Universidade da Madeira
Centro de Competência de Ciências Exactas e da Engenharia

Monitorização Ambiental em Espaços Florestais com Rede de Sensores Sem Fios

Dissertação submetida à Universidade da Madeira para obtenção do Grau de Mestre em Engenharia de Telecomunicações e Redes

Orientador: Joaquim Amândio Rodrigues Azevedo

Tiago Couto Braga

Funchal, 29 de Novembro de 2010

Agradecimentos

Ao Professor Doutor Joaquim Amândio Azevedo, orientador da tese de mestrado, agradeço todo o apoio prestado. Quero ainda agradecer ao Centro de Ciências e Tecnologia da Madeira que, através do Professor Joaquim Azevedo, facultou-me uma bolsa de estudo.

Aos meus pais, irmãs e demais família que sempre me apoiaram em todo o meu percurso académico.

Ao Eng. Filipe Santos por todo o apoio, experiência e conhecimento partilhados, fazendo com que este tenha tido um papel de grande importância no decorrer deste projecto.

Um agradecimento especial para a minha namorada Maria que sempre me apoiou incondicionalmente ao longo da vida académica.

Um contributo ainda ao meus colegas do Laboratório de Telecomunicações: Bruno Gouveia, Maurício Rodrigues, Carla Santos, Tony Pereira, Jenny Gouveia, Hugo e a todos os restantes que com os seus comentários e ideias contribuíram para que este projecto se realizasse, o meu muito obrigado.

Resumo

O objectivo deste projecto é desenvolver um sistema de monitorização florestal recorrendo às Redes de Sensores sem Fios.

Para a monitorização florestal foi desenvolvido um protótipo cuja função é obter periodicamente os valores de temperatura, humidade, luminosidade, tensão nas baterias e a indicação do nível de sinal de rádio frequência recebido (RSSI- *Received Signal Strength Indicator*).

O equipamento referido foi instalado num ambiente exterior, com características semelhantes à de interesse de modo a permitir avaliar os efeitos do ambiente no desempenho da rede.

O funcionamento das Redes de Sensores sem Fios, baseadas no protocolo *ZigBee*, foi estudado e depois aplicado para transmitir os valores obtidos.

Os dados referidos percorrem a rede *ZigBee* até alcançar a estação base, que tem como função processar, manipular, armazenar numa base de dados e disponibilizar os dados em tempo real através de uma página de Internet. Como os dados são armazenados é sempre possível efectuar uma consulta à base de dados para realização de estudos e estatísticas.

Tendo em conta a capacidade limitada dos sistemas de armazenamento de energia utilizados em ambientes exteriores, foi desenvolvido um algoritmo que permite comutar os dispositivos na rede *ZigBee* de *router* para *end-device* e vice-versa, de modo a diminuir o consumo de energia e aumentar o tempo de vida da rede. Este algoritmo foi testado numa situação em que os nós sensores estão colocados em linha, existindo um único salto entre os mesmos.

Palavras-chave: Rede de Sensores sem Fios (RSSF), WSN, *ZigBee*, monitorização florestal.

Abstract

The aim of this project is the development of a monitor system making use of wireless sensor networks.

For monitoring the forest, we developed a prototype which, periodically, will record the levels of battery, temperature, humidity, luminosity, and the received signal strength indicator (RSSI).

The prototype was installed in an outside environment, with similar characteristics to the interested one, in order to allow the evaluation of the environment factors effects in the network performance.

The wireless sensor networks, supported by the ZigBee Protocol, were studied, and then applied for the data transmission.

The data flow through the ZigBee network in order to reach the gateway, which using a web page in real time, processes, manipulates, stores and makes them available in a data-base.

Since the data is stored, consulting the data-base for studies (researches) and statistics is always available.

Considering the power storage systems limited capacity when used in an outside environment, it was necessary to develop an algorithm to allow commuting the devices ZigBee net from *router* to *end-device* and vice-versa, in order to diminish the power waste and increase the network longevity.

That algorithm was tested in a situation that the sensor nodes are lined-up, when there is a single hop among them.

Key words: Wireless sensor network (WSN); *ZigBee;* Forest monitoring.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

AES	*Advanced Encryption Standard*
API	*Application Programming Interface*
AWACS	*Airborne Warning and Control System*
BPSK	*Binary Phase Shift Keying*
CCA	*Clear Channel Assessment*
CEC	*Cooperative Engagement Capability*
DARPA	*Defense Advanced Research Projects Agency*
DSN	*Distributed Sensor Networks*
ESR	*Equivalente Series Resistance*
FDS	*Fixed Distributed System*
FFSS	*Forest-Fire Surveillance System*
I²C	*Inter-Integrated Circuit*
IC	*Integrated Circuit*
ICSP	*In-Circuit Serial Programming*
ISM	*Industrial, Scientific and Medical*
JTAG	*Joint Test Action Group*
LED	*Light-Emitting Diode*
LQI	*Link Quality Indicator*
MAC	*Medium Access Control*
MANET	*Mobile Ad hoc Network*
MMCX	*Micro-Miniature Coaxial*
MOSFET	*Metal–Oxide–Semiconductor Field-Effect Transistor*
NiMH	*Nickel-Metal Hydride*
NOAA	*National Oceanographic and Atmospheric Administration*
OpenVMS	*Open Virtual Memory*(*System*
O-QPSK	*Offset Quadrature Phase- Shift Keying)*
PCI	Placa de circuito impresso
PD-SAP	*PHY Data Service Access Point*
PHY	*Physical Layer*
PHY PIB	*Physical Layer Pan Information Base*
PIB	*Pan Information Base*
PLME	*Physical Layer Management Entity*
PLME-SAP	*Physical Layer Management Entity-Service Access Point*
PPDU	*PHY Protocol Data Unit*
PVC	*Polyvinyl Chloride*
PWM	*Pulse With Modulation*
RED	*Received Energy Detection*
RISC	*Reduced Instruction Set Computer*
RPSMA	*Reverse Polarity Subminiature Version A*
RSSF	Rede de Sensores sem Fios
RTC	*Real Time Clock*
SAP	*Service Access Point*
SensIT	*Sensor Information Technology*
SMD	*Surface Mounted Device*
SOSUS-	*Sound Surveillance System*
SPI	*Serial Peripheral Interface*

THT	*Trough Hole Technology*
TQFP	*Thin Quad Flat Pack.*
U.FL	*Miniature coaxial rf connector for high-frequency signals*
USART	*Universal Synchronous/Asynchronous Receiver/Transmitter*
UV	*Ultra-Violet*
WASN	*Wireless Ad hoc Sensor Network*
WPAN	*Wireless Personal Area Network*
WSN	*Wireless Sensor Networks*

Prefácio

Este prefácio irá conter a motivação, os objectivos e por fim a estrutura que será desenvolvida ao longo deste trabalho.

Motivação

A ilha da Madeira já dispõe de sistemas de medição de temperatura e humidade junto à costa e nos picos, faltando no entanto sistemas de monitorização ao longo das encostas.

Será de grande interesse a colocação destes sistemas nas encostas para, por exemplo, prevenção de um possível incêndio a partir dos valores obtidos principalmente pelos sensores de humidade e temperatura e realização de estudos sobre as alterações climáticas

Este projecto pretende desenvolver um protótipo de monitorização ambiental com vista à instalação de uma rede de sensores sem fios nas zonas protegidas da Ilha da Madeira.

Para isso é necessário estudar a tecnologia mais apropriada e implementar uma rede de sensores sem fios que responda aos requisitos da aplicação

Objectivos

Nos dias que decorrem, o mundo electrónico já nos proporciona vários sensores com uma grande diversidade de aplicações. Será necessário proceder a um estudo, com a finalidade de assegurar quais os sensores essenciais para a monitorização ambiental em espaços florestais.

O segundo objectivo deste projecto consiste na implementação de uma rede de sensores sem fios. Estas redes de sensores permitem aos usuários interagirem remotamente com o mundo físico a partir de um computador.

A optimização da energia consumida pelos nós sensores será outro dos objectivos a ser estudado, atendendo a que a alimentação destes nós em ambientes florestais é actualmente efectuada com recurso a baterias. Como as baterias possuem uma autonomia limitada será de grande interesse utilizar formas de alimentação que permitam a captação da energia gerada pelo meio envolvente, nomeadamente energia solar, eólica e hídrica para que se possa garantir uma maior autonomia do nó sensor.

Por fim os dados obtidos deverão ser encaminhados para uma estação base, onde serão processados e disponibilizados em plataformas informáticas. A visualização dos dados será efectuada através de uma interface gráfica, com uma possibilidade de acesso *on-line*.

Estrutura

Este relatório está estruturado em 5 capítulos.

No capítulo inicial, “Rede de Sensores sem Fios”, pretende-se introduzir ao leitor o estado de arte envolvido no contexto deste trabalho.

No capítulo 2, “Relação entre *ZigBee* e o padrão 802.15.4” serão descritos conceitos sobre a tecnologia *ZigBee* em comparação com as restantes tecnologias sem fios.

No capítulo 3,”Concepção arquitectural”, é onde é descrita a arquitectura da rede, os componentes que integram um nó sensor e a arquitectura desenvolvida para o protótipo.

No capítulo 4, “Desenvolvimento do protótipo e do software”, abordam-se os processos de construção do protótipo, a organização dos vários módulos constituintes do protótipo no interior de caixas estanques, a configuração da rede *ZigBee*, as ferramentas de *software* utilizadas ao longo do trabalho. Será ainda descrita a técnica de adormecimento dos *routers* e a disposição geográfica dos nós sensores.

No capítulo 5,”Testes e Resultados” serão disponibilizados os testes e os resultados obtidos ao longo deste trabalho.

O capítulo final contém as conclusões e os trabalhos futuros.

Existem ainda informações relevantes, relacionadas com o trabalho em questão, na secção Anexos.

1 Rede de Sensores sem Fios

Neste capítulo é efectuado o levantamento do estado das Redes de Sensores sem Fios (RSSF) para monitorização ambiental, onde serão descritos os conceitos principais, os seus principais componentes, as suas características, os projectos desenvolvidos e as possíveis aplicações.

1.1 Introdução

As redes de sensores sem fios são uma tecnologia cujo principal objectivo é a extracção de informações do meio que as rodeia. Estas informações são inicialmente processadas e encaminhadas na maior parte das vezes por radiofrequência, sendo posteriormente disponibilizadas numa plataforma informática.

Os nós constituintes das redes incorporam módulos de detecção (sensores), um módulo de processamento, um módulo de comunicação e uma fonte de alimentação, que é geralmente uma bateria de baixo custo e dimensão reduzida. Os nós da rede cooperam entre si, de modo a que os dados recolhidos alcancem o nó com funções de coordenação e que esteja ligado a um computador, que processa a informação através de uma aplicação criada para o efeito.

A colocação dos nós no meio ambiente poderá ser feita de forma aleatória. Para zonas de difícil acesso poderá ser necessário recorrer a meios aéreos, tais como uma avioneta ou um helicóptero, para lançar os nós na região para ser monitorizada. Na maior parte das aplicações práticas estes são colocados manualmente na zona a monitorizar.

Visto que a colocação dos sensores pode ser efectuada em zonas de difícil acesso, é necessário garantir *à priori* o funcionamento do nó por longos períodos. O consumo de energia deve ser o mais optimizado possível, garantindo, desta forma, uma maior durabilidade. O processo de transmissão dos dados é a actividade que consome mais energia, sendo por isso fulcral optimizar o encaminhamento dos mesmos, minimizando, desta forma, o consumo das baterias. Existem várias pesquisas em curso cujo objectivo de estudo é encontrar qual o melhor protocolo para ser empregue nas RSSF.

As RSSF podem ser aplicadas em inúmeras situações. A grande diversidade de sensores proporciona a capacidade de detecção de diversos fenómenos físicos, tais como: temperatura, humidade, pressão, aceleração, vento, luminosidade, infra-vermelhos, campos magnéticos, radiação, composição química, *stress* mecânico, entre outros. As aplicações das RSSF podem ser divididas nas seguintes áreas: militar, industrial, saúde, monitorização doméstica e monitorização ambiental, sendo a última, a aplicação em estudo neste trabalho.

1.2 Histórico das pesquisas sobre Redes de Sensores sem Fios

A história das pesquisas sobre RSSF pode ser dividida em quatro fases. Estas são mencionadas nos seguintes parágrafos.

1 Fase: Durante a Guerra Fria (década de 50) - A maioria das tecnologias, associadas à monitorização remota, iniciaram-se com aplicações em defesa militar. Durante a Guerra Fria eram usados sistemas de detecção de sons (SOSUS- *Sound Surveillance System*) que já utilizavam os conceitos presentes, nos nossos dias, nas RSSF. Este sistema consistia em colocar no fundo do oceano uma vasta quantidade de sensores acústicos dispersados estrategicamente de modo a serem detectados submarinos soviéticos[1]. Actualmente, o SOSUS é utilizado pela NOAA (*National Oceanographic and Atmospheric Administration*), desde 1994 na monitorização de eventos no oceano, tais como actividade sísmica e animal.

Ainda durante a Guerra Fria foram utilizados sistemas de defesa aeroespaciais, em que eram usados balões e aviões através do sistema AWACS (*Airborne Warning and Control System*), os quais funcionavam como sensores.

