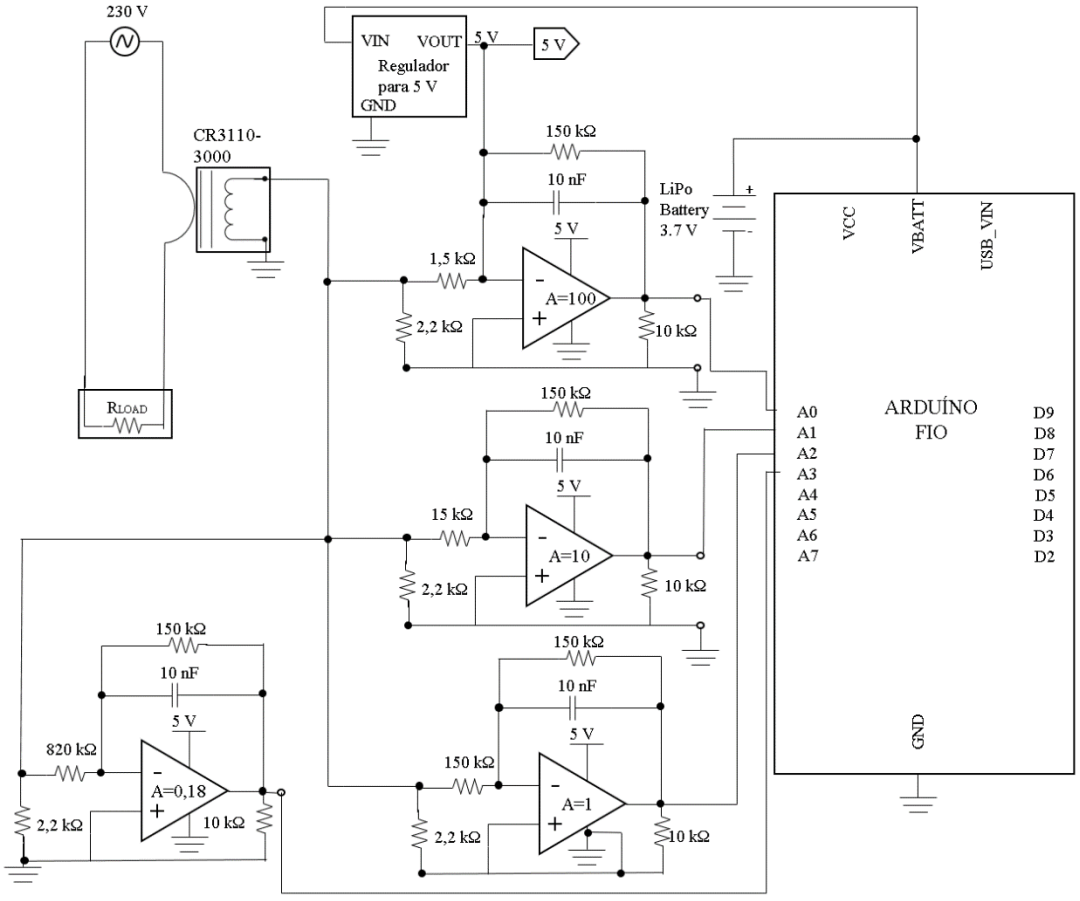


Figura 3.29 - Circuito para o módulo de medição de corrente elétrica.

A bateria selecionada para este módulo foi uma de 3,7 V com uma capacidade de 2500 mAh. O espaço utilizado dentro da caixa era suficiente para colocar uma bateria desta capacidade, dando maior autonomia e uma vez que não possuía carregamento automático. No Anexo 5 estão apresentadas fotos do módulo construído e o respetivo código do programa desenvolvido.

4. Testes ao sistema e análise de resultados

Neste capítulo são apresentados os resultados dos testes efetuados à globalidade do sistema.

4.1. Local dos testes

Na Figura 4.1 apresenta-se a planta 3D da habitação onde o sistema foi testado.

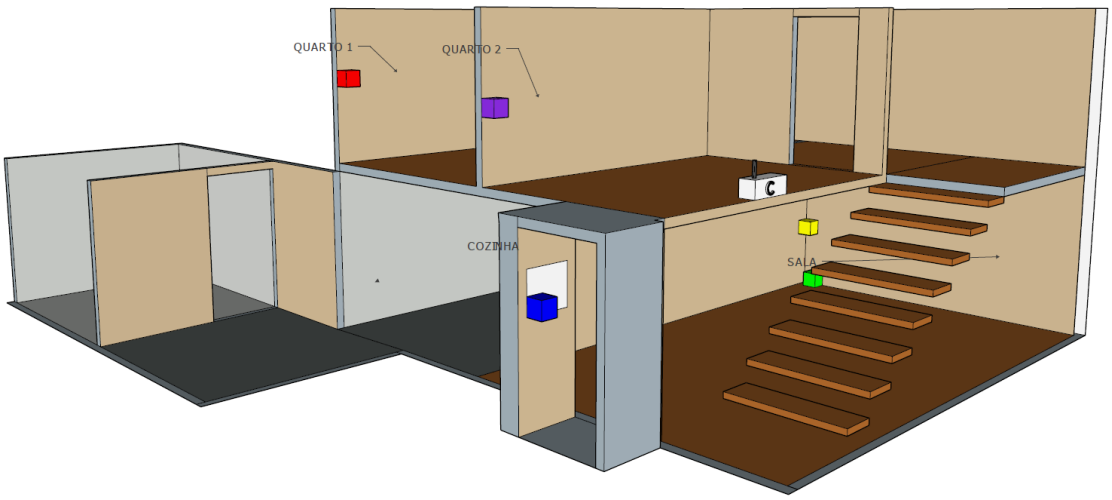


Figura 4.1 - Planta 3D da habitação onde foi testado o sistema de nós sensores sem fios.

No Quarto 2, representado como uma "caixa branca", encontra-se o módulo coordenador. As outras "caixas" serão os módulos com os sensores. A "caixa" verde representa o módulo de medição de corrente na tomada elétrica, a "caixa" azul representa o módulo de medição de corrente no quadro elétrico, a "caixa" amarela representa o módulo de medição de temperatura, de humidade, de COV, de CO_2 e de partículas em suspensão. A "caixa" roxa trata-se do módulo de medição de temperatura e de humidade com carregamento e a "caixa" vermelha, é o módulo de medição de temperatura e de humidade sem carregamento. No entanto, para os testes efetuados alguns dos módulos foram colocados no mesmo lugar para comparação de resultados.

4.2. Testes de medição de corrente numa tomada elétrica

Começou-se por efetuar medições para verificar a resposta do nó de medição numa tomada elétrica e escolheu-se um conjunto de dispositivos de uso corrente em ambientes domésticos. Para comparar os valores obtidos recorreu-se ao multímetro *Amprobe* 33XR-A. Apontaram-se os valores na Tabela 4.1, juntamente com o erro relativo.

Tabela 4.1 - Resultados dos testes ao módulo de medição de corrente numa tomada.

Equipamento	Corrente medida no multímetro (mA)	Valor medido (mA)	Erro absoluto (mA)	Erro relativo (%)
Lâmpada 1	15,9	14	1,9	11,95%
Carregador telemóvel	42	39	3	7,14%
Lâmpada 2	61	56	5	8,20%
Lâmpada 3	173	178	5	2,89%
PowerBox + TV	190	192	2	1,05%
Computador portátil	280	264	16	5,71%
Consola	310	320	10	3,23%
PowerBox + TV+Consola	490	511	21	4,29%
Secador 1	2340	2430	90	3,85%
Secador 2	7500	7633	133	1,77%
Chaleira	9390	9489	99	1,05%

Verifica-se que o erro relativo, na globalidade das medições, tem um valor percentual médio que ronda os 4,6%, sendo que o seu máximo atingiu os 11,95% no teste de medição de corrente de um dispositivo (Lâmpada 1) que consumia um valor de 15,9 mA. Apesar de ter um erro de 1,9 mA, estando-se perante uma gama baixa de valores, acaba-se por ter um erro relativo alto, apesar do seu valor absoluto ser o mais baixo. Para gamas maiores o erro relativo era, na maioria das vezes, inferior a 5%, tal como se pode observar na Figura 4.2. A linha amarela representa o valor do erro relativo, verificando-se uma diminuição à medida que os valores aumentam.

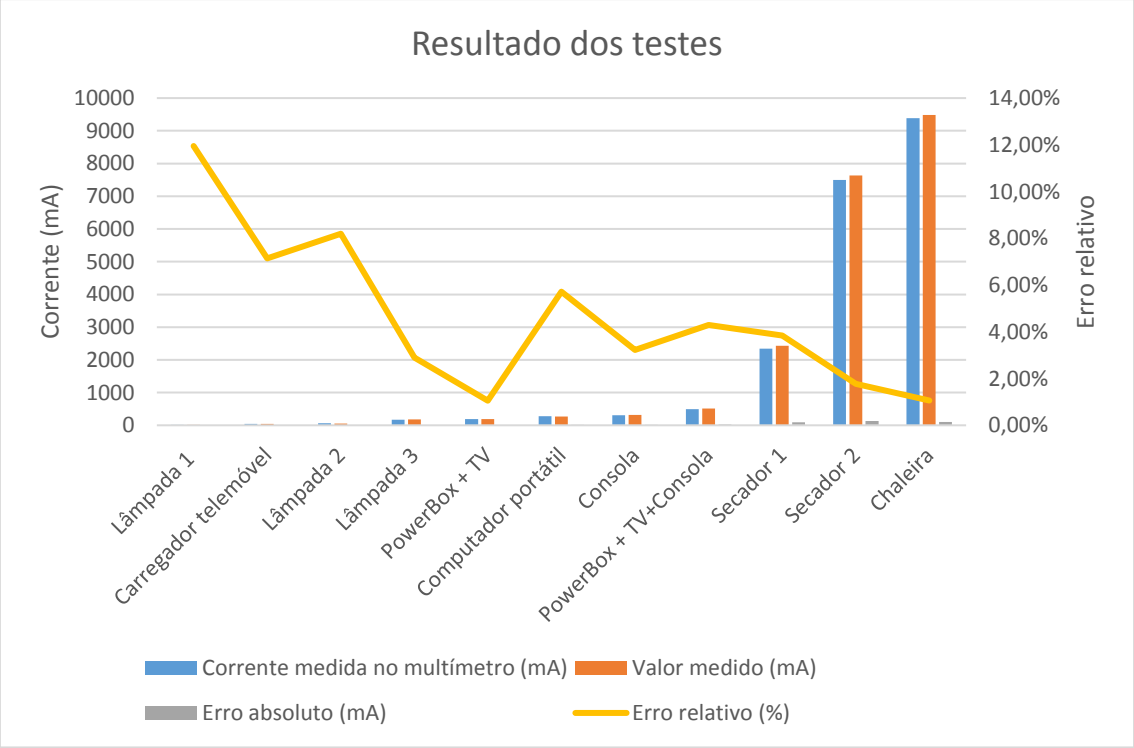


Figura 4.2 - Gráfico dos resultados dos testes.

Na Figura 4.3 apresentam-se os resultados, visualizados diretamente na plataforma *web*, da corrente consumida durante um intervalo de tempo de diferentes dispositivos, como carregadores de computador e telemóvel ou outros dispositivos de consumo relativamente baixo, que muitas vezes são deixados desnecessariamente ligados à tomada, causando consumo excessivo de energia.

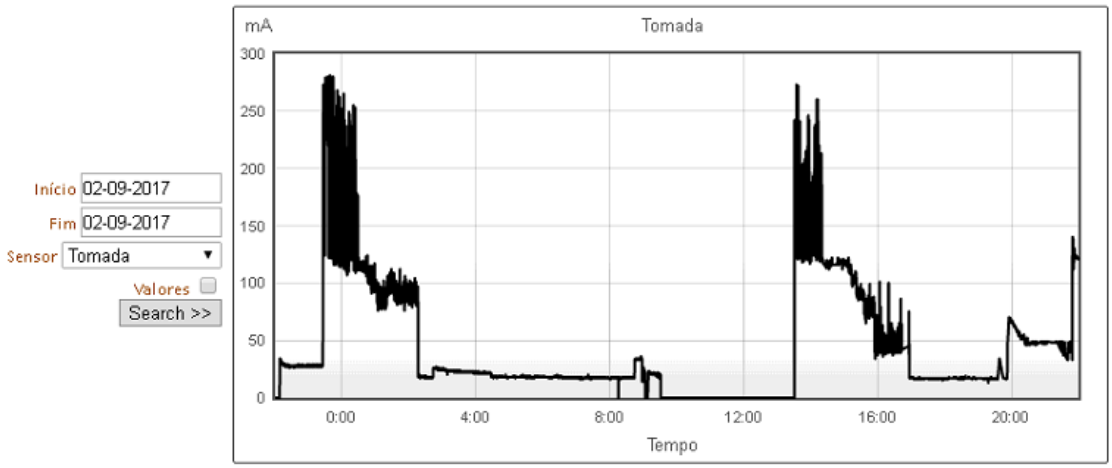


Figura 4.3 - Visualização, na plataforma *web*, dos dados da tomada elétrica.

Optou-se por não colocar este dispositivo com bateria estando sempre ligado à tomada. Optou-se, também, por não configurar este dispositivo como nó terminal, mas sim como nó *router*, aproveitando o facto de estar sempre ligado à energia, sendo um nó importante para uma maior cobertura da rede sem fios na habitação. O consumo total deste nó rondava os 100 mA, tendo em conta que tanto o Arduíno como o *XBee* estavam em utilização constante.

4.3. Testes de medição de corrente no quadro elétrico

Testou-se o módulo no quadro elétrico de uma casa. Para fazer a comparação com os valores obtidos no dispositivo desenvolvido utilizou-se o equipamento *Fluke* 80i-410 [75], que tem a capacidade de medir desde 2 A até 400 A. Este dispositivo tem que ser ligado a um multímetro, em modo voltímetro, e cada ampére, que passa através deste dispositivo, corresponderá a 1 mV. O multímetro utilizado foi o *Amprobe* 33XR-A [76]. No quadro elétrico colocaram-se ambos os dispositivos. Passou-se a testar diversos dispositivos de grande consumo estando os resultados presentes na Tabela 4.2.

Tabela 4.2 - Resultados dos testes realizados na medição de corrente no quadro elétrico.

Equipamentos	Medido no 80i410 (A)	Módulo (A)	Erro Absoluto (A)
*+Secador3	2,5	2,3	0,2
*+Secador2	5,5	6,5	1
*+Secador1	7	8,4	1,4
*+Chaleira	9	10,5	1,5
*+Forno	9	10,6	1,6
*+Exaustor+Forno	9,5	11,4	1,9
*+Chaleira+Secador2	14,5	15,9	1,4
*+Secador1+Secador2	13	12,1	0,9
*+Chaleira+Secador1	16	17,4	1,4
*+Chaleira+Secador1+Secador2	21,5	21,1	0,4
*+Forno+Chaleira+Secador2	23	22,4	0,6
*+Forno+Chaleira+Secador2+Secador3	24,5	22,7	1,8

(* - Conjunto de dispositivos que estavam em funcionamento enquanto os testes decorriam: Frigorífico, luzes, televisão, *PowerBox*).

É possível verificar que existe alguma diferença entre ambas as medições em todos os testes. Ter-se-á que considerar que os dispositivos de medição para comparação do módulo desenvolvido tinham uma determinada margem de erro sabendo que o *Fluke* 80i-410 tem um erro máximo de 3 % ± 0,4 A e o *Amprobe* 34XR-A tem um erro máximo de 0,5 % de leitura, sem desconsiderar que o próprio módulo terá um erro tendo em conta que a reta de calibração calculada não alcançou valores tão altos como os aqui medidos. Na Figura 4.4 apresenta-se o gráfico para ilustrar melhor os resultados atingidos. A laranja apresentam-se os dados medidos pelo nó sensor, a azul pelo *Fluke* 80i-410 e a cinzento apresenta-se o erro absoluto.

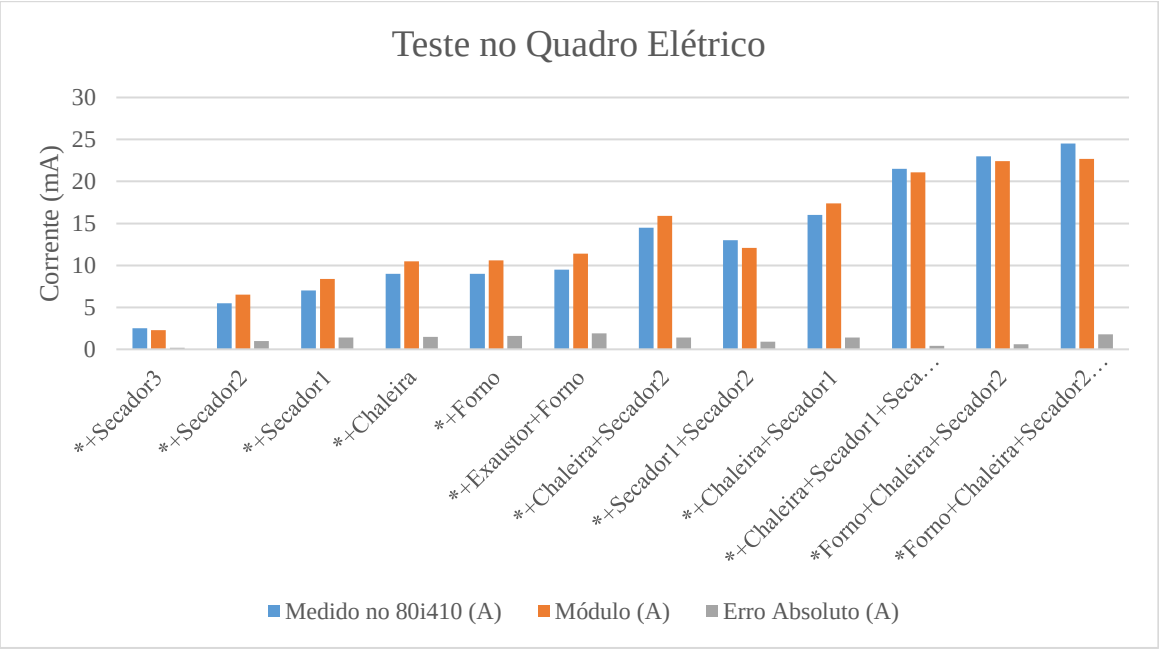


Figura 4.4 - Gráfico com os resultados das medições no quadro elétrico.

Passando à monitorização diária, que poderia ser visualizada em tempo real através da plataforma *web*, apresenta-se a Figura 4.5 que possui os dados medidos, de 10 em 10 segundos, durante 24 horas.

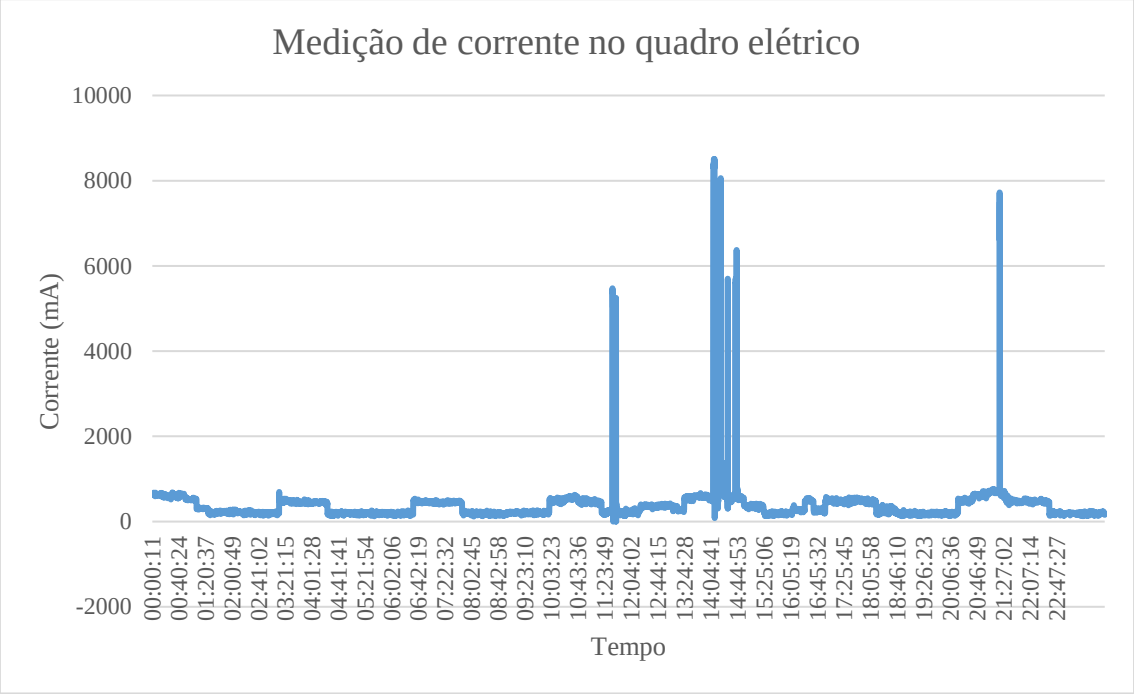


Figura 4.5 - Gráfico da monitorização do consumo de corrente durante 24 horas.

Nesta análise tem-se um valor médio de intensidade de corrente de 383 mA, nestas 24 horas, com um pico máximo de consumo de 8509 mA. É importante constatar que existem 3 momentos em que se concentram os maiores picos de consumo que coincidiram com a confeção de alimentos ou outros equipamentos de aquecimento rápido (exemplo: secador ou chaleira). Verifica-se também que existe um consumo constante relacionado com equipamentos como uma televisão, portátil ou carregadores de telemóvel. Nota-se também que existe um consumo intermitente com um período de, aproximadamente, 5 horas que neste caso tratava-se do funcionamento do frigorífico da habitação testada. Na Figura 4.6 apresenta-se a visualização, através da plataforma *web*, durante 4 dias consecutivos, nomeadamente de segunda-feira a quinta-feira.

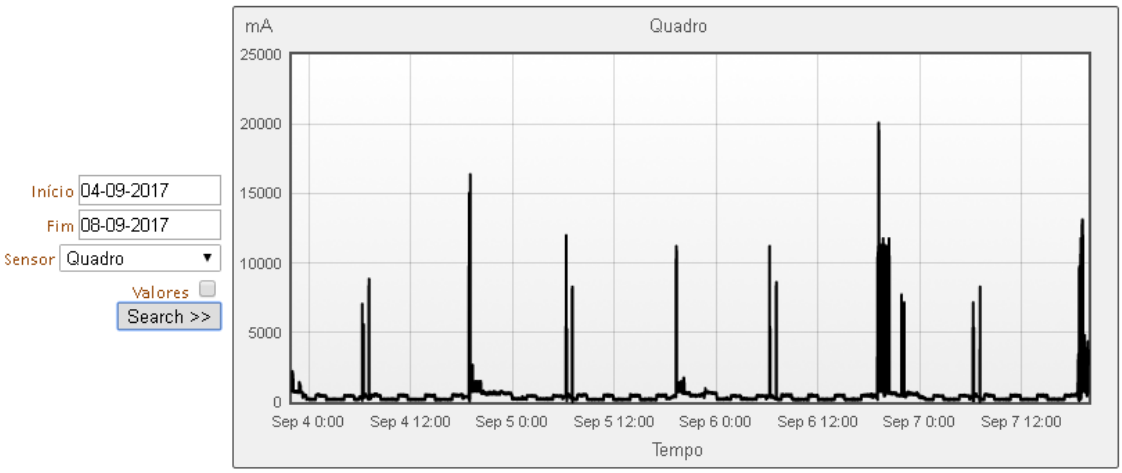


Figura 4.6 - Visualização na plataforma *web* do consumo para vários dias.

Escolheu-se este intervalo de tempo pois trata-se da rotina mais comum durante todo o ano, em que as pessoas desta habitação estão a trabalhar. Os picos de consumos que aqui se observam acontecem sempre entre as 07:00 e 08:00 e depois um segundo pico entre 19:00 e as 21:00. O primeiro pico acontece devido ao consumo de energia elétrica que acontece ao acordar, nomeadamente uso de secadores e chaleiras, e o segundo pico acontece ao final do dia, quando volta a haver presença de pessoas, para fazer o jantar e outras atividades rotineiras. Durante o resto do tempo não acontecem grandes consumos pois não existe presença de pessoas na habitação ou, essas pessoas, estão a dormir. Com estes valores, neste intervalo de tempo, obteve-se um consumo médio de 530 mA. Este módulo tinha um consumo de, aproximadamente, 10 mA.

4.4. Testes de medição de temperatura e humidade com carregamento automático de bateria

A medição de temperatura e humidade utilizando o módulo com carregamento automático de bateria foi efetuada num espaço comum de uma habitação. Optou-se por avaliar a variação da temperatura e humidade, primeiramente, ao longo de 24 horas e, depois, ao longo de 8 dias.

Na Figura 4.7 apresentam-se os valores de temperatura obtidos a cada 10 segundos, durante 24 horas, do dia 25/08/2017. A temperatura máxima foi de 25,8 ºC e a mínima de 24,3 ºC, tendo uma média de 25,1 ºC neste intervalo de tempo. Os valores mais baixos foram atingidos entre as 07:00 e as 9:00 e os mais altos entre as 13:00 e as 17:00.

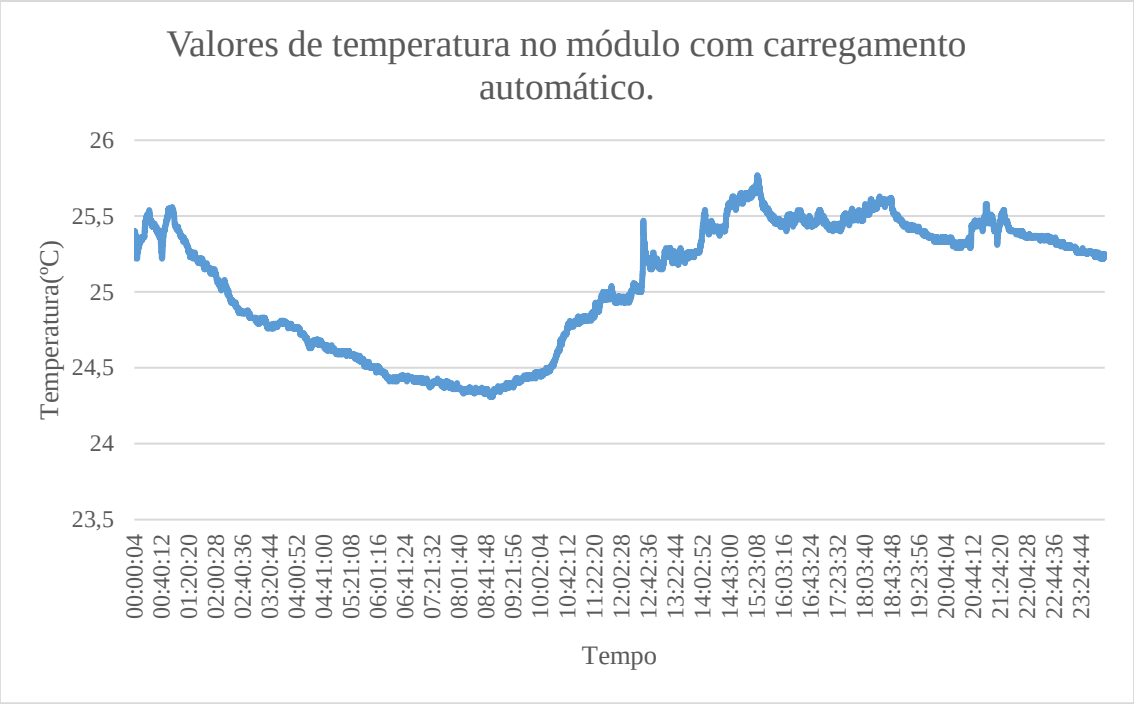


Figura 4.7 - Gráfico com valores de temperatura durante 24 horas (dia 25/08/2017) para o módulo de carregamento automático.

De acordo com [11], este valor médio de temperatura está acima do estipulado caso este fosse um local de trabalho.

Na Figura 4.8 apresentam-se os valores de humidade, enviados em conjunto com os valores de temperatura. A humidade máxima foi de 84,4% e a mínima de 64,6%, tendo uma média de 77,8% neste intervalo de tempo. Os valores mais baixos foram atingidos entre as 07:00 e as 9:00 e os mais altos entre as 13:00 e as 16:00.

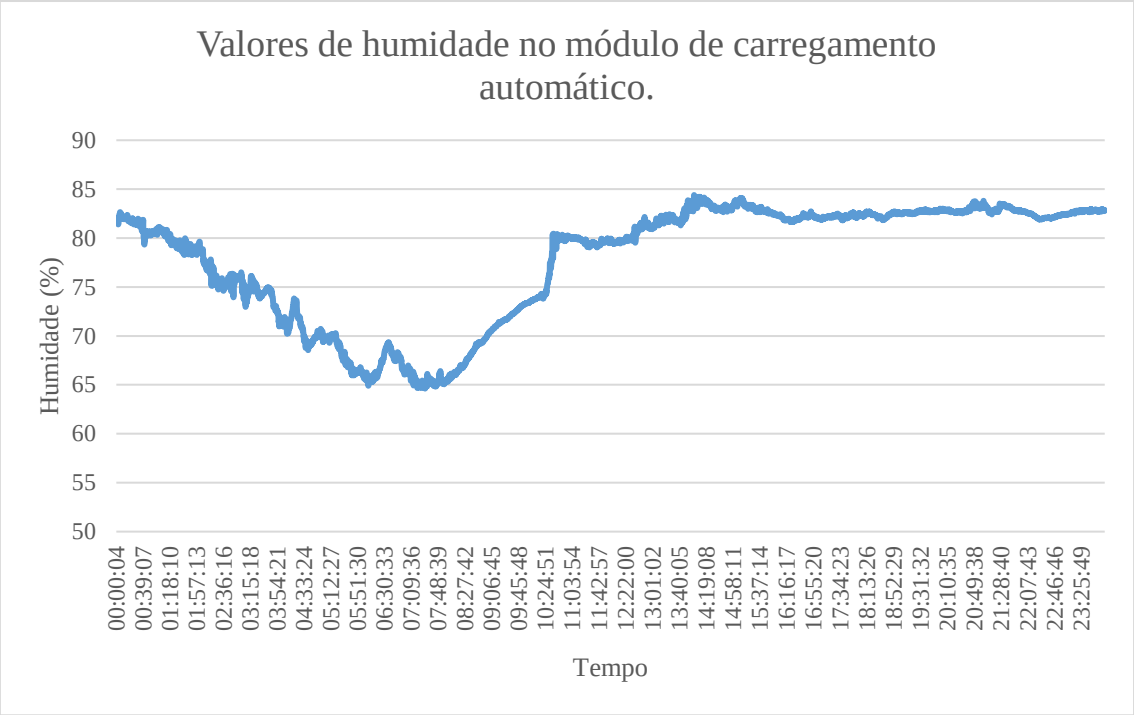


Figura 4.8 - Gráfico com valores de humidade durante 24 horas (dia 25/08/2017) para o módulo de carregamento automático.

De acordo com [11], este valor médio de humidade está acima do estipulado caso este fosse um local de trabalho.

Sabendo que este módulo foi desenvolvido com a funcionalidade de carregar automaticamente, e estando o transformador embutido no dispositivo, foi importante analisar a influência do carregamento nos valores de temperatura e humidade. Na Figura 4.9, observa-se que o valor da temperatura aumenta cerca de 0,5ºC quando bateria começa a ser carregada até ao momento em que o carregamento está completo. A temperatura depois baixa.

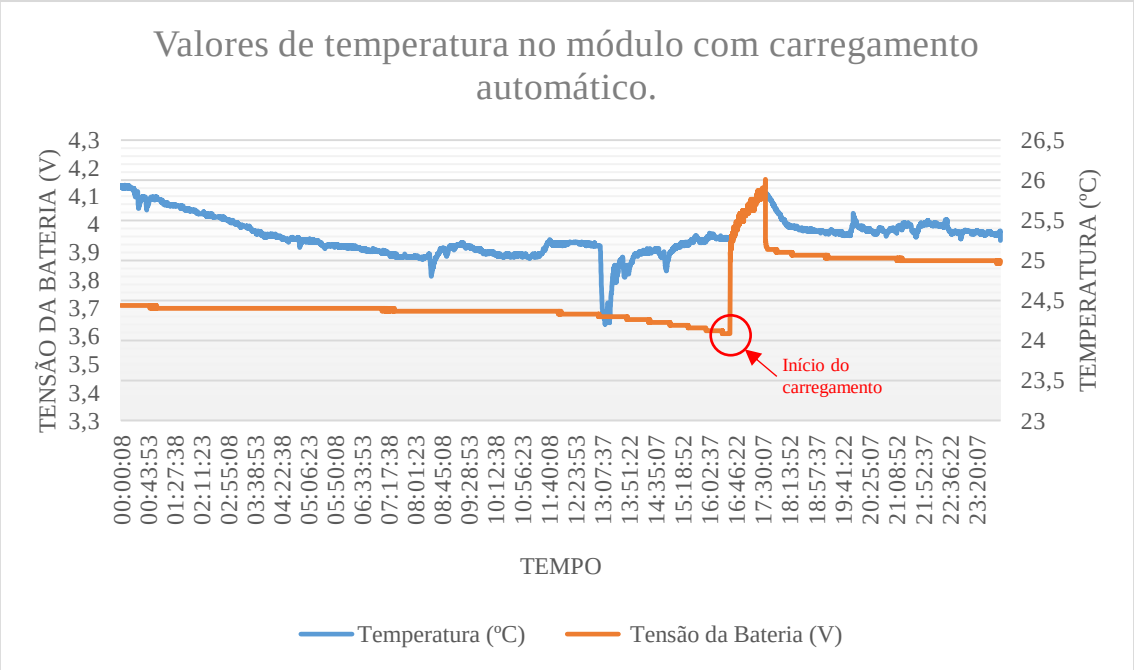


Figura 4.9 - Influência do carregamento na temperatura medida.

Constatou-se que seria possível diminuir a influência na temperatura e humidade por parte do carregamento da bateria, afastando o sensor do módulo, como a Figura 4.10 apresenta.

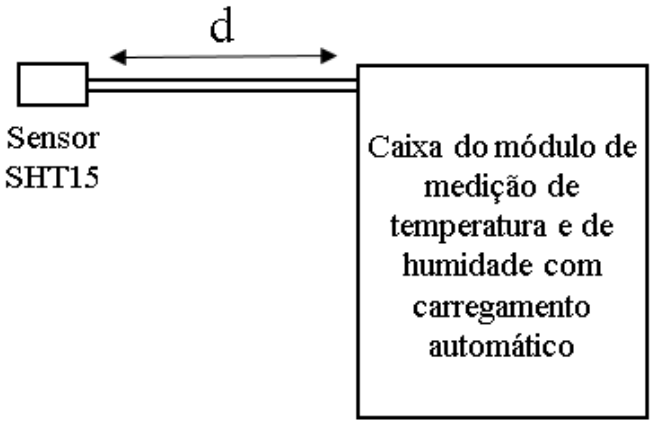


Figura 4.10 - Esquema para colocação do sensor SHT15.

Foram efetuados vários testes para verificar qual seria a melhor forma de implementar este sensor na caixa desenvolvida, sem que o carregamento afetasse em demasia os valores de temperatura medidos. Na Figura 4.10 apresenta-se a variável *d* que representa a distância entre o sensor e a caixa, sendo que para uma distância de 10 cm o carregamento afetava em cerca de 0,1 ºC, para uma distância de 3 cm o carregamento afetava em cerca de 0,5 ºC e para uma distância de 0 cm, o sensor era afetado por um valor superior a 1ºC. O módulo testado tinha o valor de distância de 3 cm, tendo em conta o compromisso entre o tamanho do protótipo e da influência dos valores medidos.

Na Figura 4.11 apresenta-se a visualização dos dados na plataforma *online* ao longo de 8 dias. Na Figura 4.11 a), apresenta-se a variação da temperatura e na Figura 4.11 b) apresenta-se o valor da tensão da bateria verificando-se dois carregamentos.

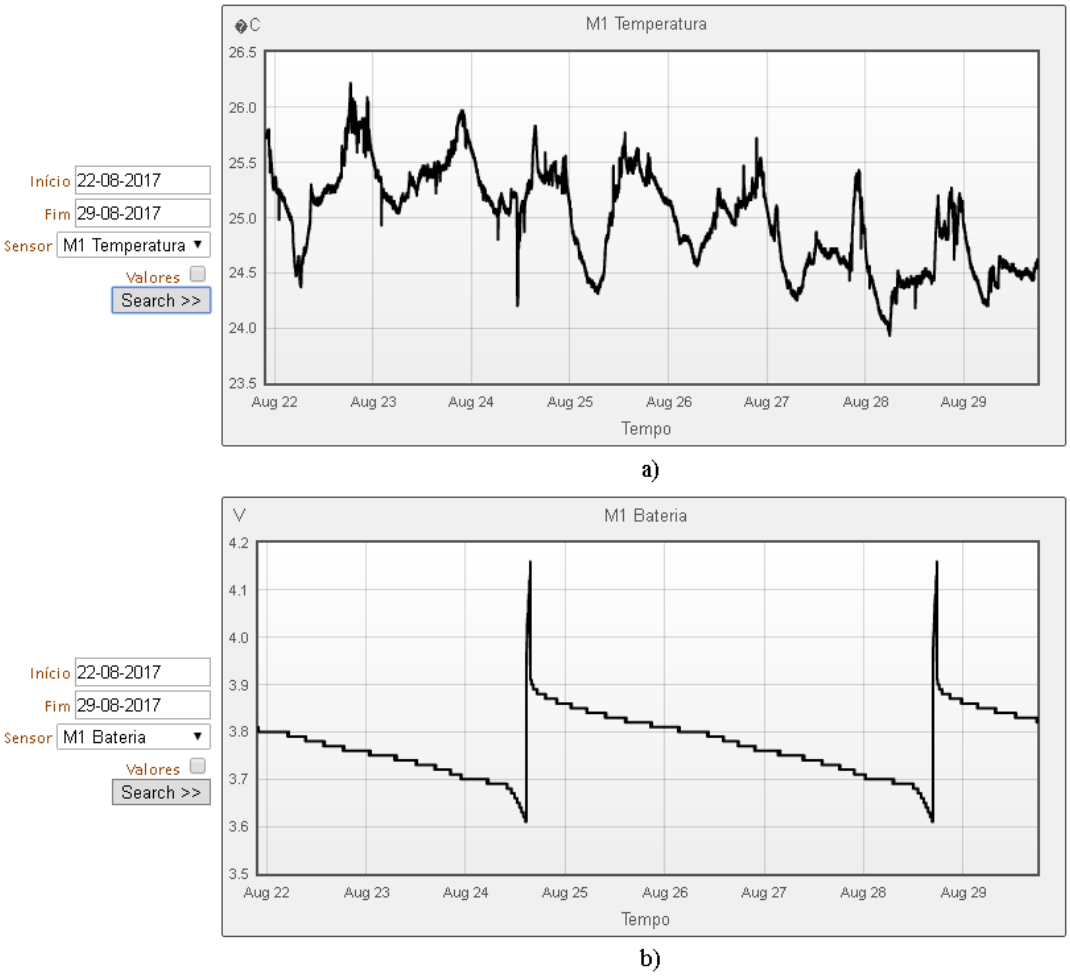


Figura 4.11 – a) Valores de temperatura; b) valores da tensão da bateria; ao longo de 8 dias, visualizados na plataforma *online*.

Pode-se, também, verificar que a bateria em utilização tinha autonomia para cerca de 4 dias, sendo que este módulo tinha um consumo que rondava os 20 mA.

4.5. Testes de medição de temperatura e humidade

A medição de temperatura e humidade utilizando o módulo sem carregamento automático foi efetuada num espaço comum de uma habitação. Optou-se por avaliar a variação da temperatura e da humidade ao longo de 24 horas e, depois, ao longo de 7 dias.

Na Figura 4.12 apresentam-se os valores de temperatura, durante 24 horas, do dia 25/08/2017. A temperatura máxima foi de 25,1 ºC e a mínima de 23,8 ºC, tendo uma média de 24,5 ºC neste intervalo de tempo. Os valores mais baixos foram atingidos entre as 07:00 e as 10:00 e os mais altos entre as 17:00 e as 19:00.

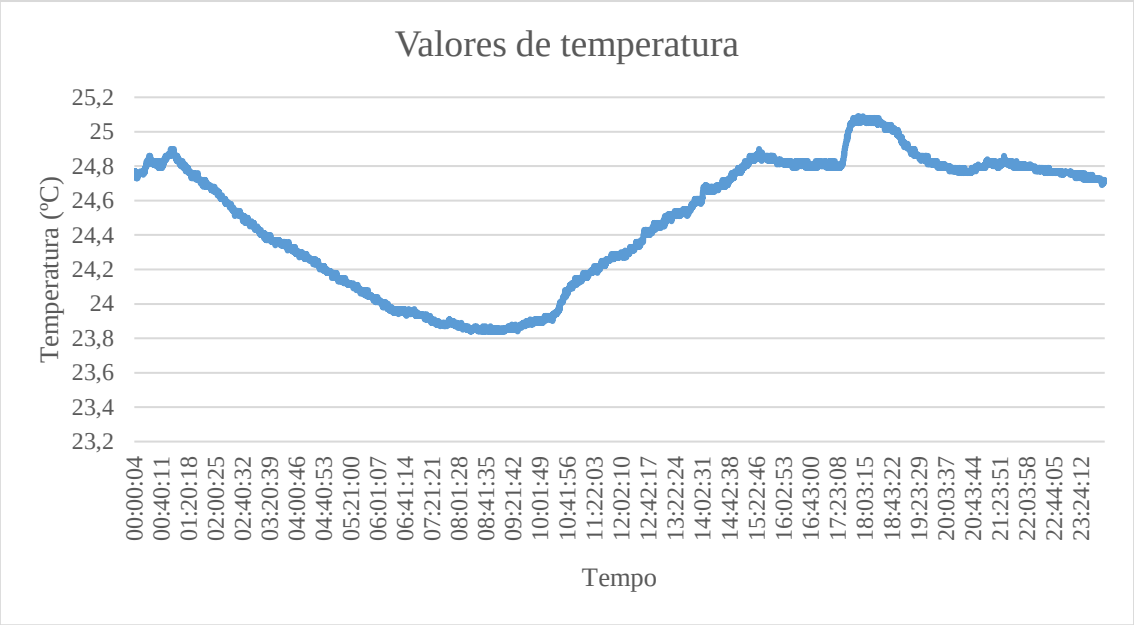


Figura 4.12 - Valores de temperatura no módulo de medição de temperatura e humidade.

De acordo com [11], este valor médio de temperatura está dentro do estipulado caso este fosse um local de trabalho.

Na Figura 4.13 apresentam-se os valores de humidade, enviados em conjunto com os valores de temperatura. A humidade máxima foi de 77,7% e a mínima de 58,1%, tendo uma média de 70,8% neste intervalo de tempo. Os valores mais baixos foram atingidos entre as 07:00 e as 08:00 e os mais altos entre as 13:00 e as 15:00.

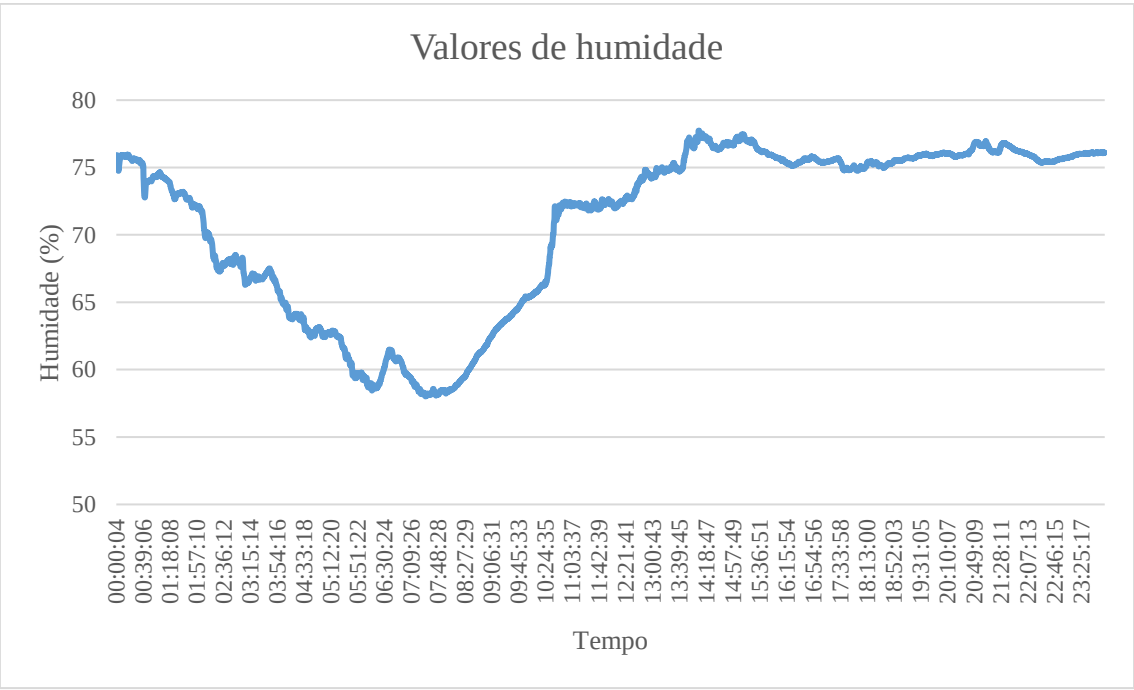


Figura 4.13 - Valores de humidade no módulo de medição de temperatura e humidade.

De acordo com [11], este valor médio de humidade está acima do estipulado caso este fosse um local de trabalho.

Na Figura 4.14, apresenta-se a visualização dos dados na plataforma *online* ao longo de 7 dias. Na Figura 4.11 a) apresenta-se a variação da temperatura e na Figura 4.11 b) apresenta-se o valor da humidade.

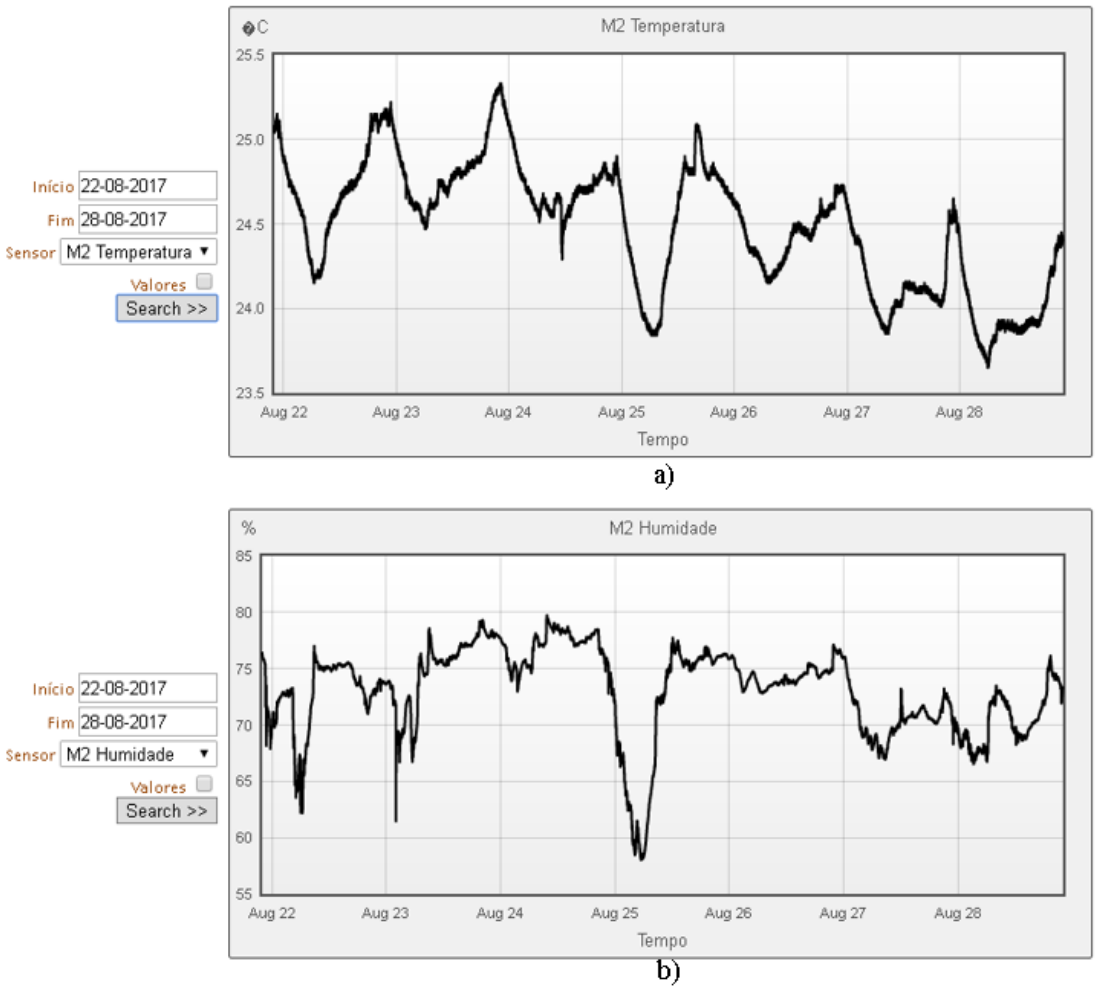


Figura 4.14 - a) Valores de temperatura; b) valores da humidade ao longo de 7 dias, visualizados na plataforma *online*.

O módulo tinha um consumo de, aproximadamente, 10 mA.

4.6. Testes de medição de temperatura, humidade, CO_2, COV e partículas em suspensão

O módulo de medição de temperatura, de humidade, de CO_2, de COV e de partículas em suspensão foi testado no mesmo espaço que os módulos anteriores. Na Figura 4.15 apresentam-se os valores de temperatura para um período de 24 horas.

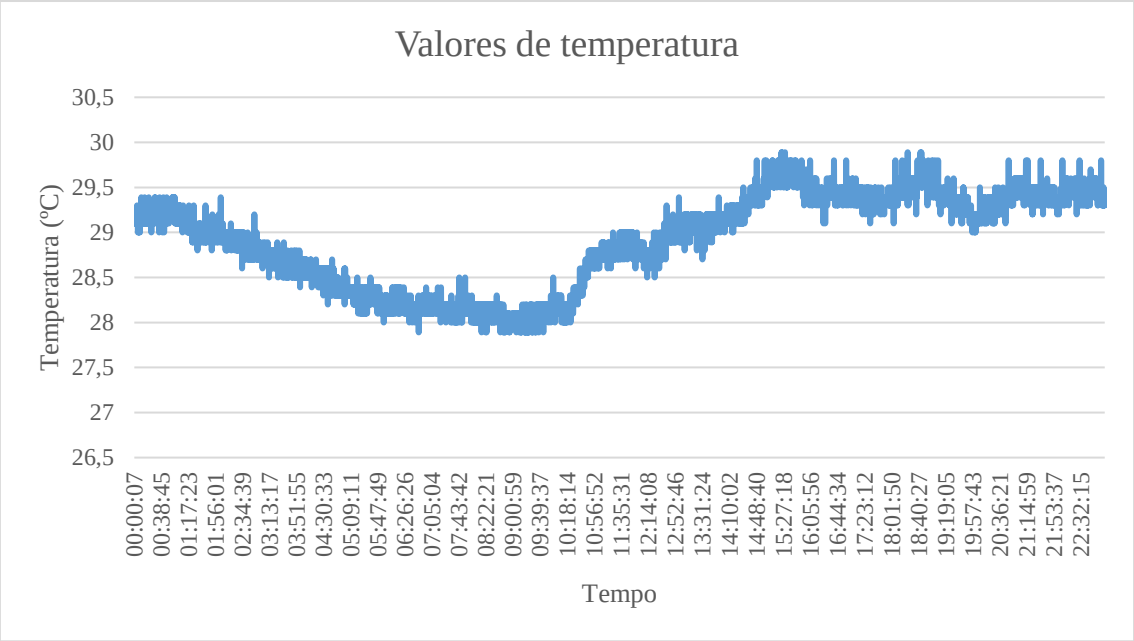


Figura 4.15 - Valores de temperatura para um período de 24 horas, medidos através do módulo com os diversos sensores.

Os dados são do dia 25/08/2017, sendo que a temperatura máxima foi de 29,9 ºC e a mínima de 27,9 ºC, tendo uma média de 28,9 ºC. Os valores mais baixos foram atingidos entre as 07:00 e as 10:00 e os mais altos entre as 14:00 e as 15:00, aproximadamente. De acordo com [11], este valor médio de temperatura está acima do estipulado caso este fosse um local de trabalho.