2 Fase: **Inicio dos Projectos Avançados de Defesa** - Por volta de 1980 iniciaram-se as primeiras pesquisas modernas sobre RSSF, quando a DARPA (*Defense Advanced Research Projects Agency*) iniciou com o projecto DSN (*Distributed Sensor Networks*). O projecto foi concebido para que os nós sensores fossem colocados numa arquitectura distribuída, cooperativa, autónoma e com um baixo custo, tendo como objectivo encaminhar informação entre os diversos nós da melhor forma possível. A rede era formada por sensores acústicos, protocolos de alto nível, técnicas de processamento, algoritmos e por softwares distribuídos[1]. De salientar que nessa altura ainda não existiam computadores pessoais, nem estações de trabalho, pelo que todo o processamento era feito em minicomputadores como o PDP-11 e em máquinas VAX onde eram usados sistemas operativos como o *Unix* e o *OpenVMS* (*Open Virtual Memory System*). Os modems atingiam taxas de operação que variavam entre 300 e 9600 bps[1]. O projecto DSN foi demonstrado utilizando um avião voando em baixa altitude, que era detectado com grande sucesso com os sensores acústicos juntamente com câmaras de TV. A Figura 1.1 mostra a distribuição dos nós sensores e o equipamento utilizado.

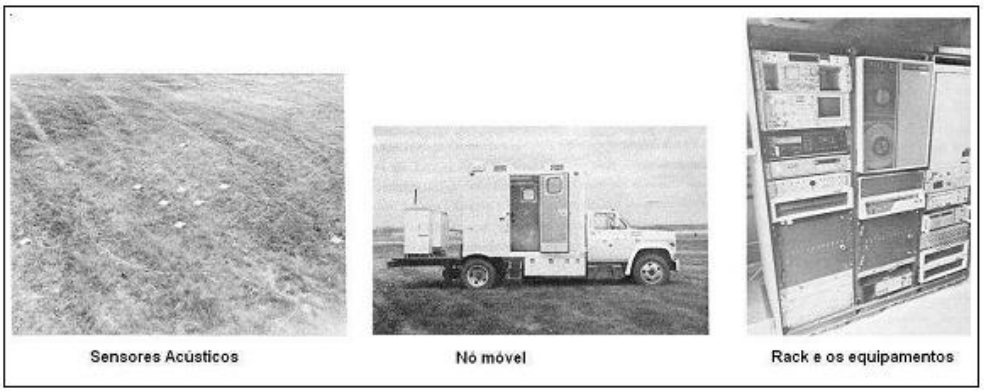


Figura 1.1– Vários componentes usados nos testes do projecto DSN-DARPA.

3 Fase: Aplicações Militares Desenvolvidas entre 1980s e 1990s - Os investigadores em rede de sensores tinham como objectivo criar uma rede com imensos pequenos sensores, mas a tecnologia não o permitia. Em simultâneo, o pessoal envolvido em sistemas militares reconheceu o benefício deste tipo de rede, para aplicação militar num ambiente de guerra, onde os sensores poderiam informar com uma resposta bastante rápida, a posição das tropas inimigas. Um exemplo deste tipo de rede desenvolvida pelo sistema de defesa americano é a CEC (*Cooperative Engagement Capability*)[2]. Um outro sistema militar utilizando redes de sensores acústicas para prevenção de ataques por submarino nos Estados Unidos da América é a FDS (*Fixed Distributed System*)[3]. Em 1999, o DARPA iniciou o projecto SensIT (*Sensor Information Technology*) com duas finalidades principais: desenvolver novas técnicas adequadas para redes *Ad Hoc* em ambientes dinâmicos e realizar processamento de informação na rede (extracção de informação dos sensores de forma optimizada e confiável)[4].

.4 Fase: Actualmente - O avanço tecnológico na área de computação e de comunicação, tornou possível a ideia inicial, das redes de sensores sem fios. Estas redes constituídas por sensores ínfimos e baratos, processadores de baixa potência baseados em sistemas micro-eletro-mecânicos (MEMS), fazem com que aplicações baseadas nestes sensores se tornem mais precisas. É importante salientar que a tecnologia para projectar e construir RSSF está comercialmente disponível e tende a se tornar cada vez mais acessível com a produção em larga escala de diferentes tipos de micro-sensores (Motes, JPL, AMPS, Dust, Pico, WINS, Millennial Net, FireFly, Z1 Mote, WaspMote, TinyNode584, Ubi-Mote2). A Tabela 1.1 apresenta uma comparação entre as características dos sensores existentes no mercado ao longo das gerações[5].

Tabela 1.1 - Características básicas dos nós sensores

	Primeira Geração 1980s a 1990s	Segunda Geração 2000s a 2008	Terceira Geração 2009
Tamanho	Caixa de sapatos	Caixa de cartas, moeda de um euro	Partícula de pó
Peso	Quilogramas	Gramas	Insignificante
Arquitectura	Detecção do meio, processamento e comunicação separadamente	Detecção do meio, processamento e comunicação integrados	Detecção de meio, processamento e comunicação completamente integrados
Protocolos	Proprietário	Proprietário, Wi-Fi, *ZigBee*, WiMax, etc.	Wi-Fi, *ZigBee*, WiMax, etc.
Topologia	Ponto-a-ponto, estrela ou multiponto	Servidor-Cliente, ponto-a-ponto	Ponto-a-ponto
Fonte de alimentação	Baterias grandes ou alimentação externa	Baterias tipo AA e AAA	Solar, eólica, outras baseadas em nano tecnologia.
Durabilidade	Horas, Dias	Dias, Semanas	Semanas a Anos

1.3 Características e factores determinantes nas RSSF

As RSSF podem ser vistas como um tipo especial de rede *Ad Hoc* (MANET - *Mobile Ad hoc Network*). São também conhecidas como WASNs (*Wireless Ad hoc Sensor Network*), WSN (*Wireless Sensor Networks*) ou simplesmente *Sensor Networks*[6].

Apesar das semelhanças com as redes *Ad Hoc*, muitos dos algoritmos utilizados nestas redes não são adequados às redes de sensores sem fios, devido às seguintes particularidades: a escalabilidade de uma rede de sensores normalmente atinge uma ordem de grandeza maior que a das redes convencionais e *Ad Hoc*; os nós de uma rede *Ad Hoc* são tipicamente móveis enquanto que em RSSF são, na maioria das vezes, estacionários; a mudança de topologia das redes *Ad hoc* ocorre normalmente devido à mobilidade dos nós, enquanto que nas RSSF a variabilidade da topologia é dada principalmente pelo recurso limitado de energia (bateria); também são diferentes do ponto de vista da comunicação, na medida em que as redes de sensores frequentemente utilizam *broadcast*, enquanto as redes *Ad Hoc* convencionais utilizam comunicação ponto-a-ponto[6].

As redes de sensores podem ser classificadas de acordo com a sua composição, organização, mobilidade, densidade, aquisição de dados, capacidade limitada de energia e segurança:

- Composição – Uma rede de sensores é denominada **homogénea** quando todos os nós sensores participantes da rede são iguais no que concerne às suas características de *hardware* (processador, memória, bateria, comunicador e dispositivo sensor). Caso contrário é denominada **heterogénea**.
- Organização - Uma RSSF pode ser caracterizada como **plana**, quando todos os nós sensores são pares entre si e tipicamente desempenham o mesmo papel, ou **hierárquica**, quando existem agrupamentos em um ou mais níveis de nós, que desempenham papéis diferentes.
- Mobilidade – Uma RSSF pode ser classificada como **estacionária** ou **móvel**. Numa RSSF estacionária, depois dos nós estarem dispostos na região de monitorização, a sua posição não se modifica. No entanto, isto não garante que a conectividade dos elementos da rede permaneça inalterada, uma vez que alguns nós com o passar do tempo se tornarão indisponíveis devido ao término de suas reservas energéticas ou por danos físicos.
- Densidade – Quanto à densidade, uma RSSF pode ser classificada como **irregular** quando os nós sensores estão espalhados de forma irregular na região de interesse, ou caso contrário, será considerada **balanceada**. Quando a concentração de nós por unidade de área é elevada a rede é denominada **densa**, caso se verifique o contrário será considerada **espaçada**.
- Aquisição de dados – A aquisição de dados pode ocorrer de várias formas: **contínua**, quando os dados são recolhidos continuamente e enviados pela rede; **sobre pedido**, isto é, os dados são enviados quando solicitados; **programada**, quando existe uma programação temporal para o evento a ser monitorizado;

reactiva, os dados são enviados se houver uma alteração no ambiente monitorizado.

- Capacidade limitada de energia – Alguns sensores de uma RSSF podem ser colocados em locais de difícil acesso, ou mesmo em lugares inóspitos. Assim sendo, a única fonte de alimentação serão as baterias. Como a substituição de baterias poderá ser na maior parte dos casos impossível ou mesmo inviável, o consumo de energia torna-se um dos principais parâmetros a ser contabilizado pelos protocolos.
- Segurança – O facto dos nós sensores utilizarem comunicação por radiofrequência compromete a sua segurança, tornado as RSSF mais vulneráveis a ataques do que as redes cabladas.

1.4 Unidades principais de um nó sensor

A Figura 1.2 ilustra, genericamente, os componentes básicos de um nó sensor, o qual é composto por: sensor (es), um microcontrolador, um dispositivo de comunicação (usualmente rádio) e uma bateria.

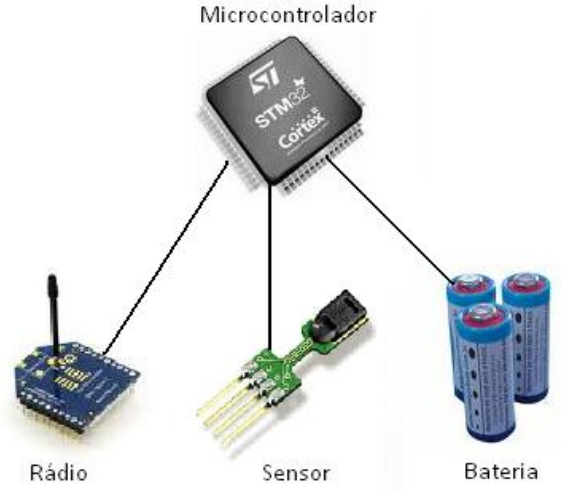


Figura 1.2 - Componentes básicos de um nó sensor.

1.4.1 Sensor

Um sensor é um componente capaz de detectar uma grandeza do mundo físico (temperatura, humidade, pressão, aceleração, luminosidade, etc.) e transformá-la num sinal eléctrico variável (analógico) que representa a amplitude desta grandeza. Muitos materiais possuem características físicas que sofrem modificações com os fenómenos da natureza[6].

1.4.2 Processador (Microcontrolador e Memória)

A maior parte dos microcontroladores utilizados em RSSF contêm memórias *flash* e RAM, conversores analógico-digitais e portas de comunicação, tudo num único *chip* com um baixo consumo de potência e relativamente barato.

Tradicionalmente um microcontrolador opera no intervalo de tensões entre 2,7 a 3,3 V, mas as novas famílias de controladores de baixa potência estão operando abaixo de 1,8 V. A frequência de relógio (*clock*) está a variar na faixa dos 4 MHz aos 400 MHz. Como a frequência e a tensão são determinantes no consumo de potência, os seus valores devem ser os mais baixos possíveis desde que atendam aos requisitos da aplicação.

Quanto à memoria, os nós sensores geralmente necessitam de espaço para armazenar temporariamente os dados do(s) sensor(es) antes de processá-los e transmiti-los. Em geral, para armazenamento do programa é utilizada uma memória *flash* de 1 a 128 KB e para memória RAM de 32 a 128 KB [7].

1.4.3 Bateria (Alimentação)

Em geral são utilizadas baterias para alimentar os nós sensores. As baterias têm uma vida útil finita, pelo que a escolha da bateria a ser aplicada ao nó sensor deve ser considerada em termos de volume, condições de temperatura e capacidade. As baterias mais utilizadas são: linear simples, lítio NR e lítio *Coin Cell*. A energia fornecida aos nós sensores é um dos aspectos mais focados no que refere às RSSF.

1.5 Dispositivo de comunicação (rádio)

Este dispositivo é composto por todo o sistema de transmissão, recepção, amplificação e antena, sendo responsável por interligar os vários nós na rede. Os dois tipos de comunicação mais utilizados nas RSSF são:

1.5.1 Comunicação por rádio frequência (RF)

Nas RSSF as frequências licenciadas de acordo com a norma IEEE 802.15.4 possuem as seguintes variações: 868 MHz, 902-928 MHz, 2,4 GHz. Um outro factor a ser levado em consideração numa transmissão por RF diz respeito às perdas de sinal. Estas perdas podem atingir elevados níveis que variam conforme a distância entre os nós sensores e possíveis obstáculos entre eles. Para além da potência de transmissão, existem outros factores que influenciam o alcance de transmissão tais como: a sensibilidade do receptor, o ganho e a eficiência da antena.

1.5.2 Comunicação por Laser (óptica)

Alguns projectos utilizam a luz (laser ou infra-vermelho) como forma de comunicação. Uma vantagem do uso deste meio de comunicação é a economia no consumo de energia, tendo como desvantagem a necessidade da comunicação ser direccional, ou seja, o transmissor deve ter linha de vista com o receptor [7]. Na Figura 1.3 pode-se observar o módulo de comunicação por laser que forma o nó sensor do *Smart Dust*.

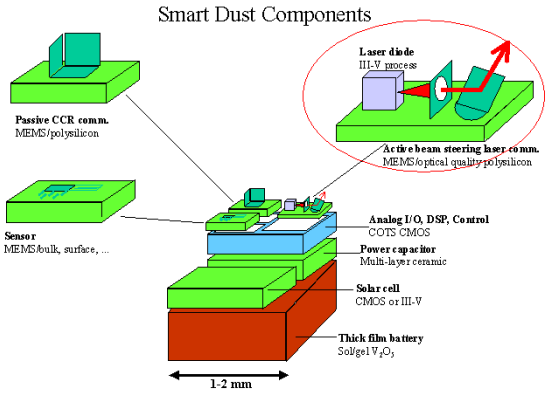


Figura 1.3 - Componentes do nó sensor Smart Dust.

1.6 Projectos de nós sensores sem fio

Existem vários projectos desenvolvidos por pesquisadores de várias Universidades, sendo uma das pioneiras nesta área a Universidade da Califórnia, Berkeley. O primeiro nó sensor foi desenvolvido por Seth Hollar em 2000 o qual denominou-o por Cots Motes. A primeira geração de Motes foi desenvolvida a partir de expansões do projecto original, sendo constituída por: WeC Motes, RF Motes, Laser Motes, Mini Motes, MALT Motes e IrDa Motes. A segunda geração dos nós sensores Motes é composta pelo Mica Mote que, em comparação com os antigos nós sensores, já possuiu uma melhoria na capacidade de comunicação e uma melhor capacidade de optimização da bateria.

A terceira geração é constituída pelos nós *Mica 2*, o *Mica2Dot* e o *MicaZ*, os quais são comercializados pela empresa *Crossbow Technology, Inc*. [8]. Estes nós sensores apresentam vantagem em termos de alcance de rádio, na capacidade de armazenamento e na utilização de vários tipos de sensores.

A quarta geração é caracterizada pelos nós Intel Motes, cujo direito de fabricação está licenciado para a Intel Inc. e o Spec Motes sendo um resultado do projecto *Smart Dust*, em que o objectivo é tornar um nó sensor de dimensões ínfimas.

Existem, no entanto outros projectos tais como: Projecto µAMPS (*Micro-Adaptative Multi-domain Power Aware Sensors*), o Projecto WINS (*Wireless Integrated Network Sensors*) da Universidade da Califórnia em *Los Angeles*, o Projecto Sensor Web 1, 2 e 3, da *Jet Propulsion Lab (JPL)* do Instituto Tecnológico da Califórnia, o Projecto BEAN (*Brazilian Energy-Efficient Architectural Node*), o projecto Medusa MK-2, o projecto MIllennialNet, e o projecto mais recente da empresa *Crossbow*, o nó sensor *Eko Sensor* concebido especificamente para monitorização ambiental [9].

A Figura 1.4 apresenta alguns exemplos de nós sensores sem fios mencionados anteriormente.

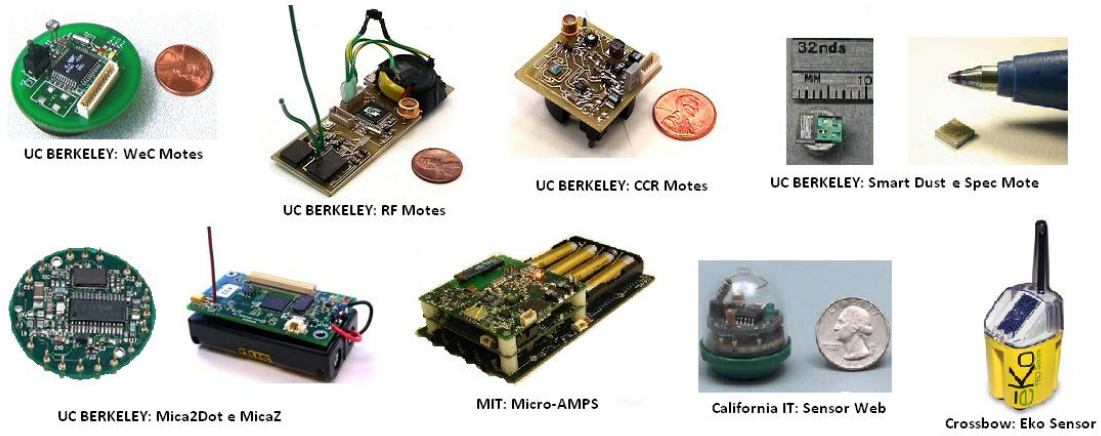


Figura 1.4 - Vários nós sensores existentes.

O *Eko Sensor* é um sensor totalmente integrado, concebido para o exterior em que já tem aplicado um pequeno painel solar para recarregar as baterias. Este nó tem uma capacidade de alcance de 3,2 km dependendo do ambiente e da configuração do *hardware*. Os nós formam uma rede sem fios em malha que pode ser utilizada para expandir a área de cobertura. Cada nó pode acomodar 4 tipos de sensores e ainda tem a facilidade de escolher outros sensores que não façam parte da *Crossbow*. O nó sensor é constituído por uma placa que contém um processador/rádio, uma antena e é alimentado por baterias recarregáveis, carregadas por um painel solar. Na Tabela 1.2 constam as especificações de dois nós EKO da *Crossbow*.

Tabela 1.2 - Especificações dos nós EKO EN2100 e EN2120**[10]**

Nó EKO	**EN2100**	**EN2120**
Sensores		
Número de portas	Cada nó suporta 4 sensores	
Intervalo de medição	Uma medição a cada 15 minutos	
Rádio		
Frequência	2,405 a 2,480 GHz	
Canais	16 canais disponíveis seleccionáveis por *switch* de rotação	
Modulação	DSSS, IEEE 802.15.4	
Potência de saída	+ 3 dBm	+18 dBm para USA; +10 dBm para a Europa
Sensibilidade de recepção	-101 dBm	
Alcance no exterior	de 152 a 457 metros em linha de vista por salto	De 0.609 a 3.2 km em linha de vista por salto
Antena	Dipolo, interna	
Alimentação		
Corrente de operação	0,4 mA de média a cada 15 minutos de amostragem (sem sensores)	0,5 mA de média a cada 15 minutos de amostragem (sem sensores)
Painel Solar	3.3cm x 6.25 cm	
Baterias	Standard: 3AA NiMh recarregáveis Expectativa de vida: 3 meses sem luz solar. Mais do que 5 anos em campo	

1.7 Aplicações de monitorização ambiental

Actualmente, os incêndios queimam por ano uma grande quantidade de área florestal. Este facto ocorre principalmente no Verão, devido às elevadas temperaturas e baixo índice de humidade. Existem soluções para minimizar a probabilidade de incêndio. São usados dois métodos tradicionais, em que o primeiro consiste em vigilância aérea, com o uso de helicópteros ou avionetas, e o segundo método baseia-se em pontos de observação, em locais estratégicos, normalmente no cimo de morros ou colinas [11]. Uma alternativa a estes métodos é o uso de redes de sensores sem fios. Os sensores podem ser colocados manualmente ou, quando a região é de muito difícil acesso, poderão ser utilizados aviões ou helicópteros para a distribuição dos mesmos.

1.7.1 Sistema FFSS

Nas montanhas do Correia do Sul foi aplicado um sistema de prevenção contra incêndios florestais em tempo real designado por FFSS (*Forest-Fire Surveillance System*) em que na possibilidade da ocorrência de um incêndio o sistema emite um alerta [12]. O FFSS usa um nó sensor TIP50CM visível na Figura 1.5, produzido pela *Maxfor co. Ltd* no ano de 2002.

Figura 1.5 - Nó sensor TIP50CM [13].

O TIP50CM é um módulo que vem incorporado com sensores de luminosidade, temperatura e humidade e fornece uma ligação flexível com periféricos[12]. Este módulo pode ser aplicado a uma larga gama de aplicações que utilizam redes em malha. As especificações deste módulo são descritas na Tabela 1.3.

Tabela 1.3 - Especificações do nó TIP50CM.

Item	Descrição
Processador	16 Bit RISC, 8 MHz
Memória	256 KB Programação Flash
Sistema Operativo	TinyOs
Rádio Multi-canal	2,4 GHz
Taxa de transmissão	250 Kbyte
Sensores	Temperatura, Humidade e Luminosidade
Rede	Multi-salto e *Ad hoc*

Interface	USB (UART)
Tamanho	68x29 mm
Alimentação	3,0 ~3,3 V
Alcance	70 m em Laboratório, Máximo: 155 m

O sistema FFSS monitoriza parâmetros ambientais da floresta e determina o nível de risco de incêndio através de uma fórmula específica para o efeito. Na Figura 1.6 pode-se visualizar os componentes do sistema FFSS, os quais fazem parte os nós sensores, um computador para recepção dos dados, sendo utilizada uma aplicação informática para colocar os dados numa base de dados e *on-line*.

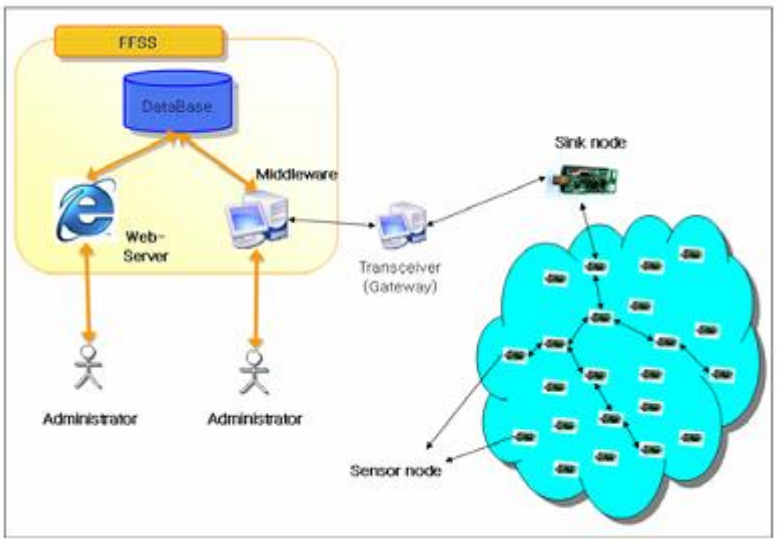


Figura 1.6 - Estrutura do sistema FFSS.

O *Sink node* é o nó que recebe os vários pacotes de dados fornecidos pelos nós sensores e reencaminha-os para a *Gateway*, a qual verifica e analisa os dados de interesse e envia-os para uma aplicação informática (Middleware) que mostra num ecrã os dados monitorizados, ou seja, temperatura, humidade e luminosidade. Esta mesma aplicação não só serve para colocar os dados numa base de dados para uma futura análise estatística, mas como também serve para colocar os dados numa aplicação Web, possível de ser acedida em qualquer parte do mundo. Na ocorrência de um incêndio, a aplicação dispara um alarme e o *software* foi concebido para mostrar ao administrador as várias opções de ajuda que deseja contactar para a extinção do incêndio florestal como mostra a Figura 1.7.

Figura 1.7 - Serviços de apoio à extinção de incêndios.

1.7.2 Sistema FireBug

Monitorizar dados em tempo real é importante para minimizar os prejuízos efectuados por incêndios nas florestas. O sistema *FireBug* foi concebido para monitorizar fogos incontroláveis utilizando nós sem fios com sensores de temperatura, humidade, luminosidade e um módulo de localização (GPS). Os sensores são agrupados numa placa de circuito impresso a qual é ligada a um nó Mica2. A Figura 1.8 mostra a placa em questão (MTS420CA) juntamente com o nó Mica2.

Figura 1.8 - Placa de circuito MTS420CA e nó MICA2.

A arquitectura do sistema *Firebug* visível na Figura 1.9 é semelhante ao do sistema FFSS. Da mesma forma, os dados após serem enviados para o computador são armazenados numa base de dados (Servers), os quais podem ser consultados por uma aplicação Web (Clientes)[14].

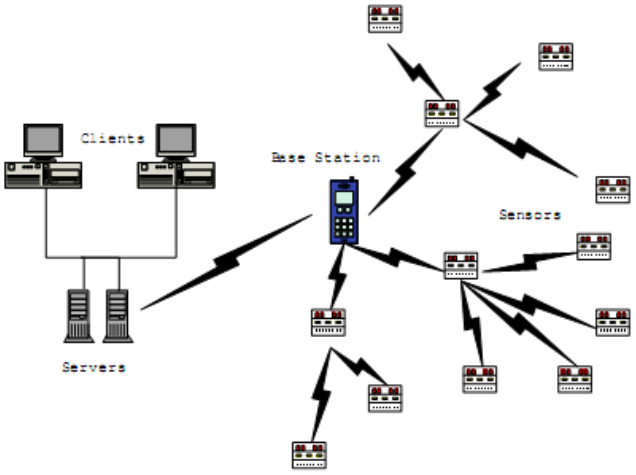


Figura 1.9 - Arquitectura do sistema.