Na Figura 4.16 apresentam-se os valores de humidade, enviados em conjunto com os valores de temperatura. A humidade máxima foi de 64,9 % e a mínima de 45,8 %, tendo uma média de 58,5 %. Os valores mais baixos foram atingidos entre as 07:00 e as 08:00 e os mais altos entre as 13:00 e as 14:00, aproximadamente.

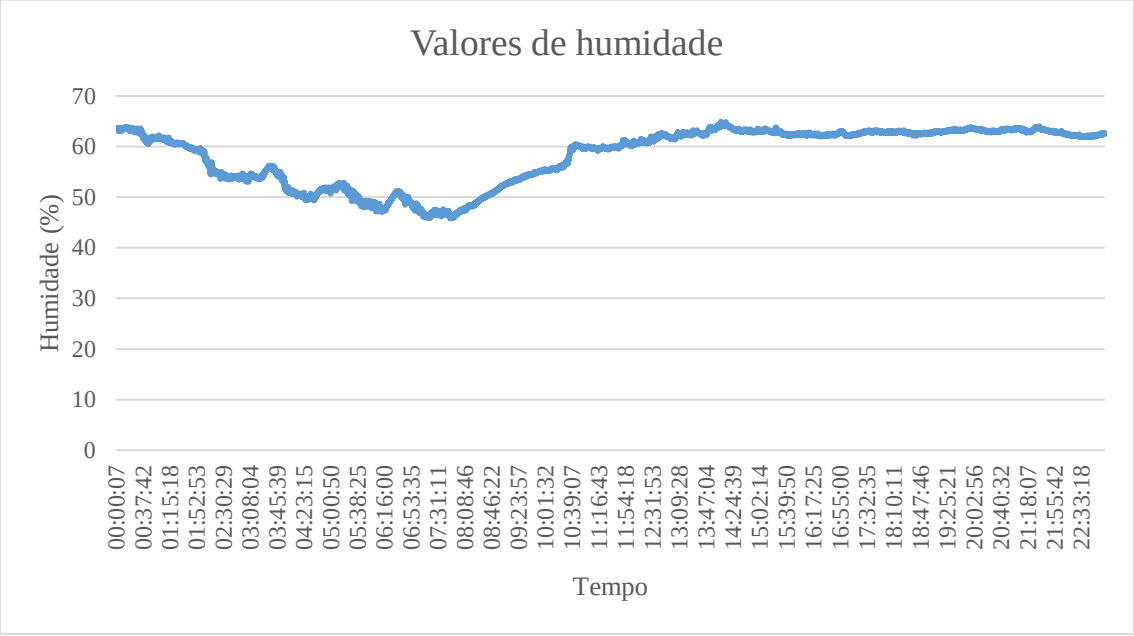


Figura 4.16 - Valores de humidade para um período de 24 horas, medidos através do módulo com os diversos sensores.

De acordo com [11], este valor médio de humidade está dentro do estipulado caso este fosse um local de trabalho.

Na Figura 4.17 apresenta-se a monitorização do dióxido de carbono nesse mesmo espaço.

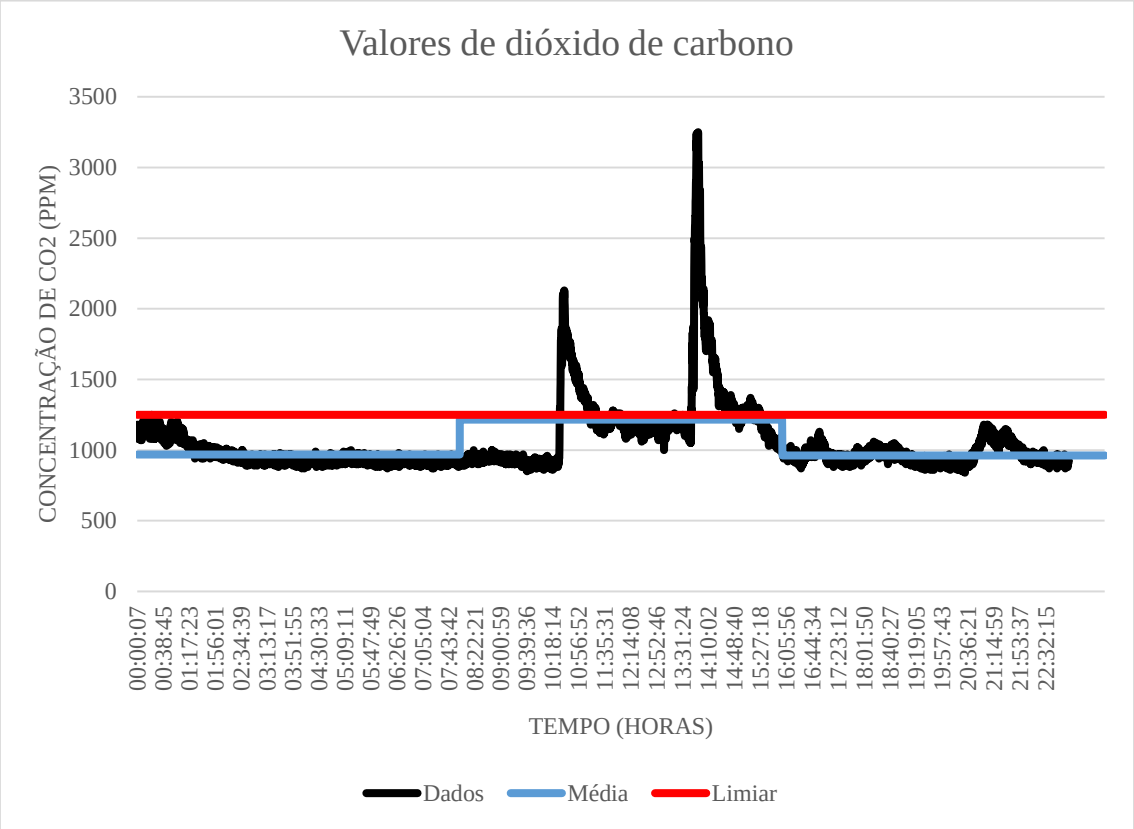


Figura 4.17 - Valores de dióxido de carbono em 24 horas.

Os dados são do dia 25/08/2017, sendo que a concentração máxima foi de 3250 ppm e a mínima de 840 ppm, tendo uma média de 1048 ppm. Os valores mais altos foram atingidos entre as 14:00 e as 15:00 e entre as 10:00 e as 11:00, aproximadamente. É de referir que a presença de pessoas perto do sensor fez com que surgissem estes momentos de maior concentração de CO_2. A linha azul representa a média num intervalo de 8 horas e a linha vermelha representa o limiar definido na legislação portuguesa no interior de edifícios, verificando-se assim que os valores de concentração de CO_2 medidos estavam dentro do que é considerado um ambiente saudável. Verificou-se, no entanto, que a média de concentração é superior à apresentada em [9], ou seja, estamos perante uma concentração média que é superior à média das habitações portuguesas. É possível ter uma concentração mais baixa, se o espaço em questão for mais arejado. Na Figura 4.18 apresenta-se o gráfico obtido durante 5 dias consecutivos.

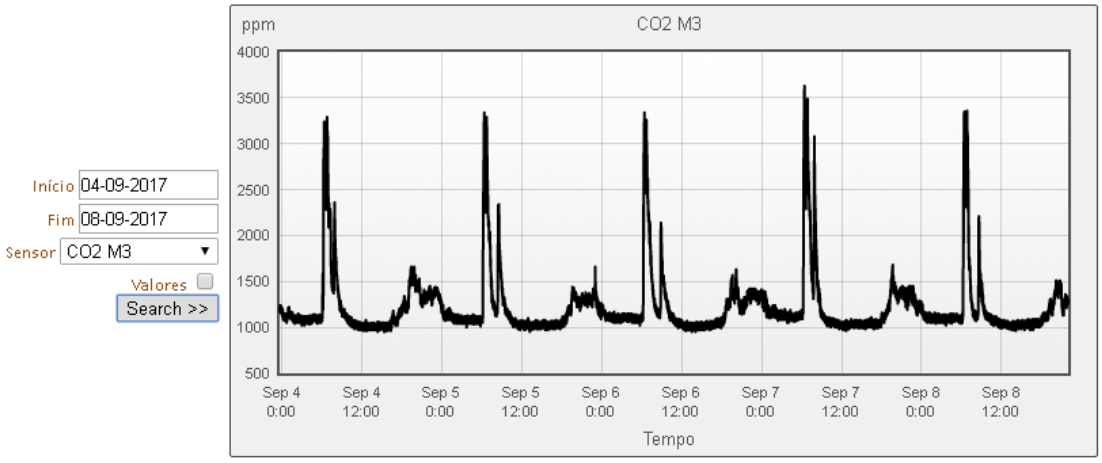


Figura 4.18 - Concentração de dióxido de carbono durante 5 dias consecutivos.

Escolheram-se estes dias porque eram dias de trabalho em que se nota a rotina diária das pessoas desta habitação. O pico maior acontecia sempre durante a manhã, antes da saída para o trabalho, e ao final do dia, quando as pessoas voltam para a habitação.

Realizaram-se outros dois testes em dias diferentes, com duas condições diferentes, em que duas pessoas dormiram num quarto desta habitação. Os resultados, das primeiras 24 horas, estão na Figura 4.19.

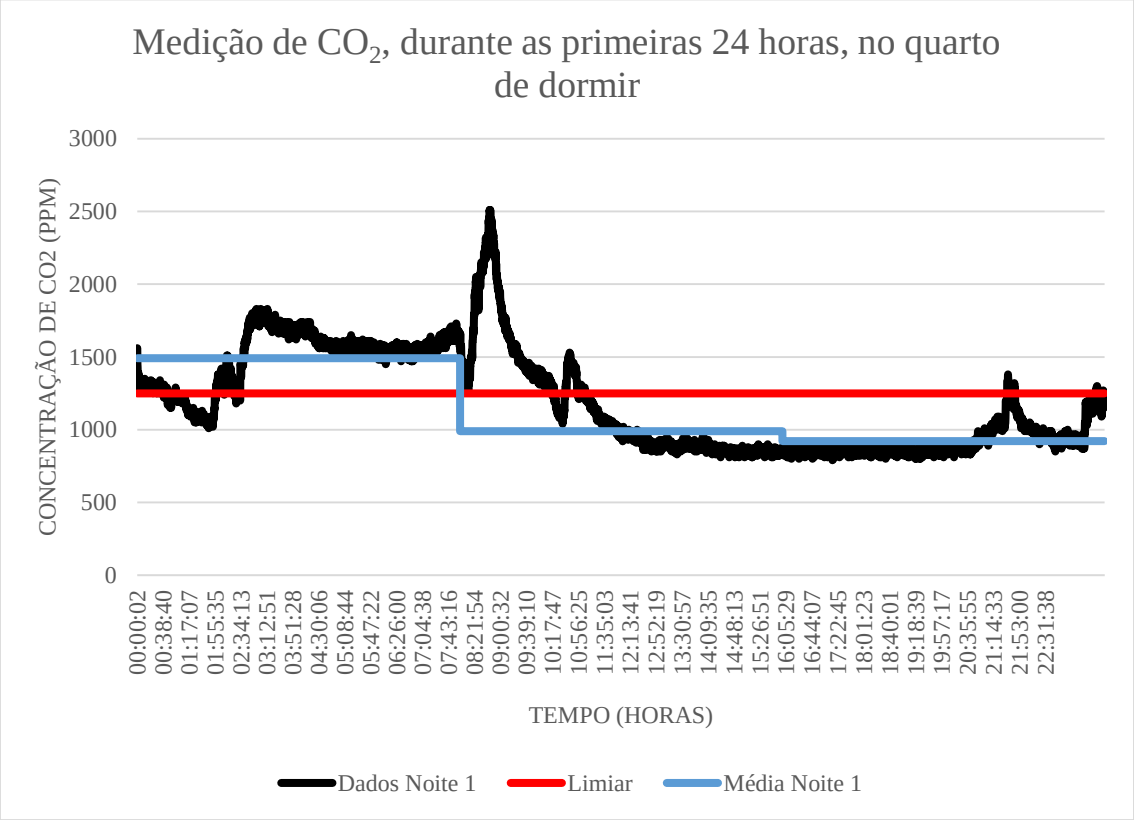


Figura 4.19 - Medição de CO_2, durante as primeiras 24 horas, no quarto de dormir.

No primeiro teste, em que os resultados no gráfico estão assinalados a preto, apresentam-se 24 horas de medição num quarto de dormir em que a janela tem uma rede para arejamento do espaço. A concentração máxima foi de 2510 ppm e a mínima de 790 ppm, tendo uma média, das 24 horas, de 1193 ppm. A linha azul representa a média num intervalo de 8 horas e a linha vermelha representa o limiar definido na legislação portuguesa e verifica-se que, mesmo com arejamento do espaço, as primeiras 8 horas têm uma média superior à legislada.

Os resultados, das segundas 24 horas, estão na Figura 4.20.

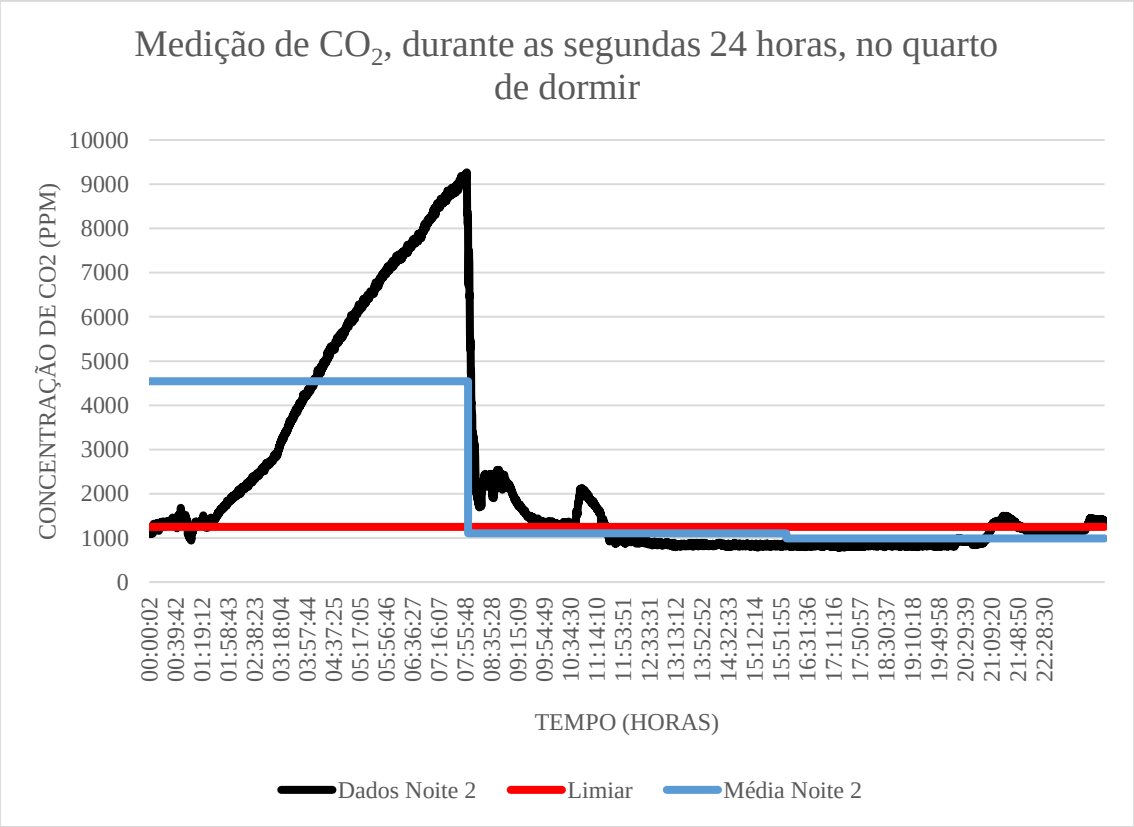


Figura 4.20 - Medição de CO_2, durante as segundas 24 horas, no quarto de dormir.

Já no segundo teste a janela esteve totalmente fechada durante a noite, obtendo-se uma concentração máxima de 9270 ppm e uma mínima de 770 ppm, tendo uma média, das 24 horas, de 2279 ppm. Os resultados obtidos entre as 00:00 e as 08:00 são completamente diferentes, sendo que no segundo teste as medições de CO_2 atingiram valores muito mais elevados que no primeiro teste. Às 08:00, no segundo teste, a janela foi aberta e a concentração desceu abruptamente.

Nas Figura 4.21 e Figura 4.22 apresentam-se os resultados para as medições de partículas em suspensão, PM_{10} e $PM_{2,5}$, respetivamente.

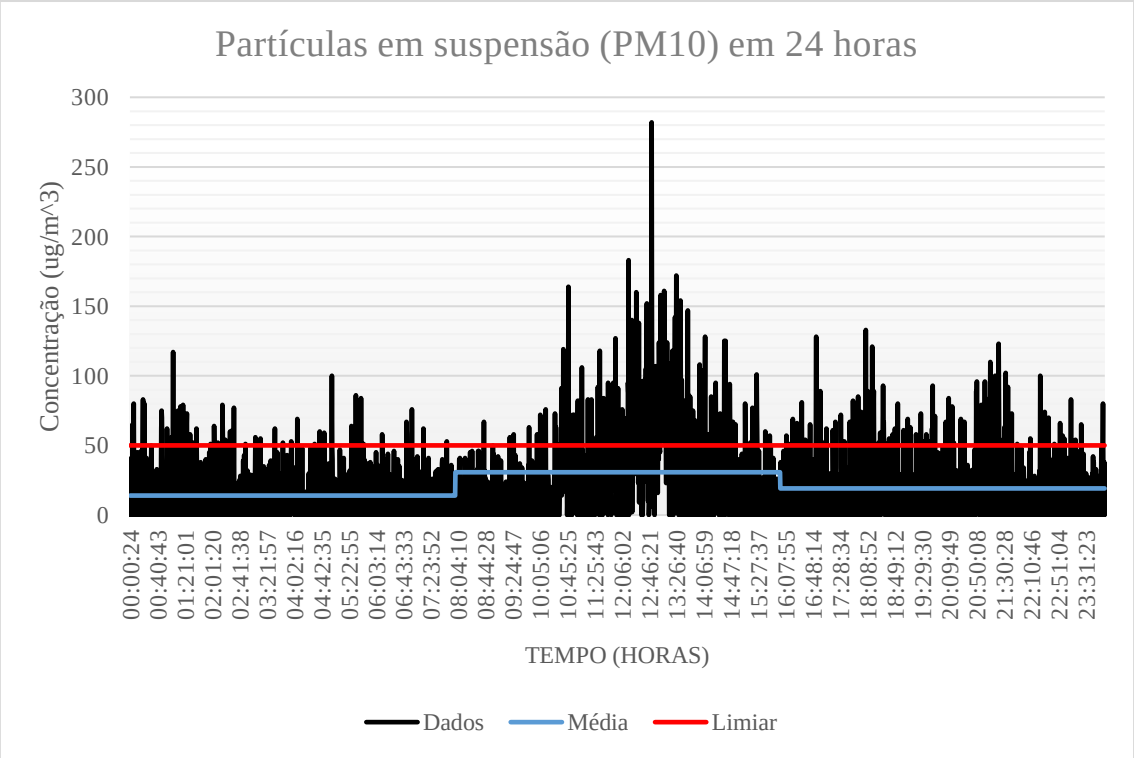


Figura 4.21 - Partículas em suspensão PM_{10} em 24 horas.

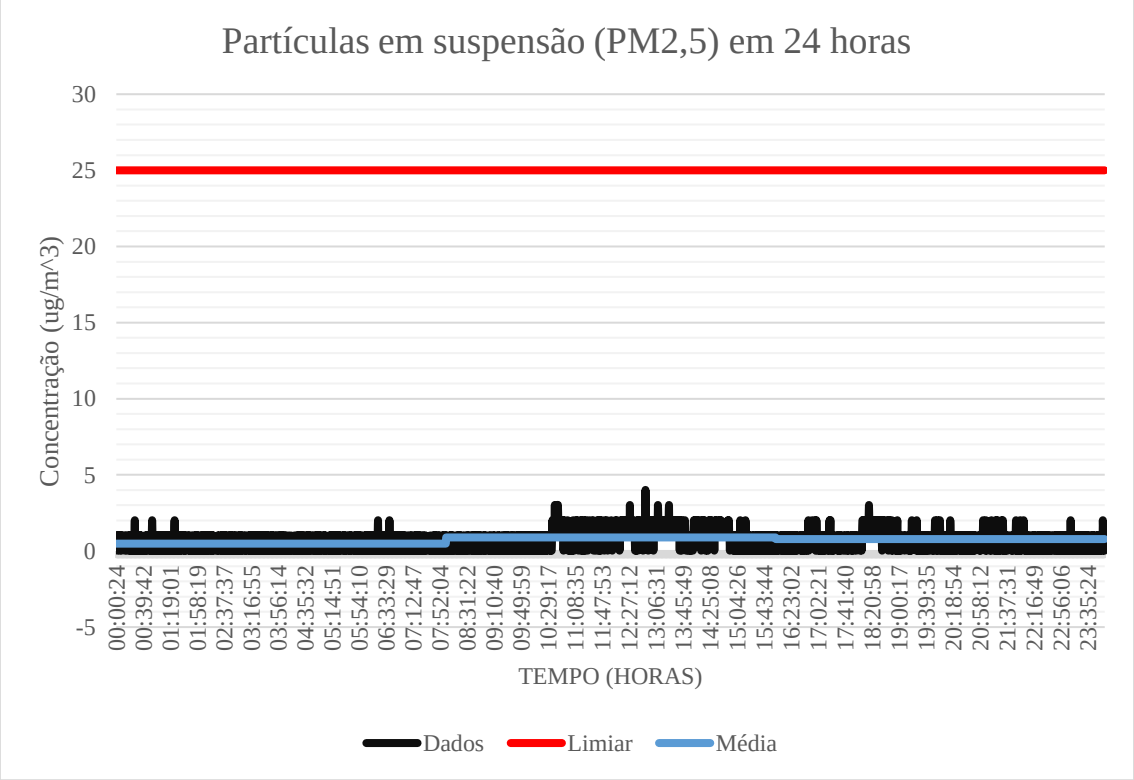


Figura 4.22 - Partículas em suspensão $PM_{2,5}$ em 24 horas.

Para as partículas em suspensão PM_{10} tem-se uma média de 21,24 µg/m^3 com um pico máximo de 282,02 µg/m^3. A linha azul representa a média num intervalo de 8 horas e a

linha vermelha representa o limiar definido na legislação portuguesa e verifica-se que esta, durante as 24 horas, está abaixo desse limiar. No que diz respeito às partículas em suspensão $PM_{2,5}$ tem-se uma média de 0,71 µg/m^3, com um pico máximo de 4,02 µg/m^3. Ambos os valores estão abaixo do limiar fixado na Tabela 2.2, sendo que a média obtida está abaixo do estudo efetuado em [9]. No entanto decidiu-se verificar a resposta do sensor num ambiente com fumo de cigarro. Na Figura 4.23 apresentam-se os resultados do teste efetuado.

Figura 4.23 - Medição de partículas em suspensão na presença de fumo de cigarro.

Observa-se claramente que existe um pico causado pela presença do fumo de cigarro, fazendo com que os níveis de concentração de partículas em suspensão atinjam facilmente valores muito elevados.

Por fim, apresenta-se a análise ao sensor de compostos orgânicos voláteis. Este sensor de baixo custo não permite ter valores de concentração, sendo sensível a múltiplos gases, não conseguindo diferenciar quais destes estão presentes. Este sensor será mais indicado para detetar alterações na densidade do gás presente e por isso os resultados aqui apresentados estão em percentagem, tendo em conta o máximo valor para o qual o sensor consegue dar resposta. Será também importante comparar os valores de COV com os de dióxido de carbono. Na Figura 4.24 apresentam-se os resultados obtidos.

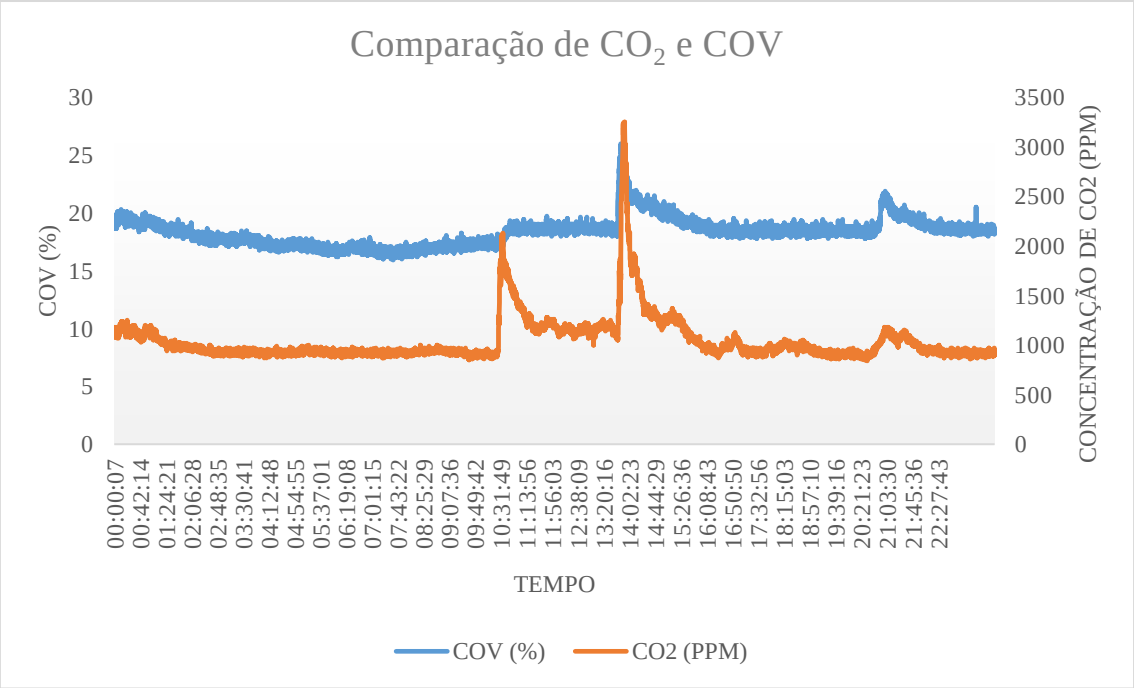


Figura 4.24 - Comparação de CO2 e COV.

A laranja apresentam-se os resultados apresentados anteriormente de CO2 comparados com os de COV, a azul. O valor de COV médio é de 18% com um máximo de 25,97%. É importante notar que existem uns picos que coincidem com os picos de concentração de CO_2 podendo-se afirmar que o sensor reagia a diferentes densidades de gases presentes.