Os testes efectuados decorreram num campo aberto em *Pinole Point Regional Park* na Califórnia [15], onde foram colocados os vários nós ao longo do campo e afixados num mastro de madeira como se pode visualizar na Figura 1.10.

Figura 1.10 - Colocação dos vários nós e sua fixação.

Neste sistema como temos o módulo GPS (Leadtek 9546) incorporado é possível detectar o local onde ocorre o incêndio com uma precisão num raio de 5 m.

1.7.3 Monitorização de habitats

A monitorização de habitats representa uma classe de aplicação das RSSF com enorme benefício para a comunidade científica e para a sociedade em geral [11]. Os investigadores estudaram durante muitos anos a interferência humana em determinadas regiões. Em algumas regiões, os investigadores têm de permanecer em contacto com a fauna ou a flora, situação predominante em ilhas isoladas, onde várias espécies se isolam para procriar, nomeadamente as aves.

De acordo com pesquisas feitas por *Mainwaring* [16], a presença de humanos por uma duração de 15 minutos próximos dos ninhos acarreta em 20% de mortalidade das aves recém-nascidas[16]. De modo a minimizar a mortalidade das aves, os sensores deverão ser colocados num período fora da época de acasalamento.

Um estudo realizado em 2002 consistiu na colocação de 150 sensores espalhados na ilha de *Great Duck* (na costa de Maine, USA), com o propósito de monitorizar as

temperaturas próximas e em alguns casos no interior das tocas das aves como mostra o biólogo na Figura 1.11

Figura 1.11 - Colocação de sensores nas tocas das aves.

A Figura 1.12 não só ilustra a colocação dos sensores na ilha como também ilustra o funcionamento da transmissão de dados. No ponto 1, tem-se os sensores dentro das tocas dos petréis e no ponto 2 à saída das tocas. O ponto 3 serve de gateway, o qual envia os dados recebidos para um computador na estação base definida como ponto 4. Os dados recebidos serão encaminhados por uma parabólica (ponto 5) para a estação terminal na Califórnia.

Figura 1.12 - Esquemático da RSSF na ilha *Great Duck*[17].

1.7.4 TUTWSN

Foi desenvolvida pela *Tampere University of Technoly* um protótipo para a elaboração de uma rede de sensores sem fio para monitorização ambiental.

O protótipo é controlado pelo microcontrolador MicrochipPIC18LF4620, um rádio *Nordic Semiconductor* nRF905 operando na frequência dos 433 MHz com uma sensibilidade de -100 dBm e uma potência de transmissão ajustável entre -10 a 10 dBm. O rádio é conectado a um dipolo dobrado numa placa de circuito impresso. O sensor de temperatura utilizado é o *Dallas Semiconductor* DS620 o que tem uma precisão de ± 0.5 ºC desde 0 a 70ºC e opera a uma gama de temperaturas entre -55 a 125 ºC. Como

fonte de alimentação é utilizada uma bateria de lítio (CR123A) com 3V/1600mAh. O circuito opera a uma tensão de 2,25V necessitando do regulador de tensão Max1725[18]. O consumo energético mínimo (potência) deste protótipo quando todos os componentes estão no estado "dormindo " é de 31 μW para uma tensão de 3V. No estado activo o consumo de energia pode variar dependendo da potência de transmissão do rádio. Para uma potência de transmissão de -10 dBm obtém-se 70 mW e para +10dBm dá 133.28 mW.

Na Figura 1.13 é possível visualizar a arquitectura do hardware desenvolvido.

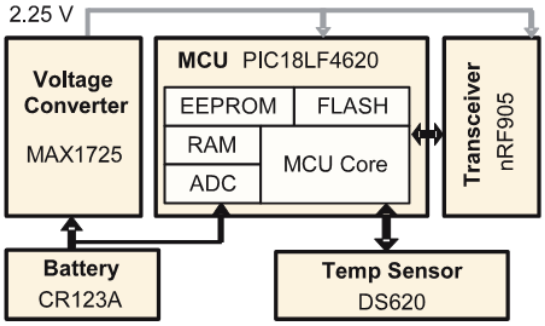


Figura 1.13 – Arquitectura do hardware do TUTWSN[18].

Em termos de dimensões o protótipo tem 255 mm x 21 mm o que lhe confere uma estrutura alongada para uma melhor fixação num ponto de monitorização. A Figura 1.14 mostra o protótipo em questão.

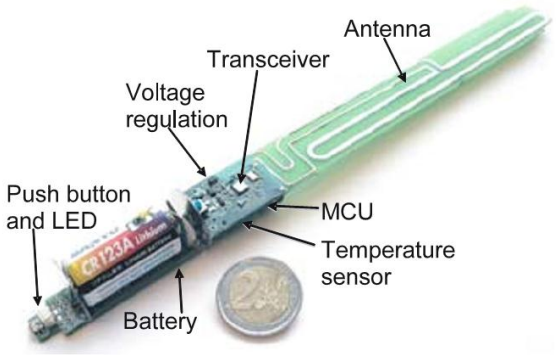


Figura 1.14 – Protótipo TUTWSN

Foram colocados 19 nós num ambiente exterior cobrindo uma área de 2 Km^2. Os nós podem não estar em linha de vista e são colocados a 1 m do solo normalmente presos a uma árvore como mostra a.

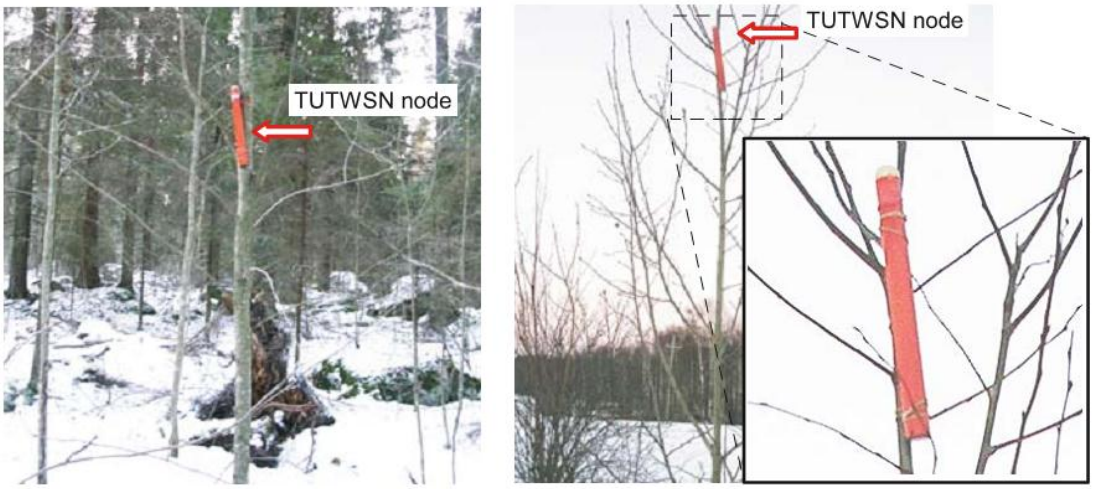


Figura 1.15 – Protótipo TUTWSN envolvido numa protecção à prova de água fixo numa árvore

É mencionado ainda em [18] que de acordo com os dados obtidos a partir dos valores das baterias é esperado que a duração da rede seja de 6 meses gerando 4 pacotes por minuto.

2 Relação entre *ZigBee* e o padrão IEEE 802.15.4

Neste capítulo introduzem-se os conceitos sobre as tecnologias sem fios existentes actualmente, salientado a tecnologia *ZigBee*. Será mencionada a pilha protocolar, as especificações da camada física e da camada MAC, os tipos de dispositivos no que diz respeito às funções desempenhadas e os modos de operação. Finaliza-se com uma tabela onde estão indicados rádios e módulos existentes no mercado que suportam o protocolo *ZigBee*.

2.1 Introdução

O desenvolvimento da tecnologia possibilitou o aparecimento de várias alternativas e protocolos, embora inicialmente tenha sido colocada ênfase principalmente na transmissão de dados e voz, com elevadas taxas de transmissão. Estes equipamentos tinham um elevado preço, resultando, desta forma, num desinteresse para aplicações mais simples.

Só após os protocolos para suporte de comunicações sem fios, de médio ou alto débito, como o *Bluetooth* ou o *Wi-Fi* estarem desenvolvidos é que se começou a estruturar um protocolo que respondesse às necessidades específicas para aplicações com sensores.

O Grupo de Trabalho 802.15 definiu três classes de redes WPANs (*Wireless Personal Area Network*) que são diferenciadas pela taxa de transferência, escoamento da bateria (descarregar) e QoS (*Quality of Service*). A classe com maior taxa de transferência, o IEEE Std 802.15.3 ™, é adequada para aplicações multimédia as quais requerem uma QoS elevada. A classe com uma transferência de dados média, aIEEE 802.15.1™/Bluetooth, foi projectada para remover quaisquer cabos existentes na utilização de dispositivos electrónicos tais como o telemóvel ou um PDA. A última classe designada por LR-WPAN (*Low Rate Wireless Personal Area Network)* é projectada para servir aplicações com consumo energético inferior às restantes classes e com uma reduzida taxa de transferência e baixa complexidade [19]. A Figura 2.1 ilustra o espaço de operação dos padrões 802 WLAN e WPAN.

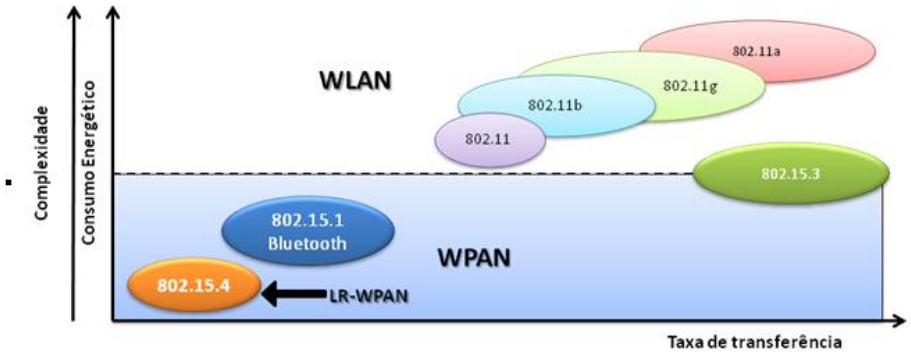


Figura 2.1 - Espaço de operação para os padrões 802 WLAN e WPAN.

A Tabela 2.1 apresenta um sumário com as principais características de uma LR-WPAN usando o padrão IEEE 802.15.4 em comparação com IEEE 802.15.1 e IEEE 802.11g.

Tabela 2.1 - Comparação da LR-WPAN com outras tecnologias.

	IEEE 802.15.4	IEEE 802.15.1	IEEE 802.11g
Débito	200 Kbps	1 Mbps	54 Mbps
Consumos	Tx > 30 mA Standby: <**10 µA**	Tx > 400 mA Standby: **0,20 mA**	Tx > 400 mA Standby: **20 mA**
Pilha Protocolar	≈32Kb	≈250Kb	≈1 Mb
Vantagens	Consumo, latência, número de nós	Interoperabilidade; ausência de cabos	Elevada taxa de transferência
Principais aplicações	Monitorização e controlo; sensores	Periféricos de PCs, telemóvel, PDAs	Internet; transferência de arquivos, vídeo/áudio
Alcance de transmissão (metros)	1-100+	1-10+	1-100+

Conforme se constata, o padrão IEEE 802.15.4 apresenta um consumo muito reduzido em comparação com o padrão IEEE 802.15.1 e IEEE 802.11g, especialmente nos períodos de inactividade sendo uma mais-valia deste tipo de tecnologia. Convém ainda mencionar alguns dos pontos mais atractivos[20]:

- *duty cycle* muito baixo e suporte para dispositivos de funções reduzidas (minimizando o consumo), possibilitando elevada autonomia quando alimentado por baterias;
- suporte a topologias de rede estáticas e dinâmicas, quer em estrela quer em malha;
- capacidade para permanecer longos períodos sem comunicação;
- permite a utilização de redes com mais de 65.000 nós, procurando garantir sempre baixa latência;
- uso de *Direct Sequence Spread Spectrum* o que permite que os dispositivos permaneçam em "*sleep-mode*" sem necessidade exigente de sincronização.

2.2 Pilha Protocolar

O Grupo de Trabalho 802.15.4 redigiu o padrão IEEE 802.15.4 em que define apenas as duas camadas básicas provenientes da ISO-OSI (*International Organization for Standardization - Open Systems Interconnection)* que são a camada física e a camada de dados. As restantes camadas não são especificadas neste padrão e normalmente são especificadas por consórcios industriais formadas por companhias interessadas no fabrico e uso particular de um padrão[19].

2.2.1 *ZigBee*

A *ZigBee Alliance*, sendo uma aliança constituída actualmente por mais de 200 empresas provindas de mais de 20 países, nas quais integra especialistas na área de telecomunicações e semicondutores, dedicou-se ao desenvolvimento das camadas superiores do padrão IEEE 802.15.4. A Figura 2.2 ilustra a pilha protocolar referente ao padrão IEEE 802.15.4/*ZigBee*.