Este módulo tinha um consumo de, aproximadamente, 200 mA.

4.7. Comparação de resultados entre módulos (temperatura e humidade)

Tendo em conta que as medições de temperatura e humidade, nos 3 diferentes módulos, foram efetuadas no mesmo espaço e dia, pôde-se comparar as diferenças entre cada. Na Figura 4.25 apresenta-se a comparação entre os módulos.

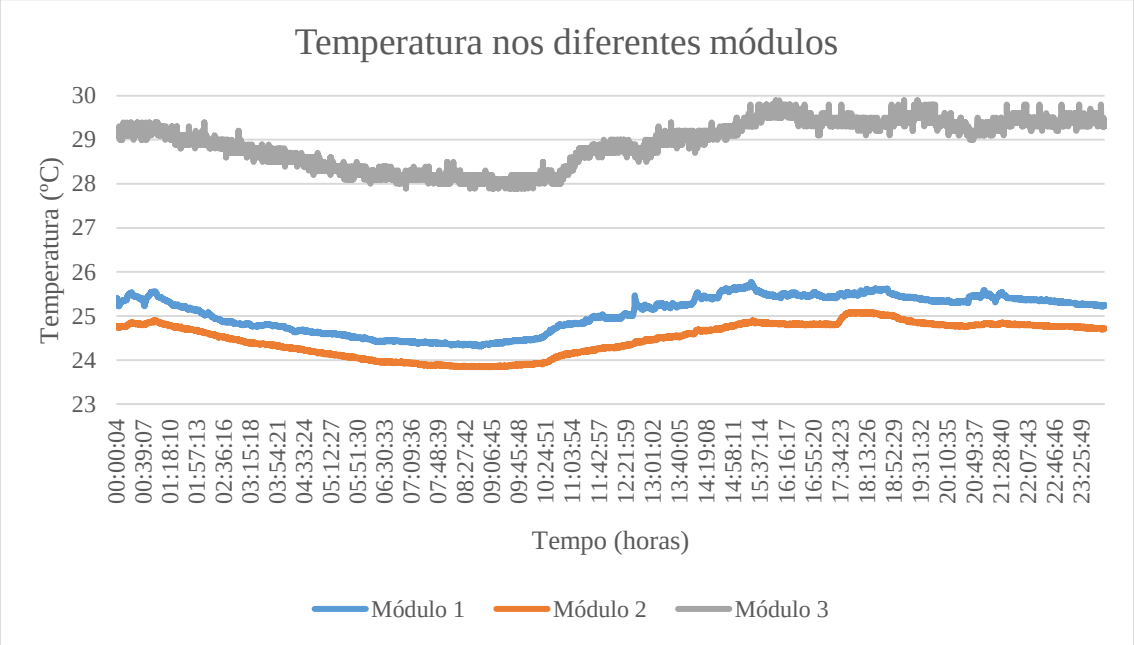


Figura 4.25 - Comparação de temperatura nos diferentes módulos.

O Módulo 1 tratava-se do que possuía o carregamento automático, o Módulo 2 não tinha carregamento automático e o Módulo 3 era o que tinha os sensores de CO_2, COV e partículas em suspensão juntos num só dispositivo. Verifica-se que o comportamento é semelhante ao longo do dia, mas existem diferenças consideráveis se se analisar num determinado instante. Os Módulos 1 e 2 que possuem os sensores SHT15 tem valores com uma diferença máxima de 0,5 ºC mas no que toca ao sensor do Módulo 3, DHT22, a diferença chega aos 4 ºC, sendo este último um valor muito elevado. O DHT22 está próximo de outros sensores que geram calor para o seu funcionamento (como é o caso do sensor de partículas em suspensão e de COV). Uma das formas que se poderia utilizar para melhorar estes resultados seria o afastamento do sensor de temperatura e de humidade da caixa, tal como foi feito para o módulo de medição de temperatura e humidade com carregamento automático.

Na Figura 4.26 apresenta-se a comparação dos valores de humidade para os mesmos módulos.

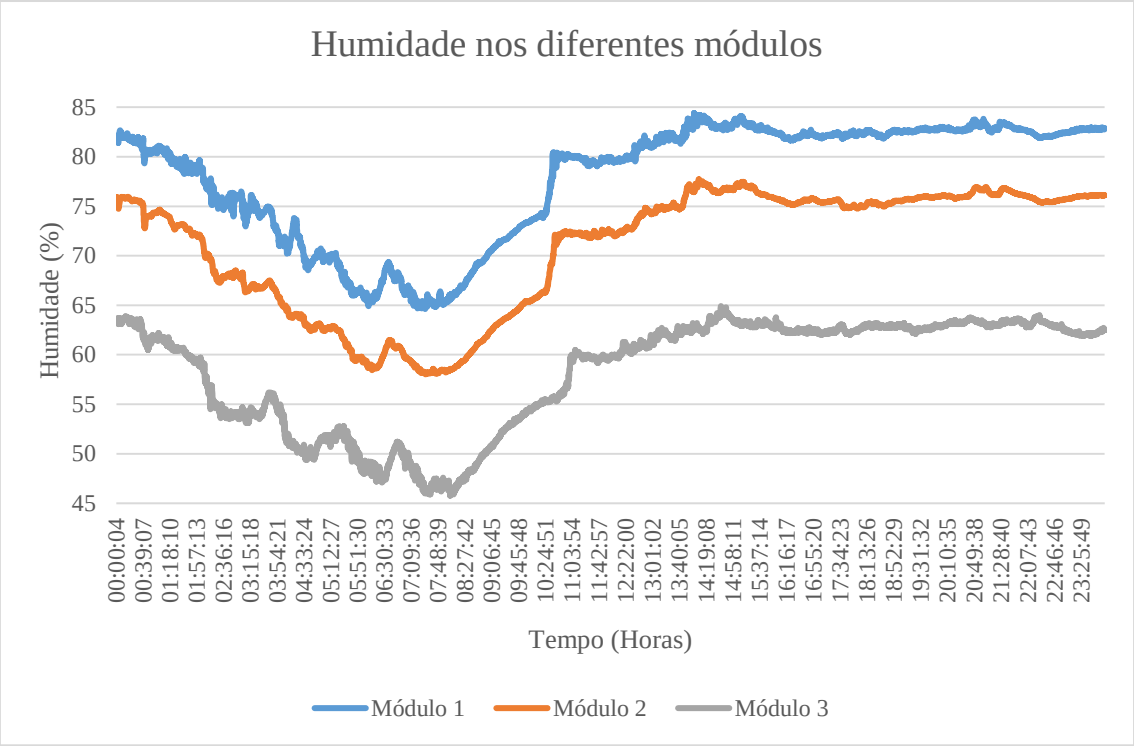


Figura 4.26 - Comparação da humidade nos diferentes módulos.

Verifica-se que a diferença entre os valores de humidade também aconteceu, tendo o mesmo comportamento ao longo do dia. A diferença entre o módulo 1 e o 2 é de cerca de 6% enquanto a diferença entre o módulo 3 e o 2 é de cerca de 12%.

4.8. Testes de alcance

Realizaram-se testes ao alcance dos nós terminais em relação ao nó coordenador. Tendo em conta que foi desenvolvida uma antena era importante verificar os testes de RSSI e cada um dos módulos enviava, em determinados intervalos de tempo o valor de RSSI em dBm. A antena utilizada para comparação foi um monopolo de 2 dBi de ganho.

De acordo com [67] a melhor forma desta antena estar posicionada dentro de uma caixa é a ilustrada na Figura 4.27.

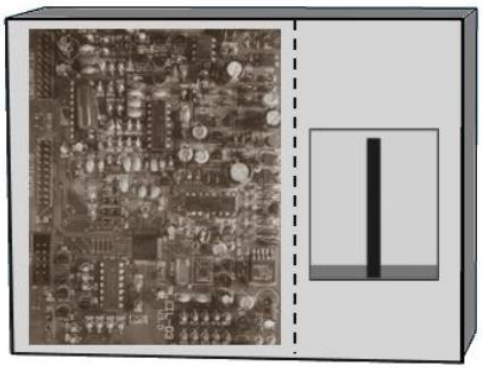


Figura 4.27 - Representação da posição da antena na caixa *[67]*.

Assim, colocou-se o módulo coordenador num andar inferior ao do módulo *router*, com a antena construída, e foram sendo efetuadas medições de RSSI durante algumas horas. Posteriormente alterou-se a antena para a de comparação e fez-se o mesmo procedimento. A média de valores de RSSI para a antena construída deu, aproximadamente, -81 dBm, enquanto para a outra antena obteve-se um resultado de -71 dBm.

A distância direta ao módulo coordenador deveria rondar os 6 metros, mas é necessário ter em conta que muitos obstáculos estavam presentes (paredes, chão, armários, portas). Outro dos fatores que é necessário ter em conta é que a antena construída estava dentro da caixa deste módulo, ao contrário da antena de comparação que não cabia dentro da caixa fechada, sendo esta menos afetada.

Cada um dos módulos, em determinados intervalos de tempo, fazia um pedido de RSSI para que fosse possível verificar a qualidade da ligação. Na Figura 4.28 apresenta-se o valor de RSSI para o módulo de medição de corrente no quadro elétrico ao longo de 24 horas.

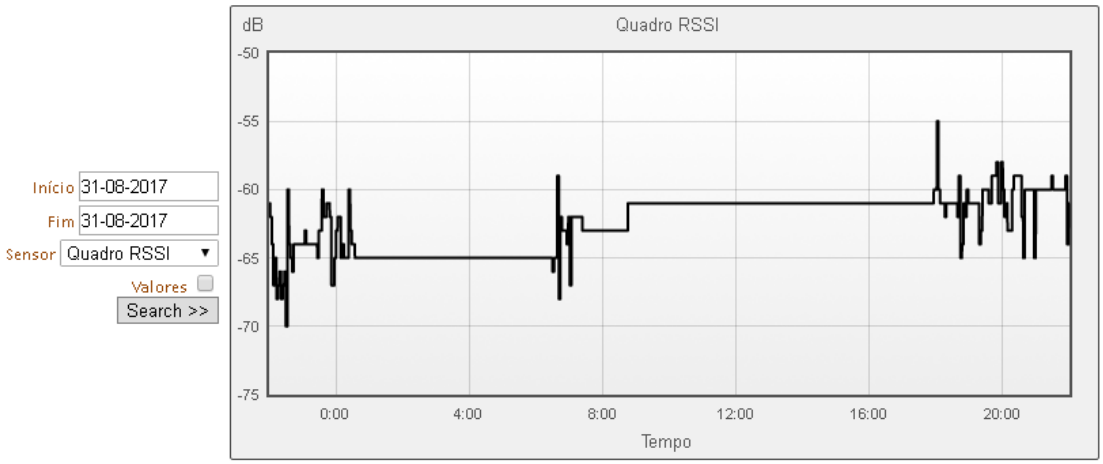


Figura 4.28 - Resultados do RSSI ao longo de 24 horas para o módulo de medição de corrente no quadro elétrico.

Como se pode observar, apesar do módulo estar sempre no mesmo local ao longo do dia, o valor RSSI era instável devido aos obstáculos móveis e à presença humana.

5. Conclusões finais

Neste capítulo estão apresentadas as conclusões do estudo realizado neste trabalho e são dadas algumas indicações de trabalhos que poderão ser feitos futuramente no sentido de melhorar e continuar a desenvolver este tema.

5.1. Conclusão

Neste trabalho desenvolveu-se uma rede de sensores sem fios para a monitorização de diversas variáveis de interesse numa habitação, com o objetivo de proporcionar informação ao utilizador acerca da qualidade do ar interior e dos consumos de energia.

A monitorização mostrou ser importante para identificar problemas na qualidade do ar interior, que muitas vezes pequenas ações, como arejar um espaço como um quarto ou uma sala, podem fazer uma diferença enorme. Tendo esta informação de dados das medições, os utilizadores poderão tomar medidas.

Estudaram-se diversas opções existentes no mercado a nível da monitorização de qualidade do ar interior e do consumo elétrico, tendo em conta a forma como estas medições eram efetuadas. Foi feita também uma pesquisa sobre trabalhos existentes na área das redes sensores sem fios.

Com base nos estudos em relação a variáveis a monitorizar numa habitação definiu-se que os parâmetros a serem medidos fossem a temperatura, a humidade, o dióxido de carbono, os compostos orgânicos voláteis, as partículas em suspensão e, para a parte do consumo de energia, a medição da corrente elétrica.

Relativamente aos módulos desenvolvidos, estudaram-se duas formas de carregamento de bateria. O módulo de carregamento automático tinha a principal vantagem de não ser necessário haver preocupação com o carregamento da bateria, o que acontecia com o módulo que não tinha carregamento automático. Ao optar por um dispositivo único, tem-se que embutir o transformador no interior do módulo e no momento do carregamento poderá acontecer influência no valor das medições, devido à potência dissipada em forma de calor, o que obriga a ter algum cuidado na inserção dos sensores na caixa de monitorização.

Analisaram-se os dados obtidos de temperatura, de humidade, de COV, de CO2 e de partículas em suspensão e comparou-se com outros estudos efetuados e com os limiares que estabelecem a qualidade do ar interior.

Conseguiu-se implementar também um dispositivo de leitura de corrente de pequena dimensão, simples e de baixo consumo para ser colocado num quadro elétrico para que este possa medir o consumo de toda a energia consumida na habitação. Desta forma poder-se-á detetar anomalias e consumos excessivos. No que diz respeito à tomada elétrica é possível ter uma maior noção dos consumos de dispositivos como portáteis e carregadores de telemóvel, que consumem correntes mais baixas, mas que ao final de algum tempo, acabam por ter um valor significativo de consumo. Aproveitou-se o facto de estar constantemente ligado a uma tomada para não utilizar uma bateria e utilizar este módulo como um nó *router*, para expandir a cobertura da rede sem fios.

A cobertura da rede sem fios é importante dado que numa habitação existem uma grande quantidade de obstáculos que degradam o sinal, podendo prejudicar a comunicação entre nós. Uma antena foi desenvolvida com o intuito de ser pequena e eficaz, e analisou-se o RSSI.

A técnica de adormecimento do microcontrolador e do dispositivo de comunicação é muito útil para um consumo mais eficiente, dando maior autonomia a cada um dos terminais.

Quanto ao módulo coordenador conseguiu-se adaptar a plataforma existente para funcionar no *Raspberry Pi*, um computador de baixo custo e de baixo consumo que providencia as características necessárias para servir de módulo coordenador, receber os dados, armazenar na base de dados e disponibilizar uma plataforma *web* para que o utilizador possa aceder às medições, não só em tempo real, mas também aos dias anteriores.

Pode-se concluir que a implementação de uma rede de sensores sem fios para monitorizar dados numa habitação, disponibilizando essas medições ao utilizador a qualquer momento, permite que esse fique mais apto para tomar medidas que visam beneficiar o próprio, quer a nível da qualidade do ar interior, quer ao nível do consumo energético.

5.2. Trabalhos futuros

Poder-se-ia propor como trabalhos futuros, estudos semelhantes a estes, em habitações maiores, em outros ambientes públicos fechados, como escolas ou em espaços empresariais onde exista a presença de pessoas em elevado número.

Na monitorização de habitação poder-se-á acrescentar outras variáveis que promovam a segurança à intrusão como, por exemplo, monitorização da abertura e fecho de portas e janelas ou de vigilância recorrendo a uma câmara (vídeo ou foto).

Também se pode utilizar energias renováveis, como energia solar, para o carregamento dos nós sensores.

Por fim, implementar uma plataforma *web* com mais funcionalidades como, por exemplo, alertar o utilizador que um dos nós deixou de funcionar ou de alertar que a bateria está descarregada e também de efetuar *backups* automáticos. Outra possibilidade interessante seria poder aceder aos dados medidos na habitação fora da rede local.

Bibliografia

[1] L. Antonio, L. Reyes, L. Antonio, L. Juan, e C. Leaño, *Certus - Revista Electrónica de Postgrado e Investigación*, 2007.

[2] US-EPA, «Indoor Air Pollution: An introduction for health professionals», pp. 13–14, 1996.

[3] «WHO | Household (Indoor) Air Pollution», *Who*, 2016. [Em linha]. Disponível em: http://www.who.int/indoorair/en/. [Acedido: 01-Fev-2017].

[4] «You May Want To Read This Outside | HuffPost». [Em linha]. Disponível em: http://www.huffingtonpost.com/dr-claudiaaguirre/post_11876_b_9925008.html. [Acedido: 05-Fev-2017].

[5] Agência Portuguesa do Ambiente Laboratório Referência do Ambiente, «Qualidade do Ar em Espaços Interiores-Um Guia Técnico», pp. 1–53, 2009.

[6] Portaria n. 353-A/2013, «(Regulamento de Desempenho Energético dos Edifícios de Comércio e Serviços (RECS) - Requisitos de Ventilação e Qualidade do Ar Interior)», *Diário da República*, vol. 1.ª série, n. 2, pp. 2–9, 2013.

[7] «Qualidade do AR | CCDR Algarve». [Em linha]. Disponível em: https://www.ccdr-alg.pt/site/info/qualidade-do-ar. [Acedido: 05-Fev-2017].

[8] D. Penney, V. Benignus, S. Kephalopoulos, D. Kotzias, M. Kleinman, e Agnes Verrier, «Guidelines for indoor air quality», *WHO Guidel.*, vol. 9, p. 454, 2010.

[9] J. Ginja, C. Borrego, M. Coutinho, C. Nunes, e M. Morais-Almeida, «Qualidade do ar interior nas habitações Portuguesas», *Congr. Innov. Sustain. Constr.*, pp. 1–10, 2012.

[10] S. Darby, «the Effectiveness of Feedback on Energy Consumption a Review for Defra of the Literature on Metering , Billing and», *Environ. Chang. Inst. Univ. Oxford*, vol. 22, n. April, pp. 1–21, 2006.

[11] Decreto de Lei n.º. 243/86, de 20 Agosto, *Diário da República* vol. I, n. 190. pp. 2099–20105, 1986.

[12] «Experimenting at Home With Air Quality Monitors - The New York Times». [Em linha]. Disponível em: https://www.nytimes.com/2015/04/16/business/experimenting-at-home-with-air-quality-monitors.html. [Acedido: 02-Fev-2017].

[13] «Carnegie Mellon University's Speck device monitors indoor pollution | TribLIVE». [Em linha]. Disponível em: http://triblive.com/news/allegheny/8008660-74/speck-particles-quality. [Acedido: 02-Fev-2017].

[14] AirBoxLab, «Foobot», *Features*, 2016. [Em linha]. Disponível em: https://foobot.io/foobotspecs.pdf.

[15] «Birdi | Tech». [Em linha]. Disponível em: https://birdihome.com/tech.html. [Acedido: 02-Fev-2017].

[16] «Full Specifications for the Netatmo Weather Station». [Em linha]. Disponível em: https://www.netatmo.com/en-US/product/weather/weatherstation/specifications.

[Acedido: 02-Fev-2017].

[17] «Awair Support». [Em linha]. Disponível em: https://support.getawair.com/hc/en-us. [Acedido: 02-Fev-2017].

[18] «Speck: Support: Technical Specifications». [Em linha]. Disponível em: https://www.specksensor.com/support/tech-specs. [Acedido: 03-Fev-2017].

[19] «CubeSensors - Feel better.» [Em linha]. Disponível em: https://cubesensors.com/#features. [Acedido: 01-Fev-2017].

[20] «Eve Room | elgato.com». [Em linha]. Disponível em: https://www.elgato.com/pt/eve/eve-room. [Acedido: 05-Fev-2017].

[21] «Current Transformer Basics and the Current Transformer». [Em linha]. Disponível em: http://www.electronics-tutorials.ws/transformer/current-transformer.html. [Acedido: 09-Fev-2017].

[22] Honeywell, «Hall Effect Sensing and Application», *Sens. Control*, p. 126, 2011.

[23] T. E. S. Worldwide, «TED Pro Energy Monitoring and Control System».

[24] «Eve Energy | elgato.com». [Em linha]. Disponível em: https://www.elgato.com/pt/eve/eve-energy. [Acedido: 03-Fev-2017].

[25] «Sense: Track energy use in real time». [Em linha]. Disponível em: https://sense.com/product.html. [Acedido: 02-Fev-2017].

[26] «Kill A Watt Meter - Electricity Usage Monitor | P3». [Em linha]. Disponível em: http://www.p3international.com/products/p4400.html. [Acedido: 05-Fev-2017].

[27] A. C. Frery, H. S. Ramos, J. Alencar-Neto, E. Nakamura, e A. a F. Loureiro, «Data driven performance evaluation of Wireless Sensor Networks.», *Sensors (Basel).*, vol. 10, n. 3, pp. 2150–68, 2010.

[28] M. FEZARI, M. S. BOUMAZA, A. Al-DAHOUD, e A. Al-DAHOUD, «WSN for AIR Quality Monitoring in Annaba City», *7th Int. Conf. Inf. Technol.*, vol. 2015, n. SEPTEMBER, pp. 283–288, 2015.

[29] S. Abraham e X. Li, «A cost-effective wireless sensor network system for indoor air quality monitoring applications», *Procedia Comput. Sci.*, vol. 34, pp. 165–171, 2014.

[30] V. Di Lecce, R. Daria, D. Politecnico, e J. Uva, «A Wireless Electronic Nose for Emergency Indoor Monitoring», n. c, pp. 274–279, 2011.

[31] J. Carlos e S. Barón, «Application of Zigbee Technology for Monitoring Enviromental Variables in Greenhouses». pp. 1384–1394.

[32] Y. Zhuang, F. Lin, E.-H. Yoo, e W. Xu, «AirSense», *Proc. 2015 Work. Pervasive Wirel. Healthc. - MobileHealth '15*, n. JUNE, pp. 17–22, 2015.

[33] D. Holstius, «GitHub Arduino code for PANDAs (Portable and Affordable Nephelometric Data Acquisition)». [Em linha]. Disponível em: https://github.com/holstius/PANDAs.

[34] D. M. Holstius, A. Pillarisetti, K. R. Smith, e E. Seto, «Field calibrations of a low-cost aerosol sensor at a regulatory monitoring site in California», *Atmos. Meas.*

Tech., vol. 7, n. 4, pp. 1121–1131, 2014.

[35] C. Mcnally, «Arduino Based Wireless Power Meter», n. May, 2010.

[36] G. Josu, «Projecto e Construção de um Sistema de Monitorização de Energia Eléctrica para uma Habitação», 2010.

[37] «Energino: an Arduino-based Energy Consumption Monitoring Shield: 5 Steps (with Pictures)». [Em linha]. Disponível em: http://www.instructables.com/id/Energino-an-Arduino-based-energy-consumption-moni/. [Acedido: 05-Fev-2017].

[38] R. F. Q. Henriques, «Desenvolvimento de um Sistema de Monitorização Remota do Consumo de Energia Elétrica e da Presença Humana em Edifícios», 2012.

[39] Texas Instruments, «LMx35, LMx35A Precision Temperature Sensors», n. 1, p. 28, 2015.

[40] Honeywell, «HIH-4000 Series Humidity Sensors». [Em linha]. Disponível em: http://www.farnell.com/datasheets/1685535.pdf.

[41] T. Liu, «Digital Humidity and Temperature sensor», *Adfruit*, pp. 1–5, 2016.

[42] AOSONG, «DHT11 Product Manual». [Em linha]. Disponível em: https://akizukidenshi.com/download/ds/aosong/DHT11.pdf. [Acedido: 02-Fev-2017].

[43] Sensirion, «SHT7x Humidity and Temperature Sensor». [Em linha]. Disponível em: http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT7x_Datasheet_V5.pdf.

[44] Sensirion, «Datasheet SHT1x». [Em linha]. Disponível em: https://www.sparkfun.com/datasheets/Sensors/SHT1x_datasheet.pdf. [Acedido: 02-Fev-2017].

[45] Measurement Specialties Inc., «HTU21D(F) - Relative Humidity sensor with Temperature output - Datasheet», n. October, pp. 1–21, 2013.

[46] J. Arling, K. O. Connor, e M. Mercieca, «Air Quality Sensor Network for Philadelphia», 2010.

[47] L. Jiankai, «Grove - Dust Sensor User Manual», p. 16, 2015.

[48] Sharp, «GP2Y1010AU0F Compact Optical Dust Sensor». [Em linha]. Disponível em: https://www.sparkfun.com/datasheets/Sensors/gp2y1010au_e.pdf. [Acedido: 09-Fev-2017].

[49] Samyoung S&C, «DUST SENSOR MODULE». [Em linha]. Disponível em: http://www.elecfreaks.com/store/download/product/Sensor/DSM501A/DSM501[1].pdf.

[50] Olimex, «MG811: CO2 Sensor», *Protocols and Product Manuals*. [Em linha]. Disponível em: www.olimex.cl/pdf/CO2b.pdf. [Acedido: 25-Fev-2017].

[51] CO2Meter.com, «Datasheet: C20 Sensor». [Em linha]. Disponível em: http://www.co2meters.com/Documentation/Datasheets/DS20-01 - C20.pdf.

[Acedido: 25-Fev-2017].

[52] CO2Meter.com, «Datasheet: K-30 Sensor». [Em linha]. Disponível em: http://co2meters.com/Documentation/Datasheets/DS30-01 - K30.pdf. [Acedido: 25-Fev-2017].

[53] COZIR, «Ultra Low Power Carbon Dioxide Sensor». [Em linha]. Disponível em: http://www.co2meters.com/Documentation/Datasheets/COZIR-Wide-Range-Datasheet.pdf. [Acedido: 25-Fev-2017].

[54] Adafruit, «Adafruit MiCS5524 CO / Alcohol / VOC Gas Sensor Breakout Guide Contents Guide Contents Overview Usage Best Practices Downloads Files Schematic Fabrication Print», 2016. .

[55] Figaro, «TGS 2602». [Em linha]. Disponível em: http://www.figarosensor.com/products/2602pdf.pdf. [Acedido: 25-Fev-2017].

[56] CO2Meter.com, «Datasheet: iAQ- 2000 Sensor». [Em linha]. Disponível em: http://www.co2meters.com/Documentation/Datasheets/DS-iAQ2000.pdf. [Acedido: 25-Fev-2017].

[57] CRMagnetics, «Split Core Current Transformer». [Em linha]. Disponível em: http://www.crmagnetics.com/Assets/ProductPDFs/CR3100.pdf. [Acedido: 28-Fev-2017].

[58] «Arduino Timer Interrupts: 6 Steps (with Pictures)». [Em linha]. Disponível em: http://www.instructables.com/id/Arduino-Timer-Interrupts/. [Acedido: 05-Mar-2017].

[59] Texas Instruments, «Compact DC Voltage and Current Sense PCB with Analog Output Photo Tips for Soldering». pp. 1–3, 2011.

[60] R. Faludi, *A Practical Guide to the ZigBee Mesh Networking Protocol. Building Wireless Sensor Networks*, n. 1. 2010.

[61] «Arduino Fio». [Em linha]. Disponível em: https://store.arduino.cc/arduino-fio. [Acedido: 10-Mar-2017].

[62] «*XBee* 2mW PCB Antenna - Series 2 (ZigBee Mesh) - WRL-11217 - SparkFun Electronics». [Em linha]. Disponível em: https://www.sparkfun.com/products/retired/11217. [Acedido: 10-Mar-2017].

[63] «SparkFun *XBee* Explorer USB - WRL-11812 - SparkFun Electronics». [Em linha]. Disponível em: https://www.sparkfun.com/products/11812. [Acedido: 10-Mar-2017].

[64] Sistemas de Sensores e Captação de Energia, «Armazenamento e visualização dos dados da rede ZigBee». Engenharia Electrotécnica-Telecomunicações da Universidade da Madeira.

[65] *Raspberry Pi*, «*Raspberry Pi* 2 Model B». [Em linha]. Disponível em: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. [Acedido: 14-Mar-2017].

[66] *Raspberry Pi*, «SD cards - *Raspberry Pi* Documentation». [Em linha]. Disponível em: https://www.raspberrypi.org/documentation/installation/sd-cards.md. [Acedido: 14-Mar-2017].

[67] L. Moreira, «Antenas para Redes de Sensores sem Fios», Universidade da Madeira, 2016.

[68] OMRON Corporation, «High Capacity and High Dielectric Strength Miniature Relay with Fully Sealed». [Em linha]. Disponível em: https://www.omron.com/ecb/products/pdf/en-g6b.pdf. [Acedido: 03-Abr-2017].

[69] «DustDuino.org». [Em linha]. Disponível em: http://dustduino.org/. [Acedido: 04-Mai-2017].

[70] «bildr » Sensing Humidity With The SHT15 + Arduino». [Em linha]. Disponível em: http://bildr.org/2012/11/sht15-arduino/. [Acedido: 10-Set-2017].

[71] Arduino, «Arduino - Wire». [Em linha]. Disponível em: https://www.arduino.cc/en/reference/wire. [Acedido: 18-Mar-2017].

[72] «Active Low Pass Filter - Op-amp Low Pass Filter». [Em linha]. Disponível em: http://www.electronics-tutorials.ws/filter/filter_5.html. [Acedido: 03-Mar-2017].

[73] Intersil, «ISL28148, ISL28248 Datasheet». [Em linha]. Disponível em: http://www.intersil.com/content/dam/Intersil/documents/isl2/isl28148-248.pdf. [Acedido: 03-Mar-2017].

[74] Texas Instruments, «LM741 Operational Amplifier 1 Features 3 Description». [Em linha]. Disponível em: http://www.ti.com/lit/ds/symlink/lm741.pdf. [Acedido: 03-Mar-2017].

[75] Fluke Corp., «Fluke i410». [Em linha]. Disponível em: http://img.elektronika.sk/files/Images/fluke_i410_instructionsheet_en.pdf. [Acedido: 05-Mai-2017].

[76] Amprobe, «34XR-A True-RMS Digital Multimeter |». [Em linha]. Disponível em: http://www.amprobe.com/Amprobe/usen/digital-multimeters/Full-Size-Multimeters/AMP-34XR-A.htm?PID=73032. [Acedido: 05-Mai-2017].

[77] Sistemas de Sensores e Captação de Energia, «Rede de sensores com módulos *XBee*». Engenharia Electrotécnica-Telecomunicações da Universidade da Madeira.

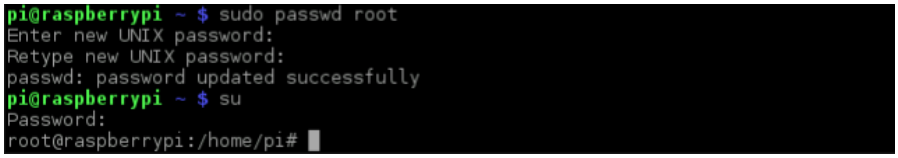
[78] «What is Schmitt Trigger | How It Works - HowToMechatronics». [Em linha]. Disponível em: http://howtomechatronics.com/how-it-works/electrical-engineering/schmitt-trigger/. [Acedido: 22-Mar-2017].

[79] «EC2 SERIES OUTLINE DRAWING AND DIMENSIONS PAD LAYOUT ». [Em linha]. Disponível em: http://www.mouser.com/ds/2/283/ec2-6037.pdf. [Acedido: 03-Mar-2017].

Anexo 1 – Instalação no *Raspberry Pi*

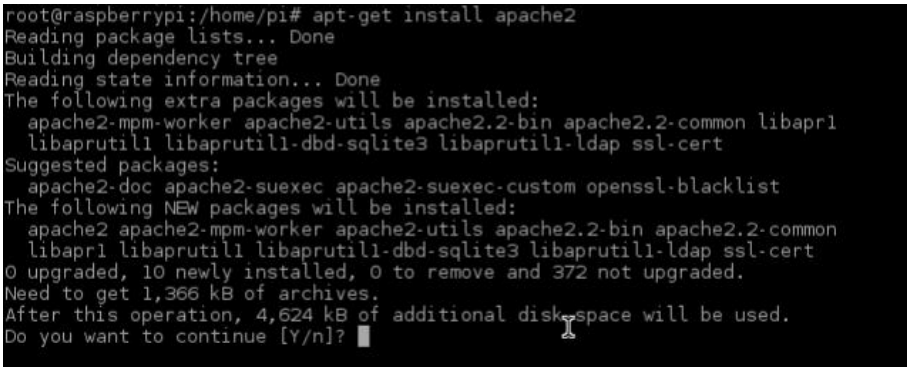
Nesta secção pretende-se apresentar um guia de instalação dos programas necessários para o funcionamento do projecto WSN no *Raspberry Pi*.

Começar por abrir a linha de comandos. Serão necessárias permissões de administrador pelo que será necessário trocar o utilizador para SuperUser, digitando `su` – se ainda não foi definida qualquer password, definir através de `sudo passwd root` , e inserir uma password e a respetiva confirmação.

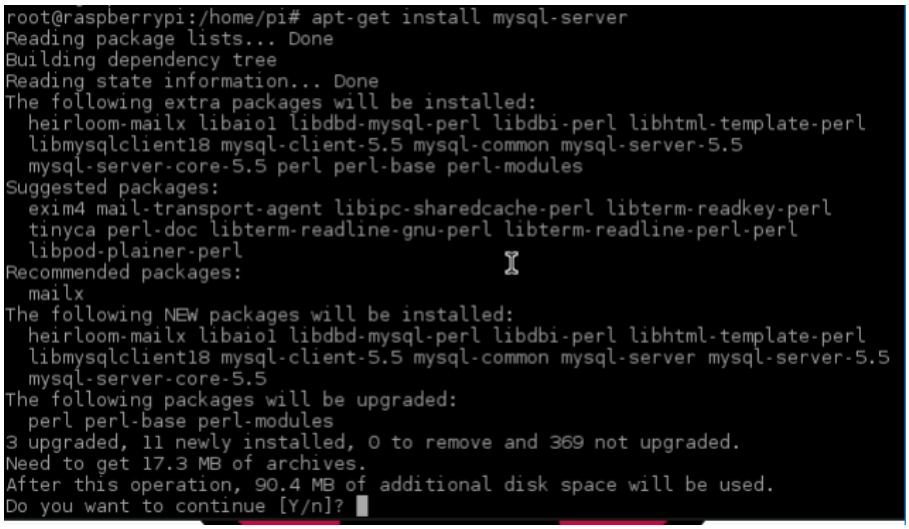


Depois, para trabalhar sempre em modo SuperUser, digitar `su` , premir Enter e inserir a password definida). Outra possibilidade para proceder à instalação será a utilização do comando `sudo`, que atribui privilégios de administrador a cada linha de comando, ou seja, antes de cada um dos seguintes comandos deverá ser digitado o comando `sudo`. Exemplo: `sudo apt-get install apache2.`

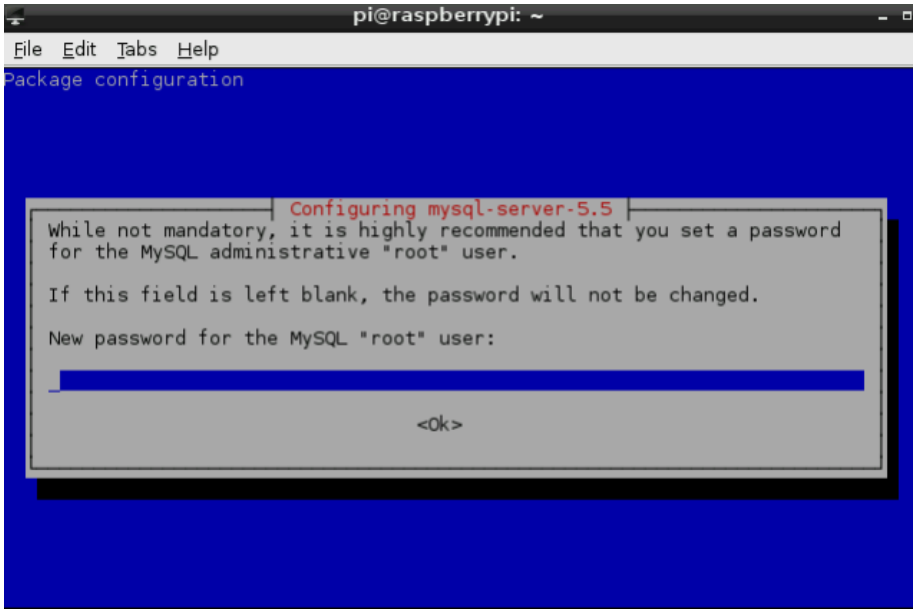
- Atualização do *Raspberry Pi*: `apt-get update`
- Instalação do servidor web: `apt-get install apache2` (Para continuar a instalação será necessário inserir "Y" e premir Enter.)



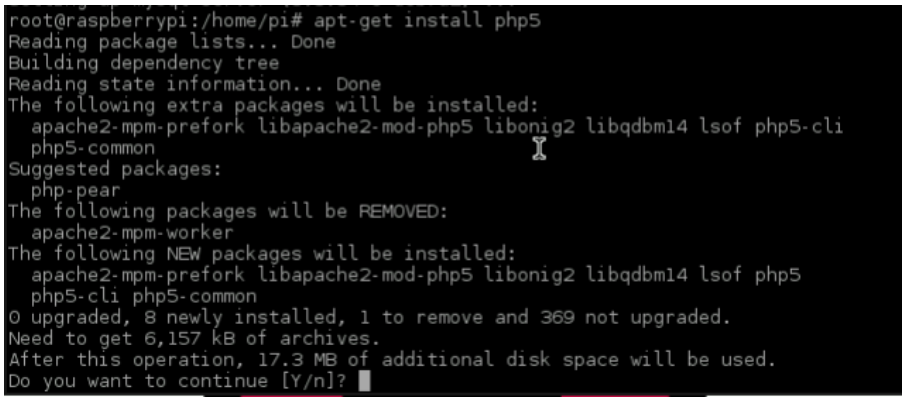
- Instalação da base de dados: `apt-get install mysql-server` (Para continuar a instalação será necessário inserir "Y" e premir Enter.)



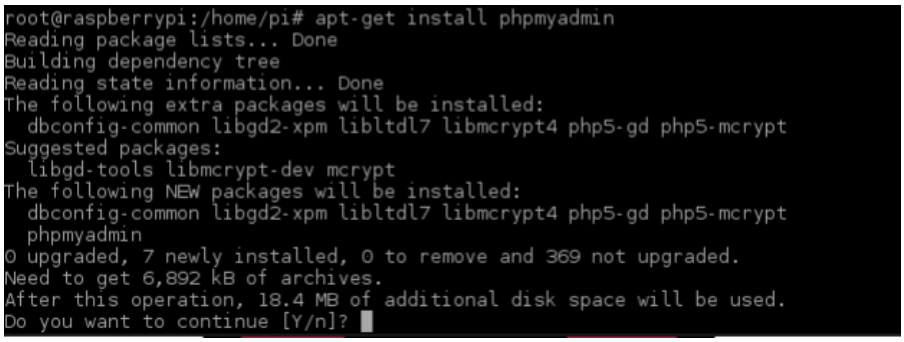
Aparecerá uma mensagem para definir uma password para o utilizador "root" do MySQL. É recomendado inserir uma password (exemplo: raspberrymysql).



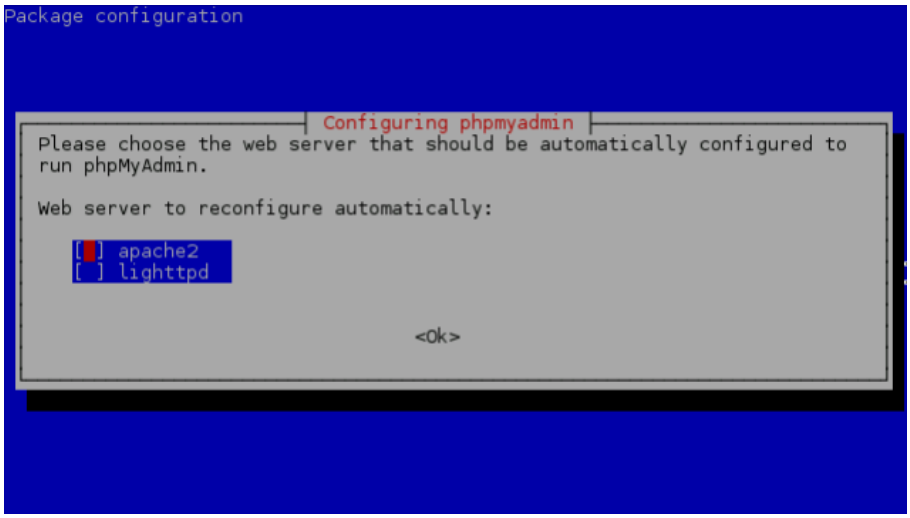
- Instalação da linguagem de programação: `apt-get install php5` (Para continuar a instalação será necessário inserir "Y" e premir Enter.)



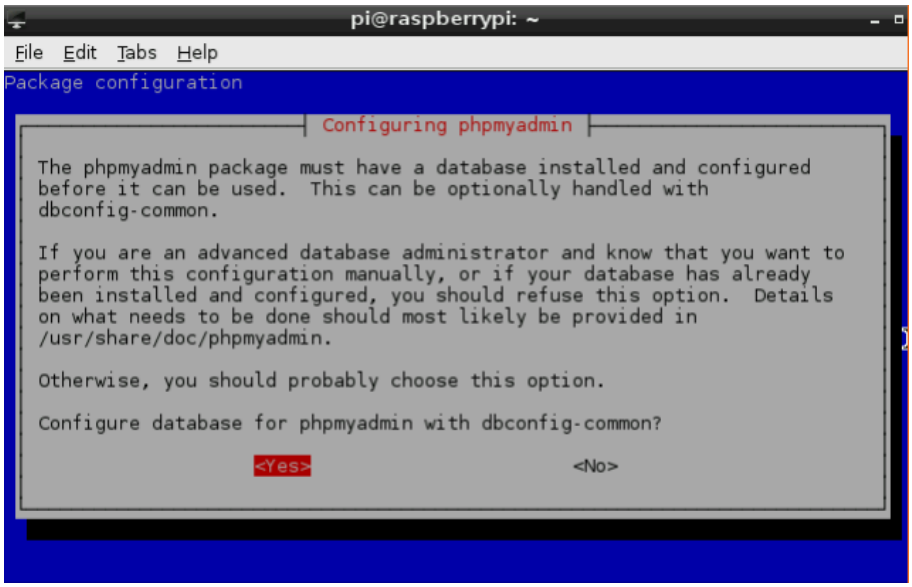
- Instalação do módulo MySQL para PHP5: `apt-get install php5-mysql`
- Instalação da ferramenta de administração da base de dados: `apt-get install phpmyadmin` (Para continuar a instalação será necessário inserir "Y" e premir Enter)



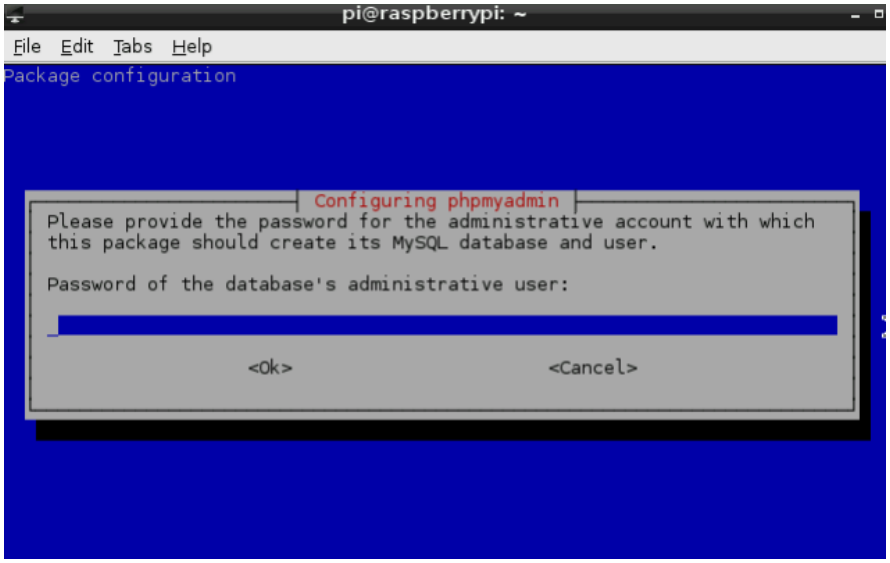
Para este caso, dever-se-á selecionar a opção "apache2".



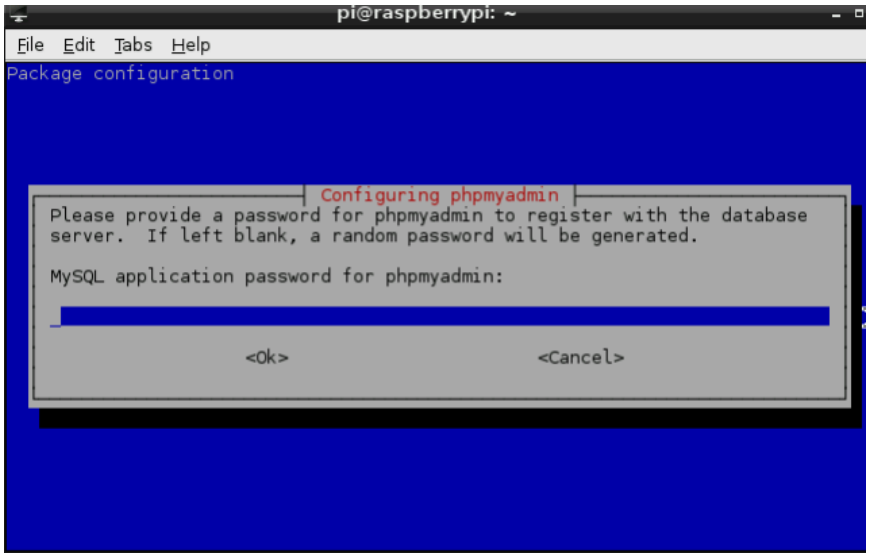
Neste passo, dever-se-á seleccionar a opção "Yes".



Aqui será necessário inserir a password criada aquando da instalação do MySQL.



E, por fim, é necessário definir a password para o phpmyadmin e confirmar a respectiva de seguida (ex: raspberryphp) .

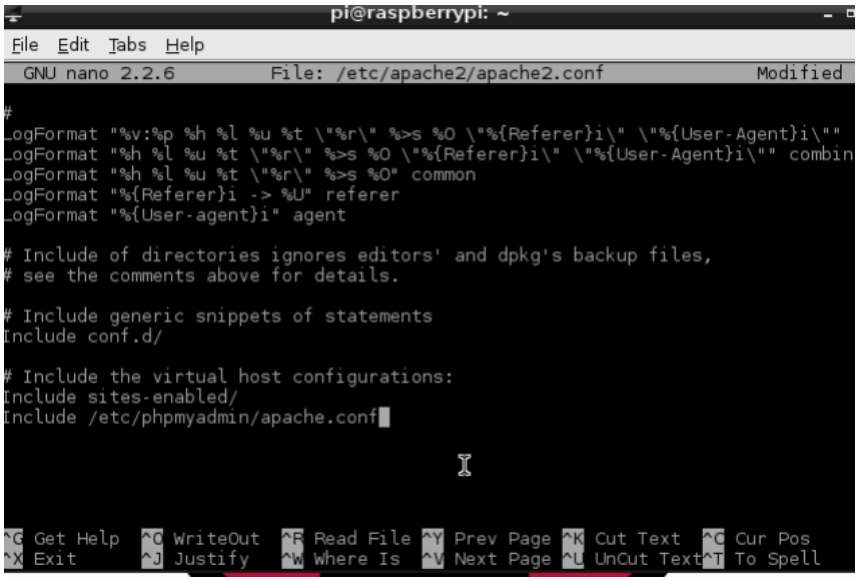


De seguida é necessário entrar no ficheiro `apache2.conf` através de

```
nano /etc/apache2/apache2.conf
```

para acrescentar a seguinte linha no final do ficheiro,

```
Include /etc/phpmyadmin/apache.conf
```



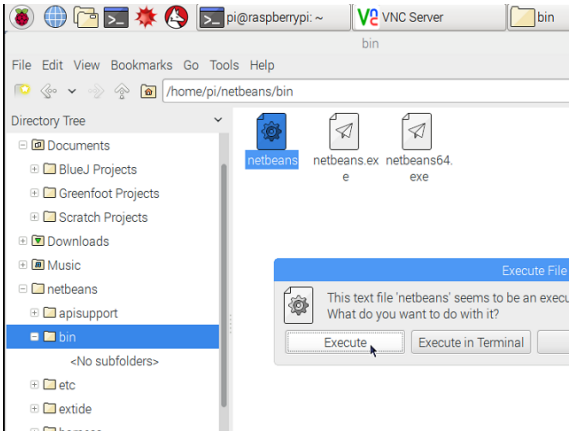
, depois, digitar CTRL+X para Sair, digitar Y para guardar, carregar novamente Enter (para guardar com o mesmo nome) e reiniciar o Apache

```
/etc/init.d/apache2 restart
```

```
root@raspberrypi:/home/pi# /etc/init.d/apache2 restart
[ ok ] Restarting apache2 (via systemctl): apache2.service.
root@raspberrypi:/home/pi#
```

Depois é necessária a instalação do NetBeans 8.2 Java SE (foi instalado através do browser, recorrendo a https://netbeans.org/downloads/ e selecionando a plataforma OS Independent.)

(Nota: o ficheiro é .zip, logo será necessário extrair o seu conteúdo. Para isso basta clicar no ficheiro com o botão direito do rato e selecionar *Extract Here.* Após a extração, o NetBeans está pronto a ser executado. Para executar, dirigir-se à pasta extraída, entrar na pasta *bin* e clicar no ficheiro *netbeans*.)



Para criar o atalho no Ambiente de trabalho, clicar com o botão direito do rato, no Ambiente de Trabalho, e selecionar Create New... > New Blank File. Atribuir o nome "NetBeans" e, dentro do ficheiro deverá conter o seguinte (atenção à localização da pasta do netbeans. Neste caso a pasta foi colocada em `/home/pi/`):

```
[Desktop Entry]
Name=NetBeans
Comment=NetBeans is a software development platform written in Java.
Icon=/home/pi/netbeans/nb/netbeans.png
Exec=/home/pi/netbeans/bin/netbeans
Type=Application
Encoding=UTF-8
Terminal=false
Categories=None;
```

É necessária ainda a instalação da livraria librxtx: `apt-get install librxtx-java`

Depois deve ser copiado o ficheiro `RXTXcomm.jar` de `/usr/share/java` para a pasta `jars` do projeto.

```
cd /usr/share/java
cp RXTXcomm.jar /home/pi/Desktop/WSN/uma_XBee_parser/jars
```

Copiar o ficheiro `librxtxSerial.so` que está na pasta `/usr/lib/jni/` , para a pasta de instalação do Java `/usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt/jre/lib/arm .`

```
cd /usr/lib/jni/
```

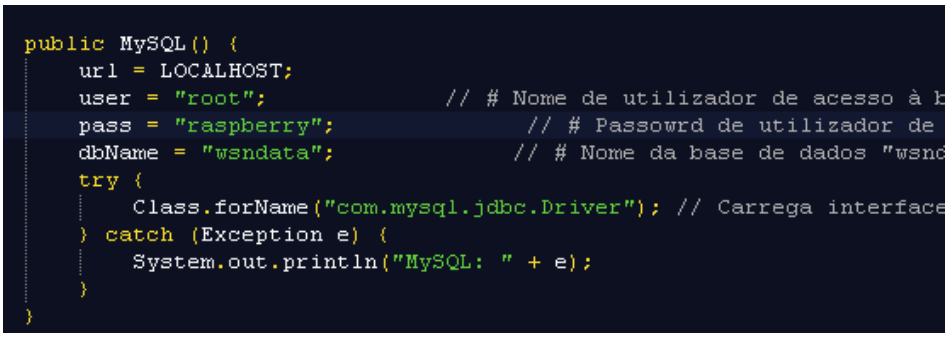
```
cp librxtxSerial.so /usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt/jre/lib/arm
```

Por fim, a pasta `wsn_web` deverá ser movida para `/var/www/html`

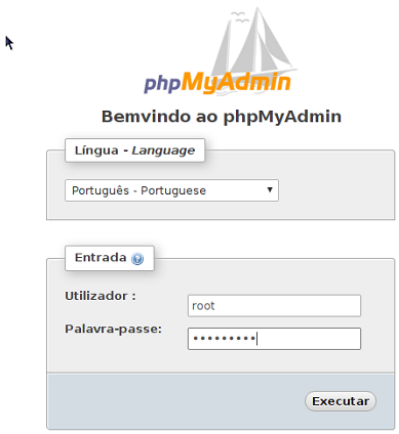
```
mv wsn_web /var/www/html
```

Na pasta `wsn_web`, é necessário editar o ficheiro `dbinfo.php` de acordo com as definições configuradas anteriormente, para que seja possível aceder à base de dados. Neste exemplo, a password definida foi `raspberry` para o utilizador `root`:

Adicionar essa informação também no ficheiro `MySQL.java` do projeto (ficheiro dentro da pasta `/WSN/uma_XBee_parser/src/uma_XBee_parser`)



No explorador de internet aceder à ferramenta phpmyadmin (utilizar os dados de login definidos na instalação do MySQL, que neste exemplo foi: Utilizador “root”; Password “raspberrymysql”), que permite visualizar a base de dados, digitando o seguinte endereço: http://localhost/phpmyadmin e escolher a linguagem (Português).