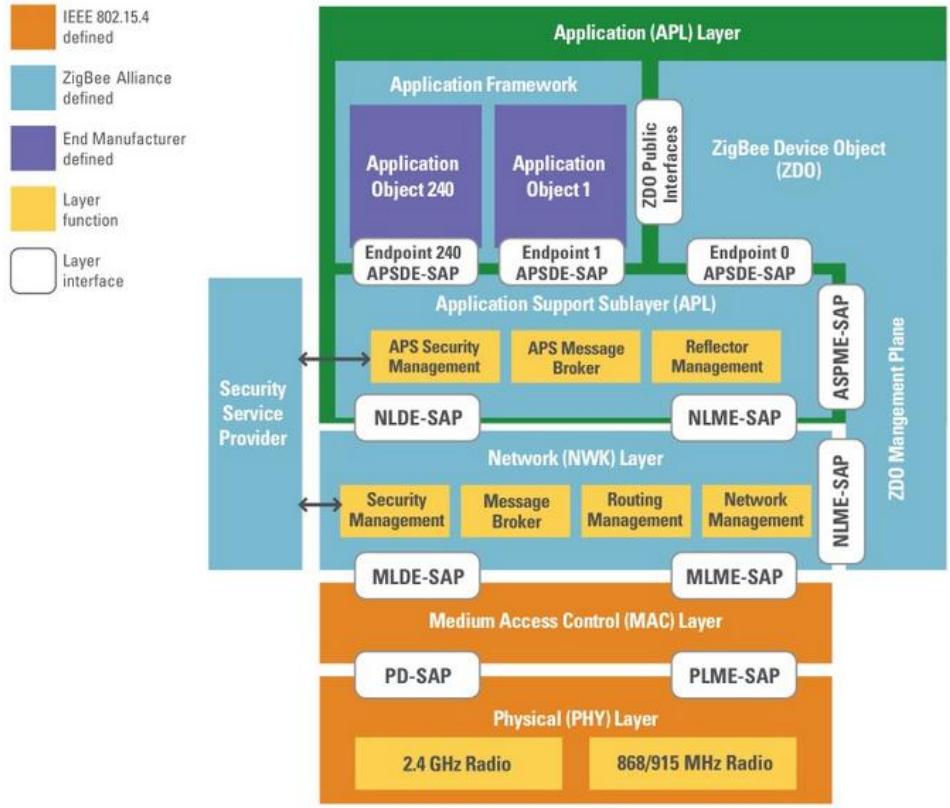


Figura 2.2 – Pilha protocolar do padrão IEEE 802.15.4[21].

Cada camada da pilha comunica com as camadas adjacentes através de um serviço designado por ponto de acesso (SAPs – *Service Access Point*). Esta é uma localização conceptual onde uma camada protocolar pode requisitar serviços das camadas adjacentes.

A camada física (PHY) específica os componentes de interface de rede, os seus parâmetros e modo de operação. Para suportar a operação da camada de controlo de acesso ao meio (MAC), a camada física inclui uma serie de funcionalidades tais como detecção de energia recebida (RED), indicador de qualidade do sinal (LQI) e verificação de canal livre (CCA)[5].

A camada MAC, como o próprio nome indica controla o acesso ao meio e pode ser efectuado por dois modos de operação: *beacon* e *non-beacon*[22].

A camada de rede (NWK) provisiona as funcionalidades necessárias para suportar a configuração da rede, descoberta de novos dispositivos, associação e desassociação, manutenção da topologia, gestão de camada MAC, encaminhamento e gestão de segurança. São suportadas três tipos de topologias: estrela, malha e árvore.

A camada de segurança aproveita os serviços básicos especificados pelo padrão IEEE 802.15.4 para fornecer suporte à infra-estrutura de segurança e aos dados da aplicação. É baseada num algoritmo simétrico de encriptação, o 128-AES que utiliza uma chave de 128 bits para encriptar a mensagem. Por defeito, a segurança não está activa, sendo possível de activar através da camada de aplicação.

A camada de aplicação consiste na subcamada de suporte de aplicação (APS), na *ZigBee device object* (ZDO) e nos objectos de aplicação definidos pelos fabricantes. A responsabilidade da APS inclui a manutenção das tabelas de encaminhamento para manter os vários dispositivos interligados de acordo com os seus serviços e necessidades, e reencaminhar as mensagens entre os dispositivos.

O ZDO pode ser visto como um objecto de aplicação em todos os nós. O ZDO tem o seu próprio perfil referido como *ZigBee device profile*. É ainda responsável pela manutenção dos dispositivos de uma forma global, incluindo o papel de um dispositivo na rede (coordenador, *router* ou *end-device*), e por estabelecer uma relação segura entre os vários dispositivos.

2.2.2 Especificações da camada física (PHY)

Esta camada executa essencialmente o papel de controlo da transmissão e recepção de dados através do meio físico, neste caso em concreto o ar. O padrão IEEE 802.15.4 atribui três frequências de operação para utilização nas LR-WPAN: 868 MHz, 915 MHz e 2,4 GHz.

A Tabela 2.2 mostra um comparativo entre as três bandas de operação.

Tabela 2.2 - Tabela comparativa entre as três frequências de operação.

Frequências (MHz)	Número de canais	Taxa de transferência (kbps)	Modulação	Isenção de licenciamento
868	1	20-100	BPSK	Europa
902-928	10	40-250	BPSK	América Austrália
2400-2483	16	250	O-QPSK	Mundial

O *ZigBee* opera em três frequências de rádio conhecidas como ISM (*Industrial, Scientific and Medical*), as quais estão isentas de licenciamento. Em termos mundiais opera na banda dos 2,4 GHz; na banda dos 915 MHz encontram-se os Estados Unidos da América e Austrália e na banda dos 868 MHz a Europa. Consoante a banda utilizada, a taxa de transmissão é variável: nos 2, 4 GHz podem ser obtidas taxas de transmissão de 250 kbps, com 16 canais disponíveis; na banda dos 915 MHz, tem-se uma taxa de transmissão que pode ir desde 40 a 250 kbps e utiliza 10 canais de comunicação; no caso de 868 MHz, possibilita 1 canal e uma taxa de transmissão de 20 a 100 kbps. Em termos de modulação, é utilizado O-QPSK para a banda dos 2,4 GHz e BPSK para os 868 e 915 MHz[23]. As frequências centrais dos canais são definidas da seguinte forma[23]:

$$F_c = 868{,}3\ MHz, \qquad para\ K = 0 \tag{2.1}$$

$$F_c = 906 + 2(k-1)MHz, \qquad para\ K = 1{,}2, \ldots, 10 \tag{2.2}$$

$$F_c = 2405 + 5(k-1)MHz, \qquad para\ K = 11{,}12, \ldots, 26 \tag{2.3}$$

em que K é o numero do canal

Na Figura 2.3 podem-se observar as três frequências mencionadas com os seus respectivos canais.

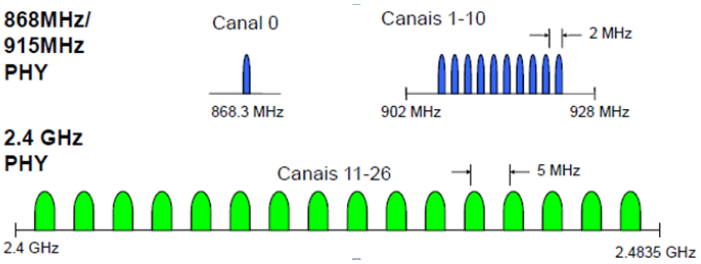


Figura 2.3 - Características de cada banda de comunicação.

2.2.3 Especificações de camada de acesso ao meio (MAC)

A camada MAC oferece a interface entre a camada física e a camada de rede definida pela *ZigBee Alliance.* Esta camada tem como principal funcionalidade controlar o acesso aos canais de radiofrequência, sendo para isso necessário apresentar mecanismos de prevenção de colisão tais como o CSMA-CA *(Carrier Sense Multiple Access – Collision Avoidance*)[23].

A camada MAC também é encarregue de executar as seguintes funções[23]:

- Gerar *beacons*, se o dispositivo é um coordenador;
- Sincronizar o dispositivo na rede *beacon*;
- Usar o CSMA-CA para aceder ao canal;
- Gerir canais de acesso GTS (*Guaranteed Time Slot*);
- Fornecer uma ligação confiável entre duas entidades MAC iguais de dois dispositivos diferentes;
- Fornecer serviços de associação e dissociação PAN;
- Fornecer suporte para segurança; a camada MAC é responsável pelo seu processamento seguro, mas as camadas superiores determinam qual é o nível de segurança que deverá ser utilizado.

2.2.3.1 Tipos de dispositivos

O grupo de trabalho IEEE 802.15.4 distingue os dispositivos com base no *hardware* e capacidade[19]. De acordo com o padrão existem duas classes de dispositivos: *Full Function Device* (FFD) e *Reduce Function Device* (RFD)[19]. As FFD são dispositivos robustos a nível de *hardware*, com o intuito de suportar uma maior funcionalidade e características suportadas pelo protocolo, assumindo desta forma múltiplas responsabilidades na rede para além de poderem comunicar com outras redes. Têm um consumo energético superior

às RFD, porque os dispositivos lógicos não podem ser colocados em modo de adormecimento, enquanto nas RFD essa funcionalidade é possível. As RFD, em relação às FFD, apresentam uma maior simplicidade a nível de *hardware*, sendo compatíveis com microcontroladores de 8 bits e comunicam apenas com dispositivos físicos FFD.

O *ZigBee* diferencia basicamente três tipos de dispositivos lógicos, que são o coordenador, o *router* e o *end-device*. A Tabela 2.3 apresenta as possíveis combinações entre os vários tipos de dispositivos numa rede *ZigBee*.

Tabela 2.3 - Tipos de dispositivos numa rede *ZigBee*

	Dispositivo lógico		
Dispositivo físico	Coordenador	*Router*	*End-device*
Full Function Device	Sim	Sim	Sim
Reduce Function Device	Não	Não	Sim

O **coordenador** é um dispositivo físico FFD e é o único nó mandatário numa rede *ZigBee*. É responsável pela gestão de actividades na rede tais como:

- seleccionar um canal e uma PAN ID para inicialmente formar uma rede;
- admitir outros nós na rede (*routers* e *end-devices*);
- atribuição de endereços de rede

Como o **coordenador** é um nó que processa muita informação e tem de estar no modo acordado é aconselhável que esteja constantemente ligado a uma fonte de energia estável e confiável[24].

O ***router*** também é um dispositivo físico FFD mas tem como principal função o reencaminhamento da informação (dados). Actua como um dispositivo intermediário para interligar os vários dispositivos lógicos na rede e reencaminhar mensagens através de vários saltos (*multihop*)[5].

O nó **terminal** (*end-device*) pode assumir os dois tipos de dispositivos, sendo o mais comum o RFD. Estes nós comunicam apenas com *routers* ou com o coordenador. Não têm a capacidade de retransmitir mensagens provenientes de outros nós terminais[5]. A Figura 2.4 ilustra um exemplo de uma rede *ZigBee* formada com os três tipos de dispositivos lógicos mencionados.

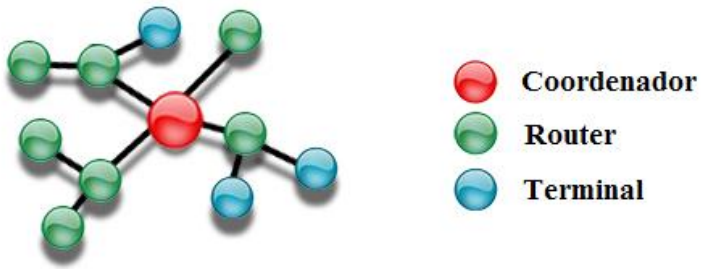


Figura 2.4 - Exemplo de uma rede *ZigBee*.

Baseadas nestes três tipos de dispositivos, uma *ZigBee* PAN pode ser organizada em uma das três topologias seguintes: **estrela**, **árvore** ou **malha** (ponto-a-ponto). Os três tipos de topologias de rede são ilustrados na Figura 2.5.

A fim de transmitir mensagens com dados de um lado para outro na rede, um pacote será reencaminhado várias vezes pelos *routers* intermediários. Este processo requer mecanismos inteligentes de descoberta que dependem da topologia empregue[24].

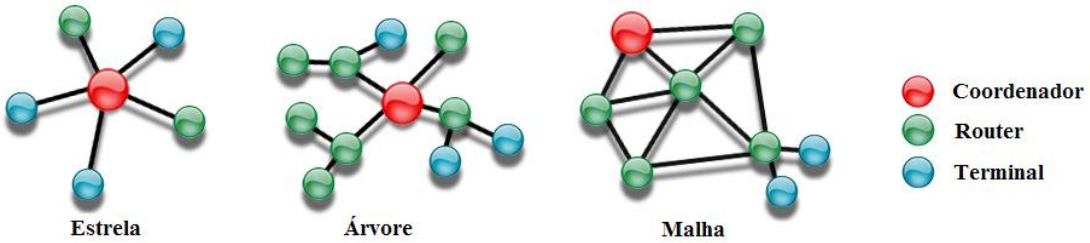


Figura 2.5 - Topologias de rede.

A topologia em **estrela** é a mais simples das topologias numa rede *ZigBee* e contém um único coordenador com até 65.536 terminais. O coordenador é responsável por inicializar e manter os terminais na rede. Após inicialização os terminais só podem comunicar com o coordenador.

Na topologia em **árvore** a relação *parent-child* é directamente utilizada. Cada nó pertencente a nível hierárquico superior é designado por *parent*, pelo que o nó que lhe fica associado é designado por *child.* Este tipo de topologia é adequado para aplicações tolerantes a latência. Como desvantagem, no caso de um nó intermédio falhar, a mensagem não alcançará o destino porque não existe um caminho alternativo (*links*).

De modo a garantir a fiabilidade numa rede *ZigBee*, a topologia **malha** é a mais adequada. A configuração em malha permite a formação de múltiplos caminhos (*links*) entre os nós utilizando o protocolo de encaminhamento *Table-driven*. Este protocolo tem por função fazer com que cada nó mantenha em memória uma ou mais tabelas de encaminhamento referentes aos nós existentes na rede[25].

Como esta topologia apresenta múltiplos caminhos, pelos quais a informação pode fluir, constata-se que é a topologia mais tolerante em caso de haver falhas, sejam provenientes dos nós ou dos caminhos pois apresenta um caminho alternativo para que a informação chegue ao destinatário, neste caso, o coordenador.

2.2.3.2 Modos de operação

O coordenador pode operar a rede com recurso a uma estrutura designada por *super-frame*. No caso de ela estar presente o modo de operação é designado por *beacon*, caso contrário será *non-beacon*

- **Modo *beacon***

Neste modo, o coordenador da rede transmite periodicamente um *frame* que é utilizado pelos dispositivos para a sincronização e determinação do envio e recebimento

das mensagens. Este modo é usado quando o coordenador opera sobre baterias e assim oferece uma maior economia da bateria

- **Modo *non-beacon***

É um modo que requer o coordenador sempre "acordado", fazendo com que haja um maior consumo energético. Neste modo qualquer dispositivo pode comunicar com o coordenador a qualquer momento.

2.3 Estudo de mercado

Existem no mercado várias opções referentes à utilização de rádios e módulos que suportam o protocolo *ZigBee* para formar uma rede de sensores sem fios. Podem ser divididos em três categorias: rádios transceptores *ZigBee*, rádios transceptores com microcontrolador integrado e ainda fornecido em módulos e *kits*. Para a utilização no projecto decidiu-se por escolher a utilização de rádios transceptores com microcontrolador integrado provenientes da *Digi International Inc*. O módulo em questão foi o *Series 2 XBee ZB*. A Tabela 2.4 apresenta a comparação de vários módulos alternativos existentes no mercado, que podiam ser utilizados como uma opção alternativa [26].

Tabela 2.4 – Comparação de módulos existentes no mercado

Fabricante	Módulo	SOC/SIP Chip	MCU core	Flash (kB)	Antena	Corrente "Dormir" (µA)	Corrente Transmissão (mA)	Corrente Recepção (mA)	Potência Emitida (dBm)	Sensibilidade (dBm)	Dimensões (mm x mm xmm)	Data lançamento	Interface	*Firmware*
Digi Making Wireless M2M Easy	Series 1 XBEE	Freescale MC13213	8bit 689S08A	60	Mini monopolo, Chip, U.FL, RPSMA	10, 50	45	50	0	-92	24.38 x 27.61	Dec, 2006	UART	*ZigBee* stack, DigiMesh network protocol
	Series 1 XBee-PRO						250, 340, 180	55	18, 10	-100	24.38 x 32.94			
	Series 2 XBee ZB	Ember EM250	16bit 12MHz RISC	128		1	40, 35	40, 38	3, 1	-96, -95	24.38 x 27.61	Apr, 2008		Ember ZNet, DigiMesh network protocol
	Series 2 XBee-PRO ZB					10	295, 170	45	17, 10	-102	24.38 x 32.94			
Crossbow	IRIS (XM2110CA)	ATmega128L + AT86RF230	8bit ATmega 1281	128	Interna	8	10, 13, 17	16	-17, -3, 3	-101	58 x 32 x 7	Apr, 2007	UART, SPI, I²C, 51-pin expansion connector	*ZigBee* stack Moteworks platform
	MICAz (MPR2400)	Chipcon CC242	8bit ATmega128L		MMCX conector	15	11, 14, 17.4	19.7	-10 -5, 0	-94		Sep, 2005		
STMicroelectronics	SPZB250	SN250	16-bit XAP2b	128	Integrated murata antenna	2	36	36	3	-92	26.5 x 16.4	Aug, 2008	UART, SPI, I²C	Ember Znet
	SPZB260	SN260				1				-95	25 x 13.7		SPI	
ATMEL	ZigBit 2.4GHz (ATZB-24-B0)	AT86RF230	8bit ATmega 1281v	128	Saída RF	< 6	18	19	-17 a 3	- 101	18.8 x 13.5 x 2	2009	UART, USART, SPI, I²C, JTAG	BitCloud – *ZigBee* PRO, Wireless MCU Software, SerialNet, Open MAC
	ZigBit 2.4GHz (ATZB-24-A2)				balanced dual chip antenna						24 x 13.5 x 2			
	ZigBit 2.4GHz Amplified (ATZB-A24-U0)				Saída RF		50	23	> 20	-104	38 x 13.5 x 2			
	ZigBit 2.4GHz Amplified (ATZB-A24-UFL)				Conector U.FL									
	ZigBit 700/800/900MHz (ATZB-900-B0)	AT86RF212			Saída RF		26	11	> 11	-110	24 x 13.5 x 2.8			

Jennic	JN5139-xxx-M00	JN5139	16MHz 32bit RISC	192	Na placa	2.6	37	37	2.5	-99	18 x 30	Junho 2008	UART SPI	JenNet stack *ZigBee* stack
	JN5139-xxx-M01				Conector SMA									
	JN5139-xxx-M03				Conector U.FL									
	JN5139-xxx-M02				Conector SMA		125	45	19	-100	18 x 41			
	JN5139-xxx-M04				Conector U.FL									
Radiocrafts Embedded Wireless Solutions	RC2200	Chipcon CC2420	8bit ATmega 128	128	Integrada, conector MMCX, Saída RF, Conector PIN	23, 1.3	27	30	0	-94	16.5 x 29.2 x 3.5	2008	UART SPI JTAG ISP	Chipcon Z-stack, *ZigBee* stack
	RC2201					1.3								
	RC2202			32			23	26						
	RC2204			64		23, 1.3	27	30						
	RC2300	Chipcon CC2430	8bit 8051	128	Integrada, Saída RF, Conector PIN	300, 0.9, 0.6	27	27	0	-92	12.7 x 25.4 x 2.5	Julho 2008	UART SPI Interface serial proprietária	
	RC2304			64										
Radios.INC	MXR-EM20	Ember 2420	8bit ATmega128L	128	Saída RF	20	17.4	19.7	0	-94	25.4 x 18.4	Outubro 2005	SPI	*ZigBee* Stack
REDWIRE	Redbee	MC13224V	32bit ARM7	120	PCB	1.1	30	25	6	-100	32 x 38 x 2	Julho 2009	UART	Contiki, Freescale Beestack, GCC

3 Concepção arquitectural

Neste capítulo pretende-se presentear ao leitor os requisitos necessários para a elaboração do protótipo. Outro dos aspectos mencionados é a arquitectura de rede sobre a qual o protótipo funciona.

Pretende-se ainda introduzir os conceitos sobre os componentes que formam um nó sensor tais como: baterias, circuitos de estabilização de tensão, rádio XBee, microcontrolador e sensores.

Um dos aspectos fulcrais deste projecto consiste na redução do consumo energético do nó desenvolvendo um algoritmo (técnica de adormecimento dos routers) para o efeito.

O último ponto deste capítulo irá recair sobre a propagação de sinais em ambientes florestais, onde serão mencionados alguns dos modelos de propagação em meios florestais mais utilizados para o cálculo da atenuação do sinal.

3.1 Requisitos para os dispositivos a desenvolver

Pretende-se desenvolver um módulo de comunicação sem fios (emissor/receptor) acoplado a determinados sensores existentes no mercado. É relevante que as características físicas deste módulo permitam serem integradas num ambiente sujeito a condições atmosféricas adversas, pelo que os módulos deverão caracterizar-se por:

- terem um reduzido consumo energético, atendendo a que serão alimentados por baterias do tipo AA e não terão acesso à rede eléctrica.
- pequenas dimensões minimizando o impacto visual atendendo a que serão colocados num ambiente exterior.
- aquisição das amostras relativas à temperatura, humidade e luminosidade presentes em cada um dos módulos.
- colocação num invólucro estanque permitindo que os valores obtidos pelas sensores sejam os mais fiáveis possível.
- um preço relativamente baixo em conformidade com o que existe no mercado.

Em relação à estação base:

- O coordenador da rede ficará conectado por USB a um computador, que servirá como alimentação e para a transferência de dados.
- Os dados provenientes dos módulos ao alcançar o coordenador serão tratados numa aplicação para o efeito, armazenados numa base de dados e disponibilizados numa página WEB.

3.2 Arquitectura da rede

Na Figura 3.1 pode observar-se a configuração proposta para o sistema de monitorização ambiental.

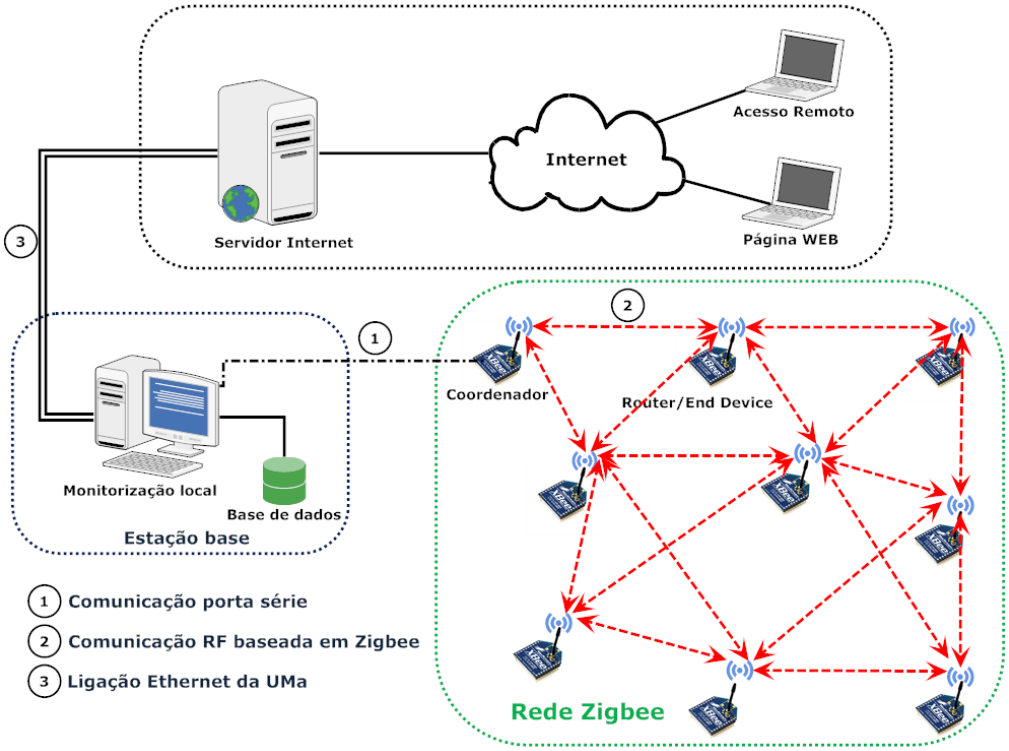


Figura 3.1 – Arquitectura do sistema.

Analisando a Figura 3.1 constata-se a existência de uma **estação base**, uma **Rede *ZigBee*** e um acesso de **ligação à Internet**. A estação base é a responsável pela recepção e manuseamento dos valores provindos dos vários sensores. O PC de monitorização local está ligado a um coordenador por porta série, sendo neste projecto utilizada uma porta USB em vez da comum porta COM.

É na Rede *ZigBee* onde reside a maior parte do trabalho deste projecto. A topologia utilizada para formar a rede poderá ser em árvore ou em malha. Na primeira, as mensagens provenientes dos nós sensores têm apenas um caminho até atingir o coordenador, não sendo uma boa opção para formar uma rede de monitorização. No entanto, esta topologia será importante para testar determinadas funcionalidades da rede, como será apresentado no capítulo 4.

No que diz respeito ao acesso pela Internet, será disponibilizado um sítio que poderá ser consultado para fins de monitorização remota e ainda um acesso remoto à estação de monitorização local, a fim de proceder a ajustes ou possíveis falhas que possam ter ocorrido.

3.3 Dispositivos da rede

3.3.1 Nós sensores

Cada nó sensor é composto por quatro partes essenciais: **alimentação**, **comunicação**, **controlo** e **monitorização**. A arquitectura do nó sensor é ilustrada na Figura 3.2.