No phpmyadmin criar uma nova base de dados com o nome wsndata. Após a sua criação, clicar nesse nome no menu do lado esquerdo da janela e depois no menu superior clicar em Importar e carregar o ficheiro wsnData.sql que está na pasta wsn_db. Em Estrutura para visualizar os campos da base de dados.

Anexo 2 – Imagens e código do protótipo criado para medição da temperatura e humidade com carregamento automático

Nesta secção apresenta-se um conjunto de fotografias do protótipo criado para a medição da temperatura e da humidade com carregamento automático e o respetivo código.

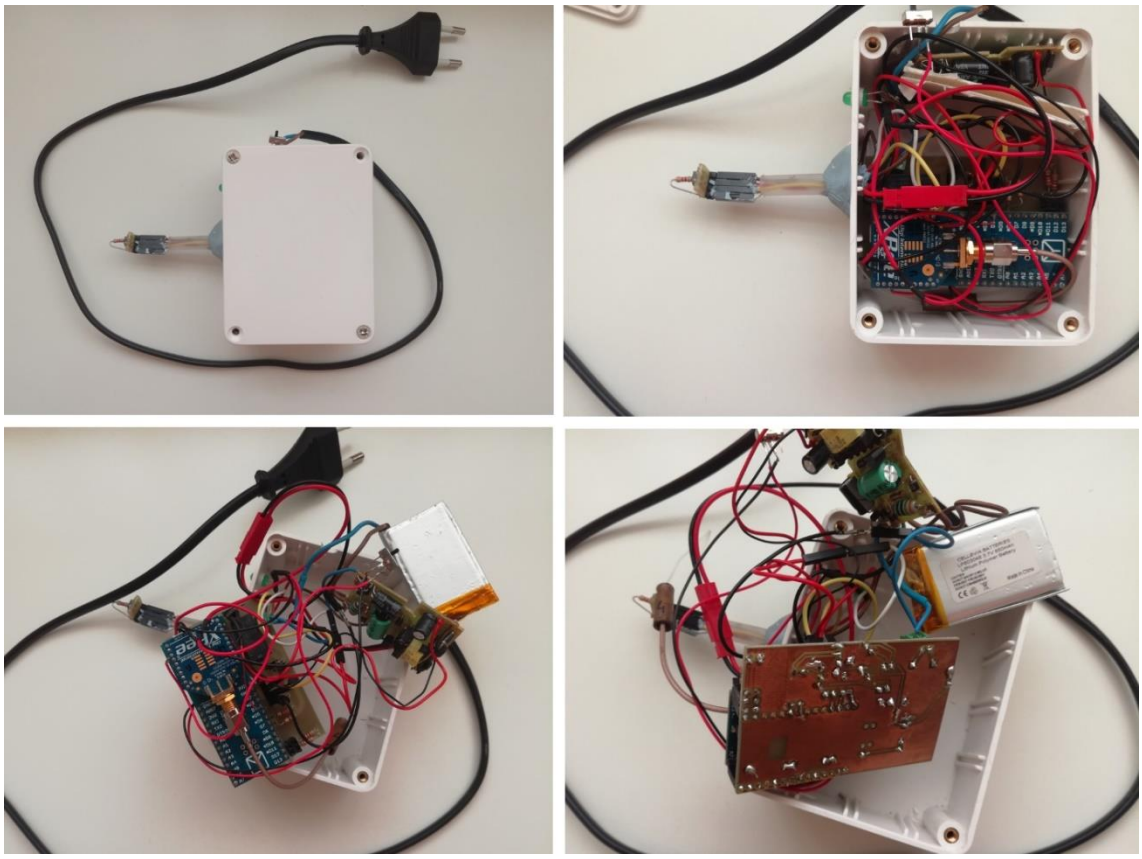


Figura 7.1 - Fotografias do protótipo criado para a medição da temperatura e da humidade com carregamento automático.

```
/*Universidade da Madeira 2016/2017
*Projeto de Mestrado: Monitorização habitacional com uma rede de sensores sem fios
*João Baptista
*Código para o módulo de temperatura e humidade com módulo de carregamento automático*/
#include <avr/sleep.h>
int i=0;
int j=0;
int db=0;
byte trama[] =
{0x7E,0x00,0x30,0x10,0x00,0x00,0x13,0xA2,0x00,0x40,0x2D,0x25,0x36,0xFF
,0xFE,0x00,0x00,'?','T','7',0x3D,0x00,0x00,0x00,0x00,0x00,0x26,'H','7'
,0x3D,0x00,0x00,0x00,0x00,0x00,0x26,'B','7',0x3D,0x00,0x00,0x00,0x00,0
x26,'R','7',0x3D,0x00,0x00,0x00,0x00,0x00};
//TEMP COMECA TRAMA[21] TERMINA TRAMA[25]
//HUMI COMECA TRAMA[30] TERMINA TRAMA[34]
//BAT COMECA TRAMA[39] TERMINA TRAMA[42]
//RSSI COMECA TRAMA[47] TERMINA TRAMA[50]
int XBeeSleep = 4;
int seconds = 10; // Período em segundos
```

```
int start = seconds*30;
int counter = 0;
unsigned int tcnt2;
int bat = 0;
int valorST = 0;
int valorSH = 0;
int valorSRSSI = 0;
byte checksum = 0;
int SHT_clockPin = 3;  // pin used for clock
int SHT_dataPin  = 2;  // pin used for data
int EstadoRele=0;
//////////////////////setup pin
void setup() {
pinMode(7, OUTPUT);//set relay
pinMode(8, OUTPUT);//reset relay
Serial.begin(9600);
digitalWrite(7, HIGH);   //set a 0
delay(20);
digitalWrite(7, LOW);    //reset a 1
EstadoRele=0;
pinMode(XBeeSleep, OUTPUT);
TIMSK2 &= ~(1<<TOIE2);
TCCR2A &= ~((1<<WGM21) | (1<<WGM20));
TCCR2B &= ~(1<<WGM22);
ASSR &= ~(1<<AS2);
TIMSK2 &= ~(1<<OCIE2A);
TCCR2B |= (1<<CS22) | (1<<CS21) | (1<<CS20);
tcnt2 = 0;
TCNT2 = tcnt2;
TIMSK2 |= (1<<TOIE2);
set_sleep_mode(SLEEP_MODE_PWR_SAVE);
sleep_enable();
}
ISR(TIMER2_OVF_vect) {
/* Reload the timer */
TCNT2 = tcnt2;
counter++;
}
void readAndSend(){
  digitalWrite(12, HIGH);
float temperature = getTemperature();
float humidity = getHumidity();
int auxt1 = temperature; //parte int temp
int auxh1 = humidity; //parte int humi
float auxt2 = (temperature-auxt1); //parte dec temp
float auxh2 = (humidity-auxh1); //part dec humi
int auxt3 = 0.00;
int auxh3 = 0.00;
auxt3 = auxt2*100; //parte dec temp convertida em inteiro
auxh3 = auxh2*100; //parte dec humi convertida em inteiro
char acBuf[5 + 1];
sprintf( acBuf, "%2d.%2d", auxt1, auxt3 );
trama[21] = acBuf[0];
trama[22] = acBuf[1];
trama[23] = acBuf[2];
if (auxt3<10){
  trama[24] = '0';
  trama[25] = acBuf[4];
  } else{
  trama[24] = acBuf[3];
  trama[25] = acBuf[4];
```

```
  }

  char acBuf1[5 + 1];
sprintf( acBuf1, "%2d.%2d", auxh1, auxh3 );

trama[30] = acBuf1[0];
trama[31] = acBuf1[1];
trama[32] = acBuf1[2];
if (auxh3<10){
  trama[33] = '0';
  trama[34] = acBuf1[4];
  } else{
  trama[33] = acBuf1[3];
  trama[34] = acBuf1[4];
  }

bat = analogRead(0);
float batv = 0.00;
batv=(bat*4.2)/1023;
int auxbatv = batv; //parte int bat
float auxtbatv1 = (batv-auxbatv); //parte dec temp
int auxtbatv2 = 0.00;
auxtbatv2 = auxtbatv1*100;
char acBuf2[4 + 1];
//char acBuf2[5 + 1];
sprintf( acBuf2, "%4d", bat);

trama[39] = acBuf2[0];
trama[40] = acBuf2[1];
trama[41] = acBuf2[2];
trama[42] = acBuf2[3];

valorSRSSI = db; // rssi
char acBuf4[4 + 1];
sprintf(acBuf4, "%4d", valorSRSSI);//converte valor pa string
trama[47]=acBuf4[0];
trama[48]=acBuf4[1];
trama[49]=acBuf4[2];
trama[50]=acBuf4[3];
// Wake up XBee
digitalWrite(XBeeSleep, LOW); // Liga XBee
delay(15); // Espera que acorde
j=j+1;
if (j==16){
    // Wake up XBee
digitalWrite(XBeeSleep, LOW); // Liga XBee
delay(15); // Espera que acorde
  rssiTest();
  delay(15);
// Sleep XBee
digitalWrite(XBeeSleep, HIGH);
  }
// Cálculo do checksum
checksum = 0;
for ( i = 3; i < sizeof(trama)-1; i++) {
checksum+= trama[i]; // Soma os valores que contam para checksum
}
trama[sizeof(trama)-1] = 0xFF - checksum; // Realiza o complemento para
2
Serial.write(trama, sizeof(trama)); // Envia a trama pela porta série
delay(100);
```

```
// Sleep XBee
digitalWrite(XBeeSleep, HIGH);
digitalWrite(12, LOW);
}
 void rssiTest(){
  byte                                trssi[]                                =
{0x7E,0x00,0x0F,0x17,0x01,0x00,0x13,0xA2,0x00,0x40,0x2D,0x25,0x36,0xFF
,0xFE,0x01,0x44,0x42,0xE6};
   Serial.write(trssi, sizeof(trssi));
 while (Serial.available() > 0) {
    int inChar = Serial.read();
    if (inChar == 'B') {
   int t = Serial.read();
      db = Serial.read();
    }
  }
      j=0;
  }
void ControlaRelay(){
/*Valor do sensor 1 a 740 significa que ira comecar a rotina de
carregamento
 * quando o valor da tensao atingir os 3.62V
*/
if (bat<740 && EstadoRele==0){
  //liga o relay para carregar bateria
  digitalWrite(8, HIGH);   //reset a 0
  delay(20);
  digitalWrite(8, LOW);    //set a 1
  EstadoRele=1;
  }

if (bat>740 && bat<850){
  EstadoRele=0;
  }
if (bat>850){
   EstadoRele=1;
  }
 /*Valor do sensor 1 a 853 significa que ira comecar a rotina de
descarregamento
 * quando o valor da tensao atingir os 4.17V*/
 if (bat>850 && EstadoRele==1){
  //desliga o relay para deixar de carregar bateria
  digitalWrite(7, HIGH);   //set a 0
  delay(20);
  digitalWrite(7, LOW);    //reset a 1
  EstadoRele=0;
  }
  if (EstadoRele==1){
    digitalWrite(13, HIGH);   //set a 1
    }
    else {
     digitalWrite(13, LOW);
      EstadoRele=0;
    }
}
/* Main loop. */
void loop() {
  ControlaRelay();
if(counter == start){

TIMSK2 &= ~(1<<TOIE2);
```

```
// executa tarefa
readAndSend();
counter = 0;
TCNT2 = tcnt2;
TIMSK2 |= (1<<TOIE2);
}
// Sleep
sleep_enable();
sleep_mode();
sleep_disable();
}
//////////////SHT15 From: http://bildr.org/2012/11/sht15-arduino/
float getTemperature(){
  //Return Temperature in Celsius
  SHT_sendCommand(B00000011, SHT_dataPin, SHT_clockPin);
  SHT_waitForResult(SHT_dataPin);

  int val = SHT_getData(SHT_dataPin, SHT_clockPin);
  SHT_skipCrc(SHT_dataPin, SHT_clockPin);
  return (float)val * 0.01 - 40; //convert to celsius
}

float getHumidity(){
  //Return  Relative Humidity
  SHT_sendCommand(B00000101, SHT_dataPin, SHT_clockPin);
  SHT_waitForResult(SHT_dataPin);
  int val = SHT_getData(SHT_dataPin, SHT_clockPin);
  SHT_skipCrc(SHT_dataPin, SHT_clockPin);
  return -4.0 + 0.0405 * val + -0.0000028 * val * val;
}

void SHT_sendCommand(int command, int dataPin, int clockPin){
  // send a command to the SHTx sensor
  // transmission start
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  digitalWrite(dataPin, HIGH);
  digitalWrite(clockPin, HIGH);
  digitalWrite(dataPin, LOW);
  digitalWrite(clockPin, LOW);
  digitalWrite(clockPin, HIGH);
  digitalWrite(dataPin, HIGH);
  digitalWrite(clockPin, LOW);

  // shift out the command (the 3 MSB are address and must be 000, the
last 5 bits are the command)
  shiftOut(dataPin, clockPin, MSBFIRST, command);

  // verify we get the right ACK
  digitalWrite(clockPin, HIGH);
  pinMode(dataPin, INPUT);

  if (digitalRead(dataPin)) Serial.println("ACK error 0");
  digitalWrite(clockPin, LOW);
  if (!digitalRead(dataPin)) Serial.println("ACK error 1");
}


void SHT_waitForResult(int dataPin){
  // wait for the SHTx answer
  pinMode(dataPin, INPUT);
```

```
  int ack; //acknowledgement

  //need to wait up to 2 seconds for the value
  for (int i = 0; i < 1000; ++i){
    delay(2);
    ack = digitalRead(dataPin);
    if (ack == LOW) break;
  }

  if (ack == HIGH) Serial.println("ACK error 2");
}
int SHT_getData(int dataPin, int clockPin){
  // get data from the SHTx sensor

  // get the MSB (most significant bits)
  pinMode(dataPin, INPUT);
  pinMode(clockPin, OUTPUT);
  byte MSB = shiftIn(dataPin, clockPin, MSBFIRST);

  // send the required ACK
  pinMode(dataPin, OUTPUT);
  digitalWrite(dataPin, HIGH);
  digitalWrite(dataPin, LOW);
  digitalWrite(clockPin, HIGH);
  digitalWrite(clockPin, LOW);

  // get the LSB (less significant bits)
  pinMode(dataPin, INPUT);
  byte LSB = shiftIn(dataPin, clockPin, MSBFIRST);
  return ((MSB << 8) | LSB); //combine bits
}
void SHT_skipCrc(int dataPin, int clockPin){
  // skip CRC data from the SHTx sensor
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  digitalWrite(dataPin, HIGH);
  digitalWrite(clockPin, HIGH);
  digitalWrite(clockPin, LOW);
}
```

Anexo 3 – Imagens e código do protótipo criado para a medição da temperatura e da humidade sem carregamento automático

Nesta secção apresenta-se um conjunto de fotografias do protótipo criado para a medição da temperatura e da humidade sem carregamento automático e o respetivo código.

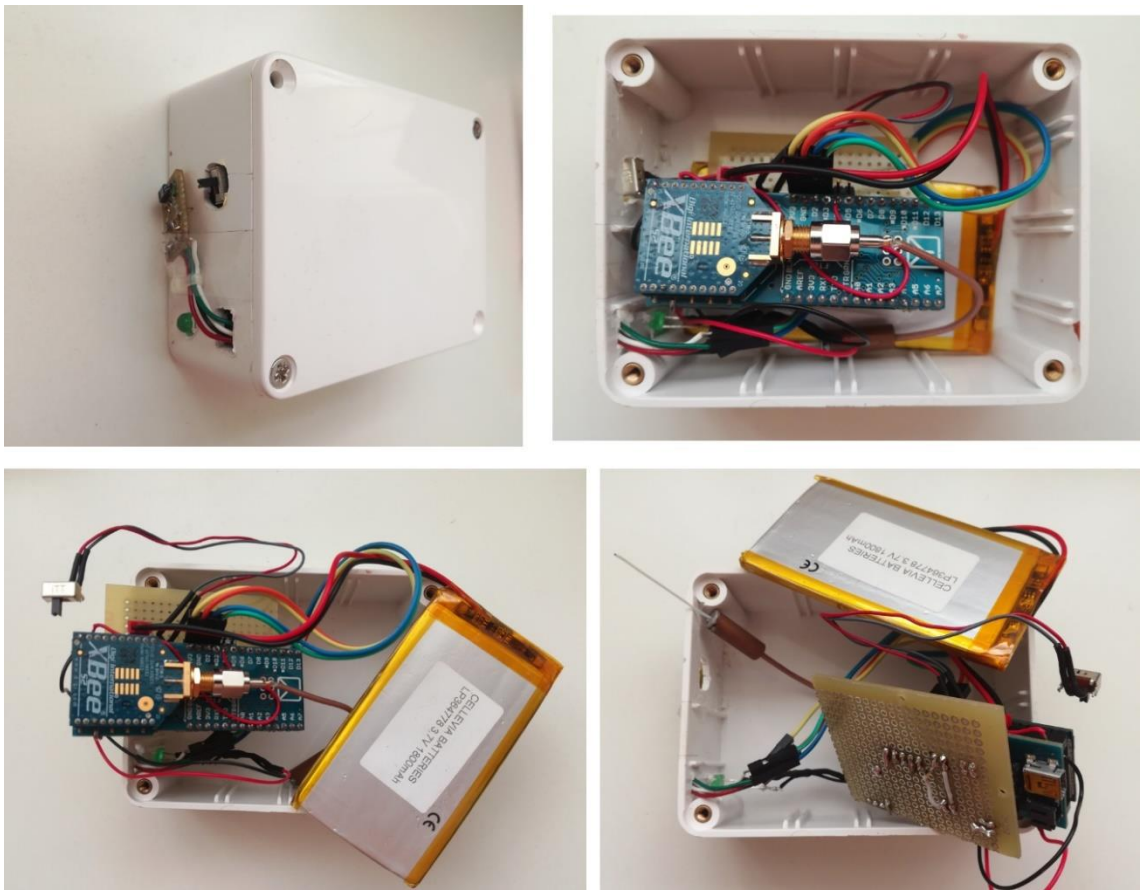


Figura 7.2 - Fotografias do protótipo criado para a medição da temperatura e humidade sem carregamento automático.

```
/*Universidade da Madeira 2016/2017
*Projeto de Mestrado: Monitorização habitacional com uma rede de sensores sem fios
*João Baptista
*Código para o módulo de temperatura e humidade sem módulo de carregamento
*/
#include <avr/sleep.h>
int i=0;
int j=0;
int db=0;
byte trama[] =
{0x7E,0x00,0x31,0x10,0x00,0x00,0x13,0xA2,0x00,0x40,0x2D,0x25,0x36,0xFF
,0xFE,0x00,0x00,'?','Y','T',0x3D,0x00,0x00,0x00,0x00,0x00,0x26,'Y','H'
,0x3D,0x00,0x00,0x00,0x00,0x00,0x26,'Y','R',0x3D,0x00,0x00,0x00,0x00,0
x26,'Y','B',0x3D,0x00,0x00,0x00,0x00,0x00,0x00};
//TEMP COMECA TRAMA[21] TERMINA TRAMA[25]
//HUMI COMECA TRAMA[30] TERMINA TRAMA[34]
//BAT COMECA TRAMA[39] TERMINA TRAMA[42]
//RSSI COMECA TRAMA[47] TERMINA TRAMA[50]
```

```
int XBeeSleep = 4;
int seconds = 10; // Período em segundos
int start = seconds*30;
int counter = 0;
unsigned int tcnt2;
int bat = 0;
int valorST = 0;
int valorSH = 0;
int valorSRSSI = 0;
byte checksum = 0;
int SHT_clockPin = 3;  // pin used for clock
int SHT_dataPin  = 2;  // pin used for data
int EstadoRele=0;
int vati=0;
int vatd=0;
int vahi=0;
int vahd=0;
int vabi=0;
int vabd=0;
int dbant=0;
void setup() {
Serial.begin(9600);
pinMode(XBeeSleep, OUTPUT);
TIMSK2 &= ~(1<<TOIE2);
TCCR2A &= ~((1<<WGM21) | (1<<WGM20));
TCCR2B &= ~(1<<WGM22);
ASSR &= ~(1<<AS2);
TIMSK2 &= ~(1<<OCIE2A);
TCCR2B |= (1<<CS22) | (1<<CS21) | (1<<CS20);
tcnt2 = 0;
TCNT2 = tcnt2;
TIMSK2 |= (1<<TOIE2);
set_sleep_mode(SLEEP_MODE_PWR_SAVE);
sleep_enable();
}
ISR(TIMER2_OVF_vect) {
/* Reload the timer */
TCNT2 = tcnt2;
counter++;
}
void readAndSend(){
float temperature = getTemperature();
float humidity = getHumidity();
int auxt1 = temperature; //parte int temp
int auxh1 = humidity; //parte int humi
float auxt2 = (temperature-auxt1); //parte dec temp
float auxh2 = (humidity-auxh1); //part dec humi
int auxt3 = 0.00;
int auxh3 = 0.00;
auxt3 = auxt2*100; //parte dec temp convertida em inteiro
auxh3 = auxh2*100; //parte dec humi convertida em inteiro
if (auxt1<0){
auxt1=vati;
auxt3=vatd;
}
else{
vati = auxt1;
vatd = auxt3;
}
char acBuf[5 + 1];
sprintf( acBuf, "%2d.%2d", auxt1, auxt3 );
```

```
trama[21] = acBuf[0];
trama[22] = acBuf[1];
trama[23] = acBuf[2];
if (auxt3<10){
  trama[24] = '0';
  trama[25] = acBuf[4];
  } else{
  trama[24] = acBuf[3];
  trama[25] = acBuf[4];
  }
if (auxh1<0){
  auxh1=vahi;
  auxh3=vahd;
}
else {
  vahi=auxh1;
  vahd=auxh3;
}
char acBuf1[5 + 1];
sprintf( acBuf1, "%2d.%2d", auxh1, auxh3 );
trama[30] = acBuf1[0];
trama[31] = acBuf1[1];
trama[32] = acBuf1[2];
if (auxh3<10){
  trama[33] = '0';
  trama[34] = acBuf1[4];
  } else{
  trama[33] = acBuf1[3];
  trama[34] = acBuf1[4];
  }
char acBuf2[4 + 1];
valorSRSSI = db*(-1); // rssi
sprintf( acBuf2, "%4d", valorSRSSI);
trama[39] = acBuf2[0];
trama[40] = acBuf2[1];
trama[41] = acBuf2[2];
trama[42] = acBuf2[3];
bat = analogRead(0);
float batv = 0.00;
float batva=0.00;
batva=(bat*5);
batv=batva/1023;
int auxbat1=batv;
float auxbat2 = (batv-auxbat1);
float auxbat3=0.00;
auxbat3=auxbat2*100;
int auxbat4 = auxbat3;
char acBuf9[5 + 1];
if (auxbat1<0){
auxbat1=vabi;
auxbat4=vabd;
}
else{
vabi = auxbat1;
vabd = auxbat4;
}
sprintf( acBuf9, "%2d.%2d", auxbat1, auxbat4);
trama[47]=acBuf9[0];
trama[48]=acBuf9[1];
trama[49]=acBuf9[2];
if (auxbat4<10){
```

```
  trama[50]='0';
  trama[51]=acBuf9[4];
  }
  else{
    trama[50]=acBuf9[3];
    trama[51]=acBuf9[4];
  }

// Wake up XBee
digitalWrite(XBeeSleep, LOW); // Liga XBee
delay(15); // Espera que acorde
j=j+1;
if (j==16){
  rssiTest();
  }
// Cálculo do checksum
checksum = 0;
for ( i = 3; i < sizeof(trama)-1; i++) {
checksum+= trama[i]; // Soma os valores que contam para checksum
}
trama[sizeof(trama)-1] = 0xFF - checksum; // Realiza o complemento para
2
Serial.write(trama, sizeof(trama)); // Envia a trama pela porta série
delay(100);
// Sleep XBee
digitalWrite(XBeeSleep, HIGH);
}

 void rssiTest(){
  int countr=0;
  int flagr=0;
  byte                              trssi[]                              =
{0x7E,0x00,0x0F,0x17,0x01,0x00,0x13,0xA2,0x00,0x40,0x2D,0x25,0x36,0xFF
,0xFE,0x01,0x44,0x42,0xE6};
   Serial.write(trssi, sizeof(trssi));
 while (Serial.available() > 0) {
  countr=countr+1;
  int inChar = Serial.read();
  if (countr>30){
    inChar='B';
    flagr=1;
    countr=0;
  }
    if (inChar == 'B') {
      if (flagr==1){
        db=dbant;
        flagr=0;
      }
      else {
      int t = Serial.read();
      db = Serial.read();
      dbant = db;
      countr=0;
      }
    }
  }
      j=0;
  }

/* Main loop. */
void loop() {
```

```
if(counter == start){

TIMSK2 &= ~(1<<TOIE2);
// executa tarefa
readAndSend();
counter = 0;
TCNT2 = tcnt2;
TIMSK2 |= (1<<TOIE2);
}
// Sleep
sleep_enable();
sleep_mode();
sleep_disable();
}

//////////////SHT15 From: http://bildr.org/2012/11/sht15-arduino/
float getTemperature(){
  //Return Temperature in Celsius
  SHT_sendCommand(B00000011, SHT_dataPin, SHT_clockPin);
  SHT_waitForResult(SHT_dataPin);

  int val = SHT_getData(SHT_dataPin, SHT_clockPin);
  SHT_skipCrc(SHT_dataPin, SHT_clockPin);
  return (float)val * 0.01 - 40; //convert to celsius
}

float getHumidity(){
  //Return  Relative Humidity
  SHT_sendCommand(B00000101, SHT_dataPin, SHT_clockPin);
  SHT_waitForResult(SHT_dataPin);
  int val = SHT_getData(SHT_dataPin, SHT_clockPin);
  SHT_skipCrc(SHT_dataPin, SHT_clockPin);
  return -4.0 + 0.0405 * val + -0.0000028 * val * val;
}

void SHT_sendCommand(int command, int dataPin, int clockPin){
  // send a command to the SHTx sensor
  // transmission start
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  digitalWrite(dataPin, HIGH);
  digitalWrite(clockPin, HIGH);
  digitalWrite(dataPin, LOW);
  digitalWrite(clockPin, LOW);
  digitalWrite(clockPin, HIGH);
  digitalWrite(dataPin, HIGH);
  digitalWrite(clockPin, LOW);

  // shift out the command (the 3 MSB are address and must be 000, the
last 5 bits are the command)
  shiftOut(dataPin, clockPin, MSBFIRST, command);

  // verify we get the right ACK
  digitalWrite(clockPin, HIGH);
  pinMode(dataPin, INPUT);

  if (digitalRead(dataPin)) Serial.println("ACK error 0");
  digitalWrite(clockPin, LOW);
  if (!digitalRead(dataPin)) Serial.println("ACK error 1");
}
```

```
void SHT_waitForResult(int dataPin){
  // wait for the SHTx answer
  pinMode(dataPin, INPUT);

  int ack; //acknowledgement

  //need to wait up to 2 seconds for the value
  for (int i = 0; i < 1000; ++i){
    delay(2);
    ack = digitalRead(dataPin);
    if (ack == LOW) break;
  }

  if (ack == HIGH) Serial.println("ACK error 2");
}

int SHT_getData(int dataPin, int clockPin){
  // get data from the SHTx sensor

  // get the MSB (most significant bits)
  pinMode(dataPin, INPUT);
  pinMode(clockPin, OUTPUT);
  byte MSB = shiftIn(dataPin, clockPin, MSBFIRST);

  // send the required ACK
  pinMode(dataPin, OUTPUT);
  digitalWrite(dataPin, HIGH);
  digitalWrite(dataPin, LOW);
  digitalWrite(clockPin, HIGH);
  digitalWrite(clockPin, LOW);

  // get the LSB (less significant bits)
  pinMode(dataPin, INPUT);
  byte LSB = shiftIn(dataPin, clockPin, MSBFIRST);
  return ((MSB << 8) | LSB); //combine bits
}

void SHT_skipCrc(int dataPin, int clockPin){
  // skip CRC data from the SHTx sensor
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  digitalWrite(dataPin, HIGH);
  digitalWrite(clockPin, HIGH);
  digitalWrite(clockPin, LOW);
}
```

Anexo 4 – Imagens e código do protótipo criado para medição da temperatura, da humidade, de CO_2, de COV e de partículas em suspensão

Nesta secção apresenta-se um conjunto de fotografias do protótipo criado para a medição da temperatura, da humidade, de CO_2, de COV e de partículas em suspensão e o respetivo código.