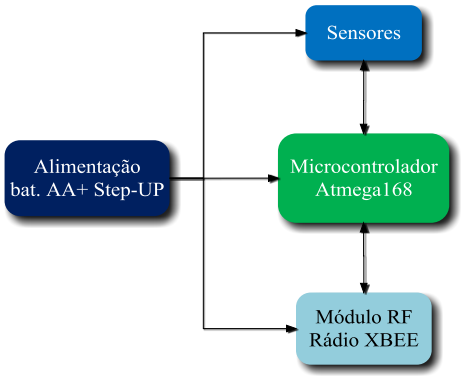


Figura 3.2 - Arquitectura do nó sensor.

- **Alimentação** – O circuito de alimentação do nó sensor é constituído por um par de baterias de formato AA com composição NiMH de1,2V. Foi necessário garantir um nível de tensão de alimentação constante à entrada de cada uma das outras unidades, utilizando-se para esse efeito um *step-up* com tensão de saída de 3,3V, sendo este o valor típico de funcionamento de cada uma das unidades. As baterias podem ser carregadas com recurso a um painel solar.
- **Comunicação** – Esta unidade tem por base a utilização de um módulo de rádio para envio das amostras recolhidas e recepção de mensagens provenientes de outros nós sensores.
- **Controlo** – o sistema de controlo é constituído por um microcontrolador Atmega168.
- **Monitorização** – realizada através do(s) sensor(es) os quais enviam um sinal com o seu valor para o microcontrolador.

O protótipo desenvolvido no âmbito deste projecto foi concebido tendo em conta os detalhes e características indicados pelos fabricantes nos *datasheets* e recorrendo a um protótipo existente nos laboratórios utilizado pelo Eng. Felipe Santos no seu projecto de mestrado. O protótipo desenvolvido apresenta algumas melhorias em termos de espaço, ou seja, com menores dimensões e com uma facilidade para diagnóstico mais acessível.

Os componentes que fazem parte de cada uma das unidades supracitadas serão descritas com um maior detalhe no próximo ponto.

3.3.2 Componentes do nó sensor

3.3.2.1 Alimentação

Baterias

A fonte de energia para o nó sensor foi definido como sendo um par de baterias recarregáveis em série do tipo AA, composição NiMH, com uma carga útil de 2500mAh. A Figura 3.3 apresenta uma das baterias utilizadas no nó sensor.

Figura 3.3 - Bateria recarregável de 1,2V.

O nível de tensão nas baterias decresce durante a descarga, podendo atingir um nível de tensão de 1,1V como mostra a Figura 3.4[27].

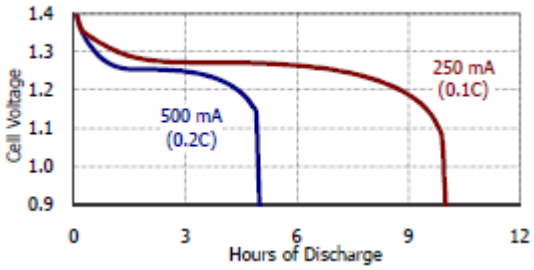


Figura 3.4 - Curvas características da descarga da bateria *Energizer* 2500 mAh.

Pelo gráfico e descrito pelo fabricante, uma bateria dura até sensivelmente 10 horas, numa descarga constante de 250 mA a uma temperatura ambiente de 21ºC. Utilizando um par de baterias em série os níveis de tensão mínimo e máximo duplicam, passando para sensivelmente 2,2V a 2,8V, o que ainda se encontra fora do valor típico de funcionamento de todo o circuito.

Para garantir uma tensão de alimentação de 3,3V utiliza-se um *step-up DC-DC Converter*, neste caso o Max1675.

***Step-up DC-DC Converter* MAX1675**

Este componente é compacto e pertence ao tipo de encapsulamento 8 µMax (8 pinos com 5,05 x 2,01 x 1,10 mm) com uma eficiência de 94% a 200 mA de corrente de saída e tem como finalidade obter uma tensão de saída constante. A gama de tensão de entrada varia desde 0,7V até 5,5V. O arranque do componente é garantido a um nível de tensão de 1,1V. O nível de tensão de saída pode ser fixo em 3,3V ou 5V ou ainda ajustável se assim o desejarmos.

Um dos circuitos fornecidos pelo fabricante para obter um nível de tensão constante de 3,3V (Vcc) pode ser visualizado na Figura 3.5[28].

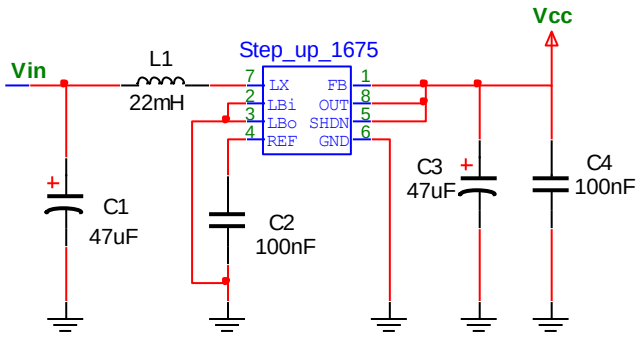


Figura 3.5 - Esquemático eléctrico do Step-up Max1675 para 3.3V.

Pelo esquemático apercebe-se que necessita apenas de seis componentes. A escolha dos componentes mais adequados para o circuito pode ser consultadana folha de características do fabricante, o qual recomenda a utilização de componentes ESR (*Equivalente Series Resistance*)[29] para obter a eficiência mencionada.

Circuito On-OFF

Ainda na alimentação optou-se por anexar um circuito para carregar as baterias com recurso a um painel solar do fabricante *Solarex* modelo MSX-005F[30]. O circuito em questão foi utilizado no trabalho de Mestrado de Lina Teixeira[31]. A Figura 3.6 mostra o circuito em questão.

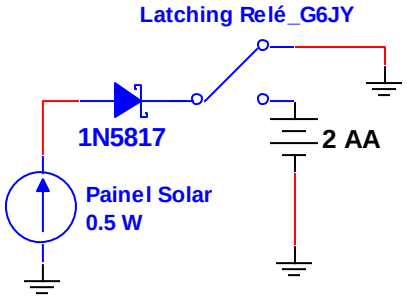


Figura 3.6 - Esquemático electrónico do circuito On-OFF

O circuito contém dois elementos de protecção: um díodo *Schottky* e um relé. O díodo serve para evitar que haja correntes a fluir das baterias para a resistência interna do painel solar, e o relé serve para evitar que as baterias ultrapassem os níveis superiores descritos pelo fabricante. O relé é controlado por dois impulsos provenientes do microcontrolador. No caso de a tensão das baterias ser inferior a 2,8V no momento da amostragem, este comuta para carregá-las, caso contrário comuta para a massa.

Step-Down DC-DC Converter TL2575

Este componente converte tensões de alimentação numa gama de 4,75 a 40V à entrada para 3,3, 5, 12 e 15V à saída dependendo do divisor resistivo colocado à saída. Tem uma eficiência de 88 % e é capaz de fornecer até 1 A de corrente de carga[32].

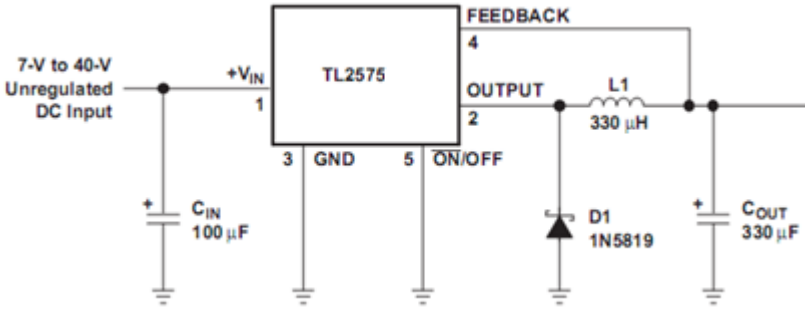


Figura 3.7 - Esquemático eléctrico do *Step-down* TL2575 para 3,3V.

Este circuito serve para alimentar um router que fica presente no terraço da Universidade da Madeira, que tem como finalidade encaminhar as mensagens para o coordenador.

3.3.2.2 Comunicação

Rádio XBee

O módulo do rádio a ser implementado no protótipo é produzido pela empresa DIGI[33] a qual atribuiu o nome de XBee. Os XBees foram concebidos para suportarem o protocolo *ZigBee* e terem um baixo custo. Os módulos XBee têm consumos energéticos baixos e facultam uma entrega consistente de informação entre os dispositivos remotos que operam na banda de frequências dos 2,4 GHz. A DIGI fornece quatro tipos de rádios ilustrados na Figura 3.8, os quais possuem diferentes tipos de antenas de acordo com a aplicação em questão. Para este projecto é utilizado o XBee com conector RPSMA sendo possível a aplicação de várias antenas com diferentes ganhos.

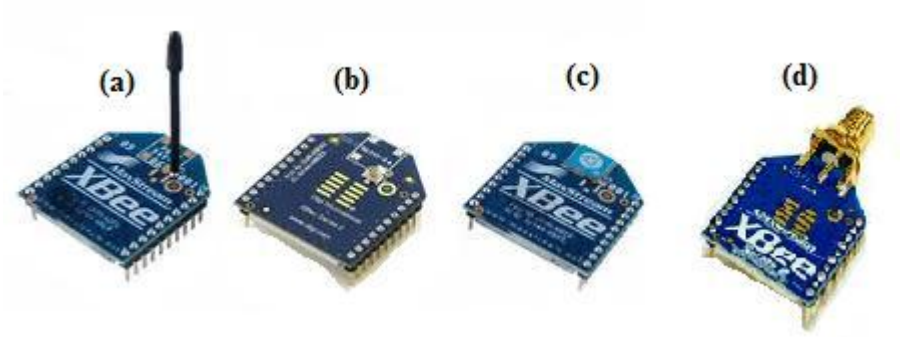


Figura 3.8 – (a) -antena *whip*; (b) - conector U.FL; (c) - antena chip; (d) - conector RPSMA.

As especificações dos módulos podem ser consultadas no

Anexo A.

Este módulo conta ainda com interface *UART* para interligar com um microcontrolador, portas E/S digitais ou analógicas dependendo do pino em questão, e ainda uma saída em PWM (Anexo B).

As portas E/S podem ser configuradas directamente no módulo ou remotamente através do *software* gratuito X-CTU, fornecido pela própria DIGI.

A Figura 3.9 exemplifica a comunicação UART entre o microcontrolador e o módulo XBee e ainda entre dois módulos XBee.

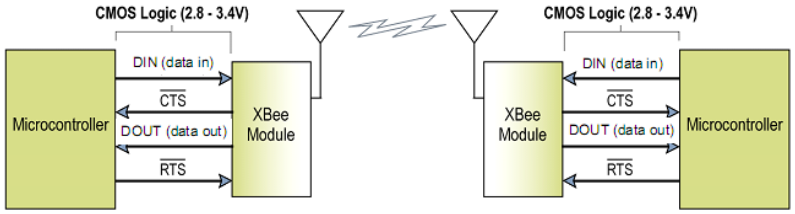


Figura 3.9 - Diagrama de fluxos de dados na interface UART.

Cada módulo XBee é fornecido com um determinado *firmware* com uma pré-configuração. A DIGI produziu até ao momento quatro tipos de *firmware*: ***XBee* 802.15.4**, ***DigiMesh* 2.4** que fazem parte da série 1 e o ***XBee ZNet* 2.5**, ***XBee ZigBee*** pertencentes à série 2. Estes *firmwares* podem ser carregados para os módulos XBee, dependendo do rádio presente no mesmo. Para a elaboração deste projecto decidiu-se optar pela série 2 em que o *firmware* presente fosse **XBee ZNet 2.5.**

As características principais das duas séries são apresentadas na Tabela 3.1.

Tabela 3.1 - Comparação entre os módulos XBee Series 1 e Series 2[34].

	XBee Series 1	XBee Series 2
Alcance RF Interior / urbano	até 30 m	até 40 m
Alcance RF Exterior/linha de vista	até 100 m	até 120 m
Potência de transmissão máxima	1 mW (0 dBm)	2 mW (+3 dBm)
Taxa de transmissão	250 Kbps	
Sensibilidade de recepção	-92dbm (1% PER)	-98dbm (1% PER)
Tensão de alimentação	2.8 - 3.4 V	2.8 - 3.6 V
Corrente de transmissão	45 mA (@ 3.3 V)	40 mA (@ 3.3 V)
Corrente de recepção	50 mA (@ 3.3 V)	40 mA (@ 3.3 V)
Corrente "a dormir"	10 µA	1 µA
Frequência	ISM 2.4 GHz	
Dimensões	0.0960" x 1.087"	

Temperatura de funcionamento	-40 to 85 ºC	
Opções de antenas	Chip, Integrated Whip, U.FL	Chip, Integrated Whip, U.FL, RPSMA
Topologias de rede	Ponto a ponto, Estrela	Ponto a ponto, Estrela, Mesh
Número de canais	16 Direct Sequence Channels	16 Direct Sequence Channels
Opções de filtro	PAN ID, Channel & Source/Destination	PAN ID, Channel & Source/Destination

Escolheu-se esta série porque apresenta melhores características face à série 1 tais como: maior alcance, maior potência de transmissão, uma melhor sensibilidade de recepção, consumos energéticos inferiores nos três estados e possibilidade de usar rádio com terminação RPSMA.