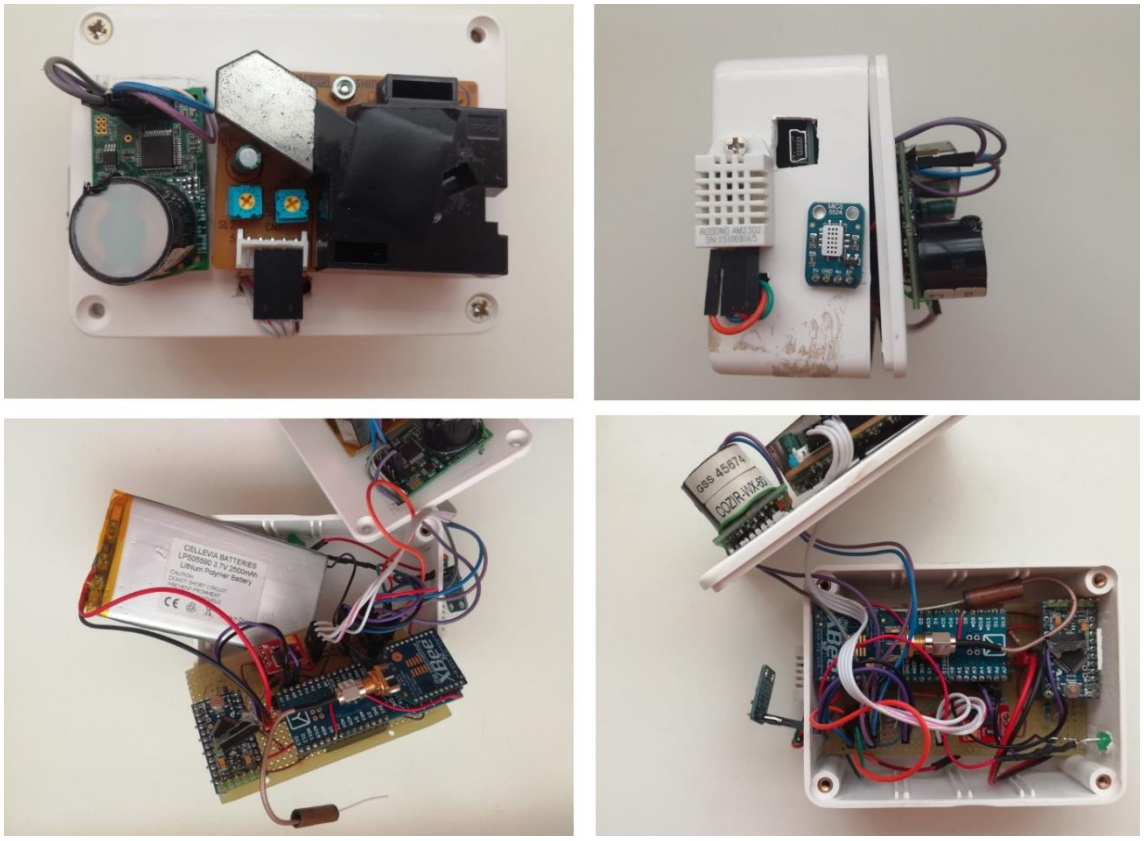


Figura 7.3 - Fotografias do protótipo criado para a medição da temperatura, da humidade, de CO_2, de COV e de partículas em suspensão.

Código para o Arduíno FIO (Master)

```
/*Universidade da Madeira 2016/2017
*Projeto de Mestrado: Monitorização habitacional com uma rede de sensores sem fios
*João Francisco Azevedo Baptista
*Código para o sensor de CO2 COV PM  TEMP HUM (Master)*/
#include "DHT.h"
#define DHTPIN 4     // what digital pin we're connected to
// Uncomment whatever type you're using!
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
DHT dht(DHTPIN, DHTTYPE);
#include <Wire.h>
long e = 0;
long d = 0;
String in = "";
int XBeeSleep = 8;
```

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(5, 6); // RX, TX
String inString = "";
int n = 0;
long b = 0;
long bant = 0;
long erro = 0;
long cc = 0;
unsigned long tempoamostraco2 = 10000;
unsigned long iniciotempoco2 = 0;
int inChar ='A';
char zchar='A';
int f=0;
int whiles=0;
int co2 = 0;
unsigned long tempoamostrarssi = 60000;
unsigned long iniciotemporssi = 0;
int db=0;
int a = 0;
int cov = 0;
int covt = 0;
int x = 0;
float cov7 = 0.000;
unsigned long tempoamostracov = 15000;
unsigned long iniciotempocov = 0;
float humi = 0.00;
float temp = 0.00;
float k = 0.00;
float j = 0.00;
unsigned long tempoamostrath = 7000;
unsigned long iniciotempoth = 0;
unsigned long starttime;
unsigned long triggerOnP1;
unsigned long triggerOffP1;
unsigned long pulseLengthP1;
unsigned long durationP1;
boolean valP1 = HIGH;
boolean triggerP1 = false;
unsigned long triggerOnP2;
unsigned long triggerOffP2;
unsigned long pulseLengthP2;
unsigned long durationP2;
boolean valP2 = HIGH;
boolean triggerP2 = false;
float ratioP1 = 0;
float ratioP2 = 0;
unsigned long sampletime_ms = 30000;
float countP1;
float countP2;
long pm25 = 0;
long pm10 = 0;

byte tramaco2[] = {0x7E, 0x00, 0x32, 0x10, 0x00, 0x00, 0x13, 0xA2, 0x00,
0x40, 0x2D, 0x25, 0x36, 0xFF, 0xFE, 0x00, 0x00, '?', 'W', 'C', 0x3D,
0x00, 0x00, 0x00, 0x00, 0x00, 0x26, 'W', 'V', 0x3D, 0x00, 0x00, 0x00,
0x00, 0x00, 0x26, 'W', 'T', 0x3D, 0x00, 0x00, 0x00, 0x00, 0x00, 0x26,
'W', 'H', 0x3D, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte checksum = 0;
int i = 0;
```

```
byte tramar[] = {0x7E, 0x00, 0x20, 0x10, 0x00, 0x00, 0x13, 0xA2, 0x00,
0x40, 0x2D, 0x25, 0x36, 0xFF, 0xFE, 0x00, 0x00, '?', 'W', 'R', 0x3D,
0x00, 0x00, 0x00, 0x00, 0x00, 0x26, 'W', 'B', 0x3D, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00};
byte checksume = 0;
int ie = 0;
int jaa=0;
float bat =0.00;

byte tramapm[] = {0x7E, 0x00, 0x23, 0x10, 0x00, 0x00, 0x13, 0xA2, 0x00,
0x40, 0x2D, 0x25, 0x36, 0xFF, 0xFE, 0x00, 0x00, '?', 'P', '1', 0x3D,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x26, 'P', '2', 0x3D, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
byte checksum3 = 0;
int i3 = 0;

void setup() {
  pinMode(XBeeSleep, OUTPUT);
  Serial.begin(9600);
  mySerial.begin(9600);
  Wire.begin();                 // join i2c bus with address #8
  dht.begin();
  mySerial.write("K 2\r\n");  // set polling mode CO2
}

void loop() {
  if ((millis() - iniciotemporssi) > tempoamostrarssi) {
//bateria
bat = analogRead(1);
float batv = 0.00;
float batva=0.00;
batva=(bat*5);
batv=batva/1023;

int auxbat1=batv;
float auxbat2 = (batv-auxbat1);
float auxbat3=0.00;
auxbat3=auxbat2*100;

int auxbat4 = auxbat3;
char acBuf99[5 + 1];
sprintf( acBuf99, "%2d.%2d", auxbat1, auxbat4);
tramar[30]=acBuf99[0];
tramar[31]=acBuf99[1];
tramar[32]=acBuf99[2];
if (auxbat4<10){
  tramar[33]='0';
  tramar[34]=acBuf99[4];
  }
  else{
    tramar[33]=acBuf99[3];
    tramar[34]=acBuf99[4];

  }

  // Wake up XBee
digitalWrite(XBeeSleep, LOW); // Liga XBee
delay(15); // Espera que acorde

//rssi
    rssiTest();
```

```
    int rssiv = db*(-1);
    char acBuf2[5 + 1];
    sprintf(acBuf2, "%5d", rssiv);//converte valor pa string
    tramar[21] = acBuf2[0];
    tramar[22] = acBuf2[1];
    tramar[23] = acBuf2[2];
    tramar[24] = acBuf2[3];
    tramar[25] = acBuf2[4];
    checksum = 0;
    for ( i = 3; i < sizeof(tramar) - 1; i++) {
      checksum += tramar[i]; // Soma os valores que contam para checksum
    }
    tramar[sizeof(tramar) - 1] = 0xFF - checksum; // Realiza o
complemento para 2
    Serial.write(tramar, sizeof(tramar)); // Envia a trama pela porta
série

// Sleep XBee
delay(100);
digitalWrite(XBeeSleep, HIGH);
    cc = 0;
    //n=0;
    iniciotemporssi = millis();
    jaa=1;
  }

  if ((millis() - iniciotempoco2) > tempoamostraco2) {
 LeCO2();
    char acBuf2[5 + 1];
    sprintf(acBuf2, "%5d", co2);//converte valor pa string
    tramaco2[21] = acBuf2[0];
    tramaco2[22] = acBuf2[1];
    tramaco2[23] = acBuf2[2];
    tramaco2[24] = acBuf2[3];
    tramaco2[25] = acBuf2[4];
  LeCOV();
char acBuf3[5 + 1];
int auxcovint = cov7;//parte inteira
float auxcovfl = 0.00;
auxcovfl = (cov7-auxcovint);
float auxcovf2 = 0.00;
  auxcovf2=auxcovfl*100;
  int auxcovf3 = auxcovf2;
  if (auxcovint<0){
    auxcovint=666;
    auxcovf3=66;
  }
    sprintf( acBuf3, "%2d.%2d", auxcovint, auxcovf3); //converte valor
pa string
    tramaco2[30] = acBuf3[0];
    tramaco2[31] = acBuf3[1];
    tramaco2[32] = acBuf3[2];
  if (auxcovf2<10){
  tramaco2[33] = '0';
    tramaco2[34] = acBuf3[4];
  }
  else {
  tramaco2[33] = acBuf3[3];
    tramaco2[34] = acBuf3[4];
  }
```

```
//temp humi
    LeTH();
int auxt1a = temp; //parte int temp
int auxh1a = humi; //parte int humi
float auxt2a = 0.00; //parte dec temp
float auxh2a = 0.00; //part dec humi
auxt2a = (temp-auxt1a); //parte dec temp
auxh2a = (humi-auxh1a); //part dec humi

float aauxt3a = 0.00;
float aauxh3a = 0.00;

aauxt3a = auxt2a*100; //parte dec temp convertida em inteiro
aauxh3a = auxh2a*100; //parte dec humi convertida em inteiro

int auxt3a = aauxt3a;
int auxh3a = aauxh3a;

    char acBuf4[5 + 1];

  sprintf( acBuf4, "%2d.%2d", auxt1a, auxt3a );
    tramaco2[39] = acBuf4[0];
    tramaco2[40] = acBuf4[1];
    tramaco2[41] = acBuf4[2];
  if (auxt3a<10){
    tramaco2[42] = '0';
  tramaco2[43] = acBuf4[4];
  }
  else{
  tramaco2[42] = acBuf4[3];
  tramaco2[43] = acBuf4[4];
  }

  sprintf( acBuf4, "%2d.%2d", auxh1a, auxh3a );
    tramaco2[48] = acBuf4[0];
    tramaco2[49] = acBuf4[1];
    tramaco2[50] = acBuf4[2];
  if (auxh3a<10){
  tramaco2[51] = '0';
  tramaco2[52] = acBuf4[4];
  }
  else{
  tramaco2[51] = acBuf4[3];
  tramaco2[52] = acBuf4[4];
  }
// Wake up XBee
digitalWrite(XBeeSleep, LOW); // Liga XBee
delay(15); // Espera que acorde

    checksum = 0;
    for ( i = 3; i < sizeof(tramaco2) - 1; i++) {
      checksum += tramaco2[i]; // Soma os valores que contam para
checksum
    }
    tramaco2[sizeof(tramaco2) - 1] = 0xFF - checksum; // Realiza o
complemento para 2
    Serial.write(tramaco2, sizeof(tramaco2)); // Envia a trama pela
porta série
// Sleep XBee
delay(100);
digitalWrite(XBeeSleep, HIGH);
```

```
    cc = 0;
    //n=0;
    iniciotempoco2 = millis();
  }
 if ((millis() - starttime) > sampletime_ms){
      Wire.requestFrom(8, 14);    // request 6 bytes from slave device
#8

  while (Wire.available()) {
 char c = Wire.read(); // receive a byte as character
    if (c=='\n'){
      pm25 = in.toInt();
      in="";
char acBuf5[7+1];
float pm10a = 0.00;
pm10a = pm10/100;
int pm10a1 = pm10a;
float pm10a2 = 0.00;
pm10a2 = pm10a - pm10a1;
float pm10a3 = 0.00;
pm10a3 = pm10a2*100;
int pm10a4 = pm10a3;

if (pm10a1<0){
  pm10a1=666;
  pm10a4=66;
}
sprintf(acBuf5, "%4lu.%2d", pm10a1, pm10a4);//converte valor pa string
tramapm[21]=acBuf5[0];
tramapm[22]=acBuf5[1];
tramapm[23]=acBuf5[2];
tramapm[24]=acBuf5[3];
tramapm[25]=acBuf5[4];

if (pm10a4<10){
tramapm[26]='0';
tramapm[27]=acBuf5[6];
}
else {
tramapm[26]=acBuf5[5];
tramapm[27]=acBuf5[6];
}
char acBuf6[6+1];
float pm25a = 0.00;
pm25a = pm25/100;
int pm25a1 = pm25a;
float pm25a2 =0.00;
pm25a2 = pm25a - pm25a1;
float pm25a3 = 0.00;
pm25a3 = pm25a2*100;
int pm25a4 = pm25a3;

if (pm25a1<0){
  pm25a1=666;
  pm25a4=66;
}
sprintf(acBuf6, "%3lu.%2d", pm25a1, pm25a4);//converte valor pa string
tramapm[32]=acBuf6[0];
tramapm[33]=acBuf6[1];
tramapm[34]=acBuf6[2];
tramapm[35]=acBuf6[3];
```

```
if (pm25a4<10){
    tramapm[36]='0';
    tramapm[37]=acBuf6[5];
}
else {
    tramapm[36]=acBuf6[4];
    tramapm[37]=acBuf6[5];
}

digitalWrite(XBeeSleep, LOW); // Liga XBee
delay(15); // Espera que acorde

checksum3 = 0;
for ( i3 = 3; i3 < sizeof(tramapm)-1; i3++) {
checksum3+= tramapm[i3]; // Soma os valores que contam para checksum
}
tramapm[sizeof(tramapm)-1] = 0xFF - checksum3; // Realiza o complemento
para 2
Serial.write(tramapm, sizeof(tramapm)); // Envia a trama pela porta
série
// Sleep XBee
delay(100);
digitalWrite(XBeeSleep, HIGH);

starttime=millis();

    }
    else if (c=='&'){
    pm10 = in.toInt();
    in="";
    }
    else{
    in += c;
    }
  }
  }

}

void LeCOV(){
float a7 = analogRead(A7);
cov7= (a7/1024)*100;
}
void LeTH(){
  k = dht.readHumidity();
  humi = k;
  j = dht.readTemperature();
  temp = j;
}
void LeCO2(){
mySerial.write("Z\r\n");
  while (zchar != 'Z'){
    whiles=whiles+1;
      if (whiles>150){
        f=1;
        zchar='Z';
        }
      else {
      zchar=mySerial.read();
        }
      }
```

```
if (f==1){
  f=0;
  whiles=0;
  b=bant;
  }
else{
  whiles=0;
    while (inChar != '\n'){
    inChar = mySerial.read();
      if (isDigit(inChar)) {
        inString += (char)inChar;
        }
      }
  b = inString.toInt();
    if (b<0){
    b=bant;
      }
  inString="";
  zchar = 'A';
  inChar = 'A';
  bant=b;
 }
  co2 = b*10;
  }
   void rssiTest(){
  byte                                trssi[]                                =
{0x7E,0x00,0x0F,0x17,0x01,0x00,0x13,0xA2,0x00,0x40,0x2D,0x25,0x36,0xFF
,0xFE,0x01,0x44,0x42,0xE6};
   Serial.write(trssi, sizeof(trssi));
 while (Serial.available() > 0) {
    int inChar = Serial.read();
    if (inChar == 'B') {
   int taa = Serial.read();
      db = Serial.read();
    }
  }
      jaa=0;
  }
```

Código para o Arduíno Pro Mini (Slave)

```
/*Universidade da Madeira 2016/2017
*Projeto de Mestrado: Monitorização habitacional com uma rede de sensores
sem fios
*João Francisco Azevedo Baptista
*Código para o sensor de CO2 COV PM  TEMP HUM (Slave)*/
#include <Wire.h>
unsigned long starttime;
unsigned long triggerOnP1;
unsigned long triggerOffP1;
unsigned long pulseLengthP1;
unsigned long durationP1;
boolean valP1 = HIGH;
boolean triggerP1 = false;
unsigned long triggerOnP2;
```

```
unsigned long triggerOffP2;
unsigned long pulseLengthP2;
unsigned long durationP2;
boolean valP2 = HIGH;
boolean triggerP2 = false;
float ratioP1 = 0;
float ratioP2 = 0;
unsigned long sampletime_ms =30000;
float countP1;
float countP2;
unsigned long pm10;
unsigned long pm25;
long num=0;
long num2=0;
char Buf[7+1];
char Buf1[5+1];
char Buf2[14+1];

void setup() {
  Wire.begin(8);                // join i2c bus with address #8
  Wire.onRequest(requestEvent); // register event
      pinMode(8, INPUT);
  pinMode(7, INPUT);
    Serial.begin(9600);
}
void loop() {
valP1 = digitalRead(8);
  valP2 = digitalRead(7);

  if(valP1 == LOW && triggerP1 == false){
    triggerP1 = true;
    triggerOnP1 = micros();
  }

  if (valP1 == HIGH && triggerP1 == true){
      triggerOffP1 = micros();
      pulseLengthP1 = triggerOffP1 - triggerOnP1;
      durationP1 = durationP1 + pulseLengthP1;
      triggerP1 = false;
  }

    if(valP2 == LOW && triggerP2 == false){
    triggerP2 = true;
    triggerOnP2 = micros();
  }

    if (valP2 == HIGH && triggerP2 == true){
      triggerOffP2 = micros();
      pulseLengthP2 = triggerOffP2 - triggerOnP2;
      durationP2 = durationP2 + pulseLengthP2;
      triggerP2 = false;
  }
  if ((millis() - starttime) > sampletime_ms) {

      ratioP1 = durationP1/(sampletime_ms*10.0);  // Integer percentage
0=>100
      ratioP2 = durationP2/(sampletime_ms*10.0);
      countP1 = 1.1*pow(ratioP1,3)-3.8*pow(ratioP1,2)+520*ratioP1+0.62;
      countP2 = 1.1*pow(ratioP2,3)-3.8*pow(ratioP2,2)+520*ratioP2+0.62;
      float PM10count = countP2; //  (particulas/0.01 cubic foot)
```

```
      float PM25count = countP1 - countP2; //     (particulas/0.01 cubic
foot)
      // first, PM10 count to mass concentration conversion
      double r10 = 2.6*pow(10,-6);
      double pi = 3.14159;
      double vol10 = (4/3)*pi*pow(r10,3);
      double density = 1.65*pow(10,12);
      double mass10 = density*vol10;
      double K = 3531.5;
      float concLarge = (PM10count)*K*mass10;
      // next, PM2.5 count to mass concentration conversion
      double r25 = 0.44*pow(10,-6);
      double vol25 = (4/3)*pi*pow(r25,3);
      double mass25 = density*vol25;
      float concSmall = (PM25count)*K*mass25;
pm10 = concLarge*100;
pm25 = concSmall*100;

      durationP1 = 0;
      durationP2 = 0;
      starttime = millis();
}
}
// function that executes whenever data is requested by master
// this function is registered as an event, see setup()
void requestEvent() {
  sprintf(Buf,"%7lu",pm10);
  sprintf(Buf1,"%5lu",pm25);
  Buf2[0]=Buf[0];
  Buf2[1]=Buf[1];
  Buf2[2]=Buf[2];
  Buf2[3]=Buf[3];
  Buf2[4]=Buf[4];
  Buf2[5]=Buf[5];
  Buf2[6]=Buf[6];
  Buf2[7]='&';
  Buf2[8]=Buf1[0];
  Buf2[9]=Buf1[1];
  Buf2[10]=Buf1[2];
  Buf2[11]=Buf1[3];
  Buf2[12]=Buf1[4];
  Buf2[13]='\n';
  Wire.write(Buf2);           // sends
}
// Wire Master Writer
// by Nicholas Zambetti <http://www.zambetti.com>

// Demonstrates use of the Wire library
// Writes data to an I2C/TWI slave device
// Refer to the "Wire Slave Receiver" example for use with this

// Created 29 March 2006

// This example code is in the public domain.
```

Anexo 5 – Imagens e código do protótipo criado para a medição de corrente no quadro elétrico

Nesta secção apresenta-se um conjunto de fotografias do protótipo criado para a medição de corrente no quadro elétrico e o respetivo código.

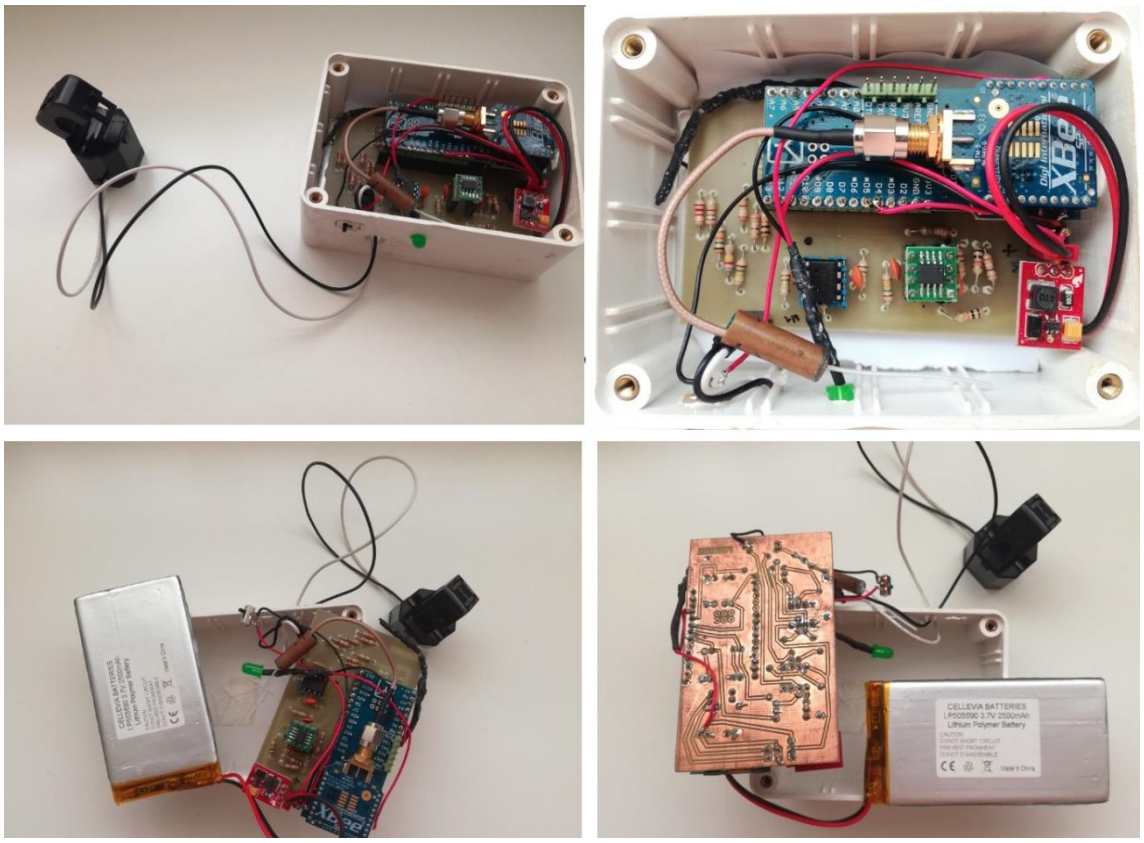


Figura 7.4- Fotografias do protótipo criado para a medição de corrente no quadro elétrico.

```
/*Universidade da Madeira 2016/2017
*Projeto de Mestrado: Monitorização habitacional com uma rede de sensores
sem fios
*João Francisco Azevedo Baptista
*Código para o sensor de corrente no quadro elétrico*/
#include <avr/sleep.h>
unsigned int tcnt2;
int seconds = 10; // Período em segundos
int start = seconds*30;
int counter = 0;
int XBeeSleep = 4;
float IrmsA3=0;
byte checksum = 0;
int vv=0;
byte                                trama2[]                                =
{0x7E,0x00,0x28,0x10,0x00,0x00,0x13,0xA2,0x00,0x40,0x2D,0x25,0x36,0xFF
,0xFE,0x00,0x00,'?','C','7',0x3D,0x00,0x00,0x00,0x00,0x00,0x26,'N','8'
,0x3D,0x00,0x00,0x00,0x00,0x26,'N','7',0x3D,0x00,0x00,0x00,0x00,0x00,0
x00};
//C7 COMECA EM TRAMA2[21] E TERMINA EM TRAMA2[25]
int w=0;
int db=0;
```

```
int pp=0;
float val1=0;
float IrmsA0=0;
float IrmsA2=0;
float IrmsA1=0;
int valmax=0;
int valmax1=0;
int valmax2=0;
int valmax3=0;
int bat=0;
unsigned long ARR[200];
int i=0;
int readValue =0;
uint32_t start_time =0;
uint32_t somatorio=0;
uint32_t somatorio2=0;
uint32_t somatorio3=0;
uint32_t somatorio31=0;
uint32_t media=0;
uint32_t mark=0;
int corrente=0;
int p;
void setup(){
 Serial.begin(9600);
  start_time = micros();
  p = 0;
  Serial.println("T=Periodo medido; N=Quantidade de amostras; G=
Ganho;");
  pinMode(XBeeSleep, OUTPUT);
/* First disable the timer overflow interrupt while we're configuring
*/
TIMSK2 &= ~(1<<TOIE2);
/* Configure timer2 in normal mode (pure counting, no PWM etc.) */
TCCR2A &= ~((1<<WGM21) | (1<<WGM20));
TCCR2B &= ~(1<<WGM22);
/* Select clock source: internal I/O clock */
ASSR &= ~(1<<AS2);
/* Disable Compare Match A interrupt enable (only want overflow) */
TIMSK2 &= ~(1<<OCIE2A);
/* Now configure the prescaler to CPU clock divided by 1024 */
TCCR2B |= (1<<CS22) | (1<<CS21) | (1<<CS20); // Set bits
tcnt2 = 0;
TCNT2 = tcnt2;
TIMSK2 |= (1<<TOIE2);
set_sleep_mode(SLEEP_MODE_PWR_SAVE);
sleep_enable();
}
ISR(TIMER2_OVF_vect) {
/* Reload the timer */
TCNT2 = tcnt2;
counter++;
}
 void rssiTest(){
  byte                                trssi[]                                =
{0x7E,0x00,0x0F,0x17,0x01,0x00,0x13,0xA2,0x00,0x40,0x2D,0x25,0x36,0xFF
,0xFE,0x01,0x44,0x42,0xE6};
   Serial.write(trssi, sizeof(trssi));
 while (Serial.available() > 0) {
    int inChar = Serial.read();
    if (inChar == 'B') {
   int t = Serial.read();
```

```
      db = Serial.read();
    }
  }
      pp=0;
  }
void LeVal(){
if (p==0){
readValue = analogRead(A3);
ARR[i]=readValue;
if (valmax<readValue){
  valmax=readValue;
}
i=i+1;
   if((micros()-start_time) >= 20000)
   {
    mark=micros();
    int x=i;
     Serial.println("");
    Serial.print("A3. G=0.183 |");
    uint32_t periodo=mark-start_time;
    Serial.print(" VALMAX = ");
Serial.print(valmax);
    Serial.print(", T = ");
    Serial.print(periodo);
Serial.print(", N = ");
Serial.print(x);
 for (int j = 0; j < x; j = j + 1) {
      unsigned long aux2=ARR[j];
      ARR[j]=aux2*aux2;
    }
 for (int j = 0; j < x; j = j + 1) {
      unsigned long aux3=ARR[j];
      somatorio2 = somatorio2 + aux3;
      ARR[j]=0;
    }
    int loo=x/2;
unsigned long aux4=somatorio2/loo;
unsigned long Arms=sqrt(aux4);
Serial.print(", V_EFICAZ = ");
Serial.print(Arms);
IrmsA3=((Arms-4.2652)/0.0089);
Serial.print(", Irms = ");
Serial.print(IrmsA3);
Serial.println(" mA");
  i=0;
 // delay(500);
  somatorio2=0;
  somatorio=0;
  media=0;
    p=1; //para fazer agora o A1 val
start_time = micros();
   }
}
else if (p==1){
readValue = analogRead(A1);
ARR[i]=readValue;
if (valmax1<readValue){
  valmax1=readValue;
}
i=i+1;
if((micros()-start_time) >= 20000) //sample for 20ms
```

```
   {
mark = micros();
    int f=i;
  Serial.print("A1. G=10 |");

    for (int j = 0; j < f; j = j + 1) {
      unsigned long aux6=ARR[j];
      ARR[j]=aux6*aux6;
  }
      for (int j = 0; j < f; j = j + 1) {
      unsigned long aux5=ARR[j];
somatorio3 = somatorio3 + aux5;
    ARR[j]=0;
    }
int y=f/2;
unsigned long aux7=somatorio3/y;
unsigned long Arms1=sqrt(aux7);
IrmsA1=(Arms1+74.325)/1.1917;
Serial.print(" VALMAX = ");
Serial.print(valmax1);
Serial.print(" T = ");
unsigned long tmed=mark-start_time;
Serial.print(tmed);
Serial.print(", N = ");
Serial.print(f);
Serial.print(", V_EFICAZ ");
Serial.print(Arms1);
Serial.print(", Irms = ");
Serial.print(IrmsA1);
Serial.println(" mA");
i=0;
somatorio3=0;
p=2;
start_time = micros();
}
}
else if (p==2){
 readValue = analogRead(A0);
ARR[i]=readValue;
if (valmax2<readValue){
  valmax2=readValue;
}
i=i+1;
if((micros()-start_time) >= 20000) //sample for 20ms
   {
mark = micros();
    int g=i;
  Serial.print("A0. G=100 | ");

    for (int j = 0; j < g; j = j + 1) {
      unsigned long aux6=ARR[j];
      ARR[j]=aux6*aux6;
  }
      for (int j = 0; j < g; j = j + 1) {
      unsigned long aux5=ARR[j];
somatorio3 = somatorio3 + aux5;
    ARR[j]=0;
    }
int y=g/2;
unsigned long aux7=somatorio3/y;
unsigned long Arms3=sqrt(aux7);
```

```
IrmsA0=(Arms3-5.5323)/4.4248;
Serial.print(" VALMAX = ");
Serial.print(valmax2);
Serial.print(" T = ");
unsigned long tmed=mark-start_time;
Serial.print(tmed);
Serial.print(", N = ");
Serial.print(g);
Serial.print(", V_EFICAZ ");
Serial.print(Arms3);
Serial.print(", Irms = ");
Serial.print(IrmsA0);
Serial.println(" mA");
i=0;
somatorio3=0;
p=3;
start_time = micros();
 }
}
else {
 readValue = analogRead(A2);
ARR[i]=readValue;
if (valmax3<readValue){
  valmax3=readValue;
}
i=i+1;
if((micros()-start_time) >= 20000) //sample for 20ms
   {
mark = micros();
    int g1=i;
  Serial.print("A2. G=1 |");

    for (int j = 0; j < g1; j = j + 1) {
      unsigned long aux61=ARR[j];
      ARR[j]=aux61*aux61;
  }
      for (int j = 0; j < g1; j = j + 1) {
      unsigned long aux51=ARR[j];
somatorio31 = somatorio31 + aux51;
    ARR[j]=0;
    }
int y1=g1/2;
unsigned long aux71=somatorio31/y1;
unsigned long Arms4=sqrt(aux71);
IrmsA2=((Arms4-31.13)/0.0636);
Serial.print(" VALMAX = ");
Serial.print(valmax3);
Serial.print(" T = ");
unsigned long tmed=mark-start_time;
Serial.print(tmed);
Serial.print(", N = ");
Serial.print(g1);
Serial.print(", V_EFICAZ ");
Serial.print(Arms4);
Serial.print(", Irms = ");
Serial.print(IrmsA2);
Serial.println(" mA");
i=0;
somatorio31=0;
p=0;
valmax=0;
```

```
valmax1=0;
valmax2=0;
valmax3=0;
delay(10);
start_time = micros();
 }
}
}
void Mostra(){
char acBuf3[6+1];
char acBuf[5+1];
char acBuf1[4+1];
if (IrmsA0 < 113){
  if (IrmsA0<0){
    corrente = 0;
  }
  else{
      corrente = IrmsA0;
  }
  Serial.println(corrente);
  sprintf(acBuf, "%5d", corrente);//converte IrmsA3 pa string
trama2[21]=acBuf[0];
trama2[22]=acBuf[1];
trama2[23]=acBuf[2];
trama2[24]=acBuf[3];
trama2[25]=acBuf[4];
Serial.println("A0 seleccionado");}
else if (IrmsA2>1900){
  corrente=IrmsA3;
       if (corrente<0){
      corrente=0;
     }
  sprintf(acBuf, "%5d", corrente);//converte valor pa string
trama2[21]=acBuf[0];
trama2[22]=acBuf[1];
trama2[23]=acBuf[2];
trama2[24]=acBuf[3];
trama2[25]=acBuf[4];
Serial.println("A3 seleccionado");}
else if (IrmsA1>840){
   corrente=IrmsA2;
        if (corrente<0){
      corrente=0;
     }
  sprintf(acBuf, "%5d", corrente);//converte valor pa string
trama2[21]=acBuf[0];
trama2[22]=acBuf[1];
trama2[23]=acBuf[2];
trama2[24]=acBuf[3];
trama2[25]=acBuf[4];
  Serial.println("A2 seleccionado");
}
else {
     corrente=IrmsA1;
     if (corrente<0){
      corrente=0;
     }
  sprintf(acBuf, "%5d", corrente);//converte valor pa string
trama2[21]=acBuf[0];
trama2[22]=acBuf[1];
trama2[23]=acBuf[2];
```

```
trama2[24]=acBuf[3];
trama2[25]=acBuf[4];
  Serial.println("A1 seleccionado");
}
int dbneg=db*(-1);
sprintf(acBuf1, "%4d", dbneg);//converte valor pa string
trama2[30]=acBuf1[0];
trama2[31]=acBuf1[1];
trama2[32]=acBuf1[2];
trama2[33]=acBuf1[3];
bat = analogRead(4);
float batv = 0.00;
float batva=0.00;
batva=(bat*5);
batv=batva/1023;
int auxbat1=batv;
float auxbat2 = (batv-auxbat1);
float auxbat3=0.00;
auxbat3=auxbat2*100;
int auxbat4 = auxbat3;
char acBuf9[5 + 1];
sprintf( acBuf9, "%2d.%2d", auxbat1, auxbat4);
trama2[38]=acBuf9[0];
trama2[39]=acBuf9[1];
trama2[40]=acBuf9[2];
if (auxbat4<10){
  trama2[41]='0';
  trama2[42]=acBuf9[4];
  }
  else{
    trama2[41]=acBuf9[3];
    trama2[42]=acBuf9[4];
  }
// Wake up XBee
digitalWrite(XBeeSleep, LOW); // Liga XBee
delay(15); // Espera que acorde
pp=pp+1;
if (pp==16){
  rssiTest();
}
// Cálculo do checksum
checksum = 0;
for ( vv = 3; vv < sizeof(trama2)-1; vv++) {
checksum+= trama2[vv]; // Soma os valores que contam para checksum
}
trama2[sizeof(trama2)-1] = 0xFF - checksum; // Realiza o complemento
para 2
Serial.write(trama2, sizeof(trama2)); // Envia a trama pela porta série
delay(100);
// Sleep XBee
digitalWrite(XBeeSleep, HIGH);
}
void loop(){
LeVal();
if(counter >= start){
TIMSK2 &= ~(1<<TOIE2);
// executa tarefa
Mostra();
counter = 0;
TCNT2 = tcnt2;
TIMSK2 |= (1<<TOIE2);
```

```
}
// Sleep
sleep_enable();
sleep_mode();
sleep_disable();
}
```

Anexo 6 – Imagens e código do protótipo criado para a medição de corrente numa tomada elétrica

Nesta secção apresenta-se um conjunto de fotografias do protótipo criado para a medição de corrente na tomada elétrica e o respetivo código.

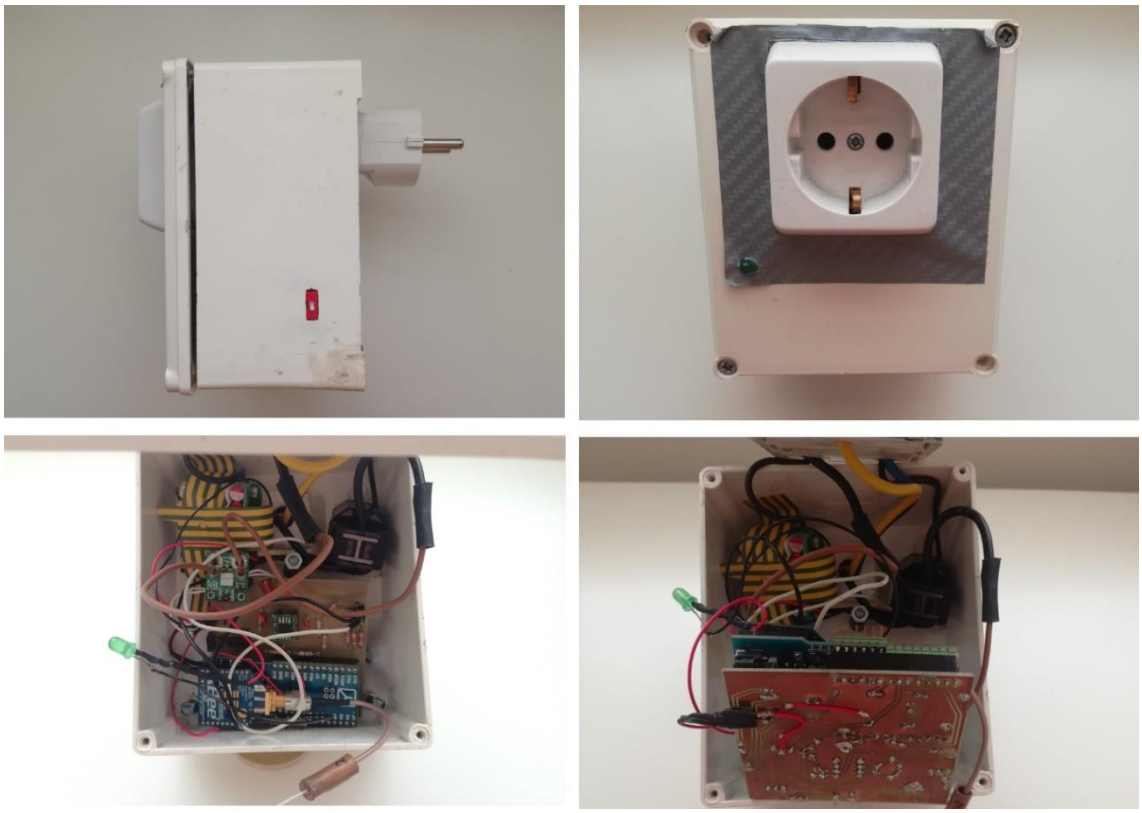


Figura 7.5 - Fotografias do protótipo criado para a medição de corrente numa tomada elétrica.

```
/*Universidade da Madeira 2016/2017
*Projeto de Mestrado: Monitorização habitacional com uma rede de sensores
sem fios
*João Francisco Azevedo Baptista
*Código para o sensor de corrente na tomada elétrica*/
float valor=0;
byte checksum = 0;
byte                                trama2[]                                =
{0x7E,0x00,0x27,0x10,0x00,0x00,0x13,0xA2,0x00,0x40,0x2D,0x25,0x36,0xFF
,0xFE,0x00,0x00,'?','J','3',0x3D,0x00,0x00,0x00,0x00,0x00,0x26,'G','3'
,0x3D,0x00,0x00,0x00,0x00,0x26,'D','3',0x3D,0x00,0x00,0x00,0x00,0x00};
int w=0;
int db=0;
float val1=0;
float Irms3=0;
float Irms2=0;
unsigned long ARR[200];
int i=0;
int readValue =0;
uint32_t start_time =0;
uint32_t somatorio=0;
uint32_t somatorio2=0;
uint32_t somatorio3=0;
```

```
uint32_t media=0;
uint32_t mark=0;
int corrente=0;
int p;
void setup(){
 Serial.begin(9600);
  start_time = micros();
  p = 0;
}
void loop(){
if (p==0){
readValue = analogRead(A0);
ARR[i]=readValue;
i=i+1;
   if((micros()-start_time) >= 20000)
   {
    mark=micros();
    int x=i;
     Serial.println("");
    Serial.print("DoneA0.");
    uint32_t periodo=mark-start_time;
    Serial.print(", Tempo de amostra = ");
    Serial.print(periodo);
////////////////
    for (int j = 0; j < x; j = j + 1) {
       somatorio = somatorio + ARR[j];
        }
media = somatorio/x;
Serial.print(", Numero de amostras ACS712");
Serial.print(x);
Serial.print(", Media ACS712 = ");
Serial.print(media);
    for (int j = 0; j < x; j = j + 1) {
      unsigned long aux=ARR[j];
           if (aux>=media){
               ARR[j]=aux-media;
              }
           else{
               ARR[j]=media-aux;
               }
            }

 for (int j = 0; j < x; j = j + 1) {
      unsigned long aux2=ARR[j];
      ARR[j]=aux2*aux2;
    }
 for (int j = 0; j < x; j = j + 1) {
      unsigned long aux3=ARR[j];
      somatorio2 = somatorio2 + aux3;
      ARR[j]=0;
    }
int aux4=somatorio2/x;
float Arms=sqrt(aux4);
//float Irms=Arms*mVperAmp;
Serial.print(", VALOR_EFICAZ = ");
Serial.print(Arms);

  valor=abs(((Arms-0.4621)/13.319));

Serial.print(", Corrente RMS = ");
Serial.print(valor);
```

```
Serial.println(" mA");
  i=0;
 // delay(500);
  somatorio2=0;
  somatorio=0;
  media=0;
    p=1; //para fazer agora o A1 val
start_time = micros();
   }
}

else if (p==1){
readValue = analogRead(A1);
ARR[i]=readValue;
i=i+1;
if((micros()-start_time) >= 20000) //sample for 20ms
   {
mark = micros();
    int f=i;
  Serial.print("DoneA1.");

    for (int j = 0; j < f; j = j + 1) {
      unsigned long aux6=ARR[j];
      ARR[j]=aux6*aux6;
  }

      for (int j = 0; j < f; j = j + 1) {
      unsigned long aux5=ARR[j];
somatorio3 = somatorio3 + aux5;
    ARR[j]=0;
    }
int y=f/2;
unsigned long aux7=somatorio3/y;
unsigned long Arms1=sqrt(aux7);
Irms2=(Arms1+108)/1.763;

Serial.print(" Periodo medido = ");
unsigned long tmed=mark-start_time;
Serial.print(tmed);
Serial.print(", Numero de amostras = ");
Serial.print(f);
Serial.print(", Valor eficaz ");
Serial.print(Arms1);
Serial.print(", Corrente RMS = ");
Serial.print(Irms2);
Serial.println(" mA");

i=0;
somatorio3=0;
p=2;
delay(5000);
start_time = micros();
}
}

else{

 readValue = analogRead(A2);
ARR[i]=readValue;
i=i+1;
if((micros()-start_time) >= 20000) //sample for 20ms
```

```
   {
mark = micros();
    int g=i;
  Serial.print("DoneA2.");

    for (int j = 0; j < g; j = j + 1) {
      unsigned long aux6=ARR[j];
      ARR[j]=aux6*aux6;
  }
      for (int j = 0; j < g; j = j + 1) {
      unsigned long aux5=ARR[j];
somatorio3 = somatorio3 + aux5;
    ARR[j]=0;
    }
int y=g/2;
unsigned long aux7=somatorio3/y;
unsigned long Arms3=sqrt(aux7);
Irms3=(Arms3+9.4265)/4.8683;
Serial.print(" Periodo medido = ");
unsigned long tmed=mark-start_time;
Serial.print(tmed);
Serial.print(", Numero de amostras = ");
Serial.print(g);
Serial.print(", Valor eficaz ");
Serial.print(Arms3);
Serial.print(", Corrente RMS = ");
Serial.print(Irms3);
Serial.println(" mA");

Mostra();
i=0;
somatorio3=0;
p=0;
start_time = micros();
 }
}

}
void Mostra(){
    char acBuf3[6+1];
char acBuf[5+1];
char acBuf1[4+1];
if (valor > 0.5){
  corrente =(int)(valor*1000);
  Serial.println(corrente);
  sprintf(acBuf, "%5d", corrente);//converte valor pa string
trama2[21]=acBuf[0];
trama2[22]=acBuf[1];
trama2[23]=acBuf[2];
trama2[24]=acBuf[3];
trama2[25]=acBuf[4];
Serial.println("A0 seleccionado");}
else if (Irms3<125){
  corrente=Irms3;
  sprintf(acBuf, "%5d", corrente);//converte valor pa string
trama2[21]=acBuf[0];
trama2[22]=acBuf[1];
trama2[23]=acBuf[2];
trama2[24]=acBuf[3];
trama2[25]=acBuf[4];
```

```
Serial.println("A2 seleccionado");}
else{

   corrente=Irms2;
  sprintf(acBuf, "%5d", corrente);//converte valor pa string
trama2[21]=acBuf[0];
trama2[22]=acBuf[1];
trama2[23]=acBuf[2];
trama2[24]=acBuf[3];
trama2[25]=acBuf[4];
  Serial.println("A1 seleccionado");
}

sprintf(acBuf1, "%4d", db);//converte valor pa string
trama2[30]=acBuf1[0];
trama2[31]=acBuf1[1];
trama2[32]=acBuf1[2];
trama2[33]=acBuf1[3];

sprintf(acBuf1, "%4d", w);//converte valor pa string
trama2[38]=acBuf1[0];
trama2[39]=acBuf1[1];
trama2[40]=acBuf1[2];
trama2[41]=acBuf1[3];

// Cálculo do checksum
checksum = 0;
for ( i = 3; i < sizeof(trama2)-1; i++) {
checksum+= trama2[i]; // Soma os valores que contam para checksum
}
trama2[sizeof(trama2)-1] = 0xFF - checksum; // Realiza o complemento
para 2
Serial.write(trama2, sizeof(trama2)); // Envia a trama pela porta série
}
```

Anexo 7 – Imagem do módulo coordenador

Nesta secção apresenta-se a imagem do módulo coordenador e do *Raspberry Pi* que foi utilizado.

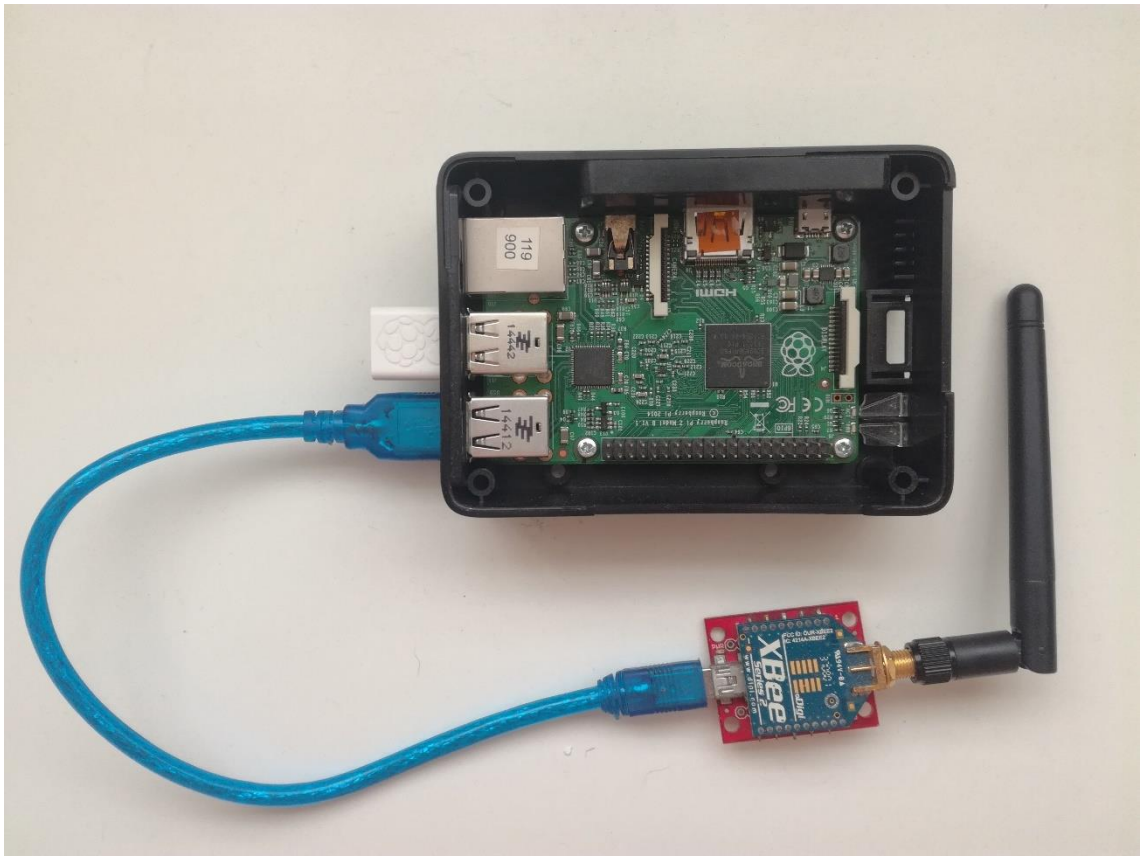


Figura 7.6 - Módulo coordenador e *Raspberry Pi* utilizado.