Os módulos XBee podem operar em dois modos diferentes**: AT** e **API.** O modo **AT** é designado por **modo transparente** e neste modo os módulos actuam como se fossem uma linha de comunicação série. O modo **API** é um modo em que a informação é enviada e recebida por tramas (*frames*), os quais podem conter várias informações, tais como endereçamentos, comandos, mudanças de estado, entre outros. Na Figura 3.10 pode observar-se o formato da trama API.

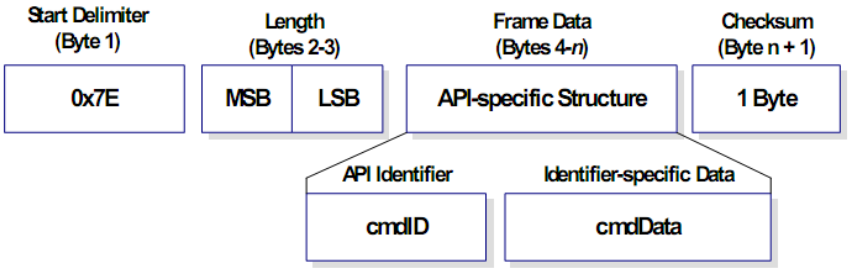


Figura 3.10 - Estrutura da frame de dados UART e especificação da estrutura API.

Legenda:

Start Delimiter	é o primeiro byte da trama e inicia sempre por 7E.
Length	é o comprimento obtido pela soma do número de bytes referentes ao campo *Frame Data.*
Frame Data	é composto por dois campos: *API Identifier* e *Identifier-specific Data*. O campo *API Identifier* indica que tipo de dados API estarão contidas no *Identifier-specific Data*. O campo *Identifier-specific Data* pode ser especificado dependendo do tipo de comando atribuído em *API Identifier*. O Anexo C, contem a estrutura específica dos vários comandos que podem ser utilizados com o *firmware* XBee ZNet 2.5
Checksum	serve para testar a integridade da mensagem. O seu cálculo consiste na soma dos valores dos bytes existentes no campo *Frame Data*. Ao seu resultado é subtraído 0xFF, para obtenção do valor do *checksum.*

3.3.2.3 Controlo

Microcontrolador ATMega168

Existe no mercado uma grande variedade de microcontroladores. Os aspectos principais para a escolha deste microcontrolador são:

- Baixo custo
- Baixo consumo energético
- Capacidade de processamento,
- Possibilidade de utilização de linguagem C para programação utilizando o protocolo USB ISP500.

O ATmega168 é um microcontrolador de 8 bit de tecnologia *CMOS* baseado em arquitectura *RISC* e contém 32 pinos. Apresenta uma memória *Flash* de 16 KB, três *timers*/contadores dois dos quais a 8 bits e o outro a 16 bits, 6 canais *PWM*, 8 canais *ADC* multiplexados para um *ADC* de 10 bit no caso de ser encapsulamento do tipo *TQFP* e *QFN*/*MLF*, uma *UART* programável, Interface *SPI* Mestre/Escravo, tensão de operação entre 2,7 e 5,5V, velocidade de processamento entre 0 a 10MHz a níveis de tensão entre 2,7 a 5,5V e entre 0 a 20 MHz com níveis de tensão entre 4,5 a 5,5V, consumos de 250 µA a uma frequência de 1 MHz e 15 µA a 32 KHz no estado activo e 0,1µA no estado adormecido a uma tensão de 1,8 V[35]. Na Figura 3.11 pode visualizar-se o componente utilizado para o projecto.

Figura 3.11 - Microcontrolador ATmega168 com encapsulamento TQFP[36].

O Anexo D contém uma tabela com a indicação dos pinos que foram utilizados no decorrer do projecto.

Este ATMega168 servirá para obter os valores das amostras provenientes não só por fontes analógicas, como é o caso da tensão de bateria e luminosidade, mas também por fontes digitais, como é o caso do sensor SHT15. Como o microcontrolador possui um ADC de 10 bits e utiliza uma tensão de referência (Pino 20 - AREF) de valor 1,1V será necessário dimensionar a entrada dos ADCs para um valor inferior a 1,1V. O conversor utiliza dois registos internos, o ADCL e o ADCH. O ADCH precisa de sofrer um deslocamento de 8 bits à esquerda para depois ser somado com ADCL e obter a conversão correcta. 0x000 corresponde à

massa e 0x3FF corresponde ao valor da tensão de referência subtraindo um bit menos significativo [35]. A fórmula de conversão é a seguinte:

$$ADC = \frac{V_{IN} * 1024}{V_{REF}} \quad (3.1)$$

onde V_{IN} é a tensão de entrada no pino utilizado e V_{REF} a tensão de referência.

3.3.2.4 Monitorização

Sensor SHT15

O sensor SHT15, fornecido pela empresa *Sensirion*, serve para extrair o valor da temperatura e humidade relativa do ar. A tecnologia CMOS aplicada garante uma excelente confiabilidade e estabilidade a longo prazo e está acoplado a um conversor analógico-digital de 14 bit e a leitura dos valores é obtida através de um canal de comunicação série dedicado, o que permite uma óptima qualidade de sinal, uma resposta rápida e oferece uma grande imunidade às perturbações externas. Este sensor apresenta uma porta digital de saída e um baixo consumo de corrente sendo ideal para inúmeras aplicações[37]. Na Figura 3.12 pode observar-se um sensor SHT1x comparado com a cabeça de um fósforo para uma melhor percepção do seu tamanho.

Figura 3.12 - Sensor SHT1x.

São utilizados dois pinos do microcontrolador para a interface com o sensor SHT15. Pelo Anexo E verifica-se que a alimentação deste sensor está numa gama entre 2,4 e 5,5V, sendo um valor típico de 3,3V o que torna este sensor numa mais valia para o projecto atendendo a que o nível de tensão pretendido foi o último a ser supracitado. O circuito da Figura 3.13 apresenta o esquema eléctrico típico utilizado neste sensor.

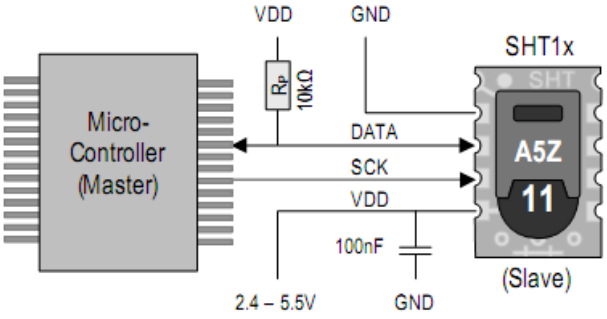


Figura 3.13 - Esquema eléctrico típico de interligação do sht15 com um microcontrolador.

Este sensor é alimentado por uma tensão de 3,3V a qual está acoplada um condensador de 100 nF para minimizar efeitos de ruído. Uma resistência de 10 KΩ é necessária para colocar o sinal no estado *high* (*pull-up*). Este sensor não pode ser endereçado pelo protocolo I^2C (***Inter-Integrated Circuit***), no entanto pode ser interligado num bus I^2C, onde o microcontrolador é o *Master* e o sensor o *Slave. O* pino SCK (*Serial clock input*) é usado para sincronizar a comunicação entre o microcontrolador e o sensor. O pino DATA é usado para transferir dados do sensor para o microcontrolador e vice-versa[37]. Para o correcto funcionamento do sensor, o fabricante disponibiliza o código em linguagem C para ser carregada no microcontrolador. Neste código foram feitas pequenas adaptações e os cálculos finais da temperatura e humidade passam a ser efectuados na estação base para poupar processamento no microcontrolador. O valor da temperatura para o modelo em questão pode ser obtido na estação base através da seguinte fórmula[37]:

$$temp\ (ºC) = amostra_{temp} * 0{,}01 - 40\ (ºC) \qquad (3.2)$$

A humidade é obtida por uma fórmula mais complexa em que é dependente do valor da temperatura e de coeficientes predefinidos pelo fabricante,

$$humid\ (\%) = (temp - 25) * (t1 + t2 * amostra_{humid}) + RH_{linear} \qquad (3.3)$$

$$RH_{linear}(\%) = c1 + c2 * amostra_{humid} + c3 * amostra_{humid}\text{^}2 \qquad (3.4)$$

em que *t1=0,01; t2=0,00008; c1=-4; c2=0,0405; c3=-0,0000028*

Sensor S1087 (fotodíodo de Silício)

Para obter amostras da iluminação solar recorreu-se à utilização dum fotodíodo fornecido pela *Hamamatsu* visível na Figura 3.14. Um fotodíodo é um díodo de junção PN construído de forma especial, de modo a possibilitar a utilização da luz como factor determinante no controlo da corrente eléctrica. A aplicação de luz na junção resulta numa transferência de energia das ondas luminosas incidentes (na forma de fotões) para a estrutura atómica, resultando num aumento do número de portadores minoritários e um aumento do nível da corrente inversa. A corrente "escura" traduz a corrente que existirá sem nenhuma iluminação aplicada. Em resumo, um fotodíodo é um dispositivo que converte a luz recebida numa determinada quantidade de corrente eléctrica [38].

Figura 3.14 - Sensor S1087[39].

Este fotodíodo responde a comprimentos de onda de luz visível entre 320 nm a 730 nm com um pico de sensibilidade nos 560 nm como mostra a Figura 3.15.

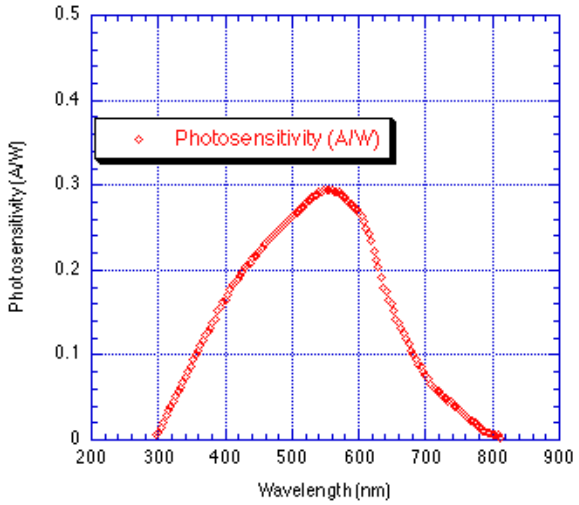


Figura 3.15 - Resposta espectral do fotodíodo S1087.

Como o fotodíodo gere uma corrente à saída, não pode ser conectado directamente a um pino *ADC* do microcontrolador. Uma solução rápida passa por colocar uma resistência em paralelo como mostra a Figura 3.16 para gerar uma queda de tensão a qual já poderia ser lida no *ADC* do microcontrolador.

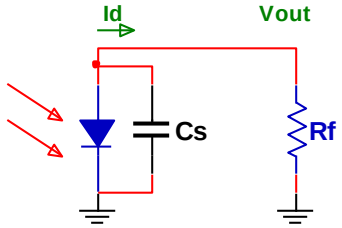


Figura 3.16 - Paralelo com o S1087.

A aplicação de uma resistência em paralelo produz dois problemas. O primeiro surge no caso em que a resistência é elevada o que produz um tempo de resposta lento visto que $t=C_s * R_f$. No caso de ser uma resistência pequena, o tempo de resposta passa a ser rápido, mas o ganho vai ser baixo[40]. Uma forma de eliminar este problema é com recurso a um circuito de transimpedância ou transresistência.

É recomendado pelo fabricante a utilização de um circuito de transimpedância para obter os valores das correntes[41]. Este circuito consiste em converter a corrente proveniente do fotodíodo numa tensão, sendo necessário aplicar um circuito com um amplificador operacional e andar de ganho como mostra a Figura 3.17.

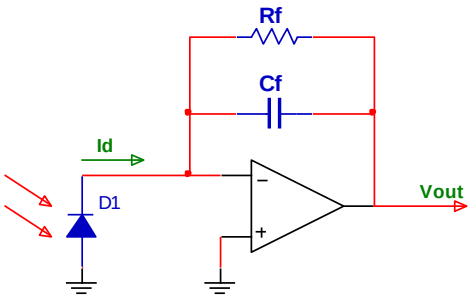


Figura 3.17 - Circuito de transimpedância para o sensor S1087.

A tensão de saída do circuito é dada pela seguinte expressão:

$$Vout\ (V) = Id * Rf \tag{3.5}$$

O tempo de resposta do sinal passa a ser t= R_f*C_f e ganho é definido pela resistência $R_{f..}$ No Anexo F apresenta-se as características do componente em comparação com os outros sensores do mesmo tipo, ou seja, cerâmicos e ainda um gráfico da corrente em função da intensidade luminosa, a qual é dada pela seguinte expressão:

$$Luz(Lux) = Id * 10^9 \tag{3.6}$$

3.3.3 Arquitectura do nó sensor

A arquitectura do nó sensor para monitorização florestal a ser desenvolvido pode ser visualizado na Figura 3.18. Da figura consta os blocos dos componentes mencionados na subsecção anterior.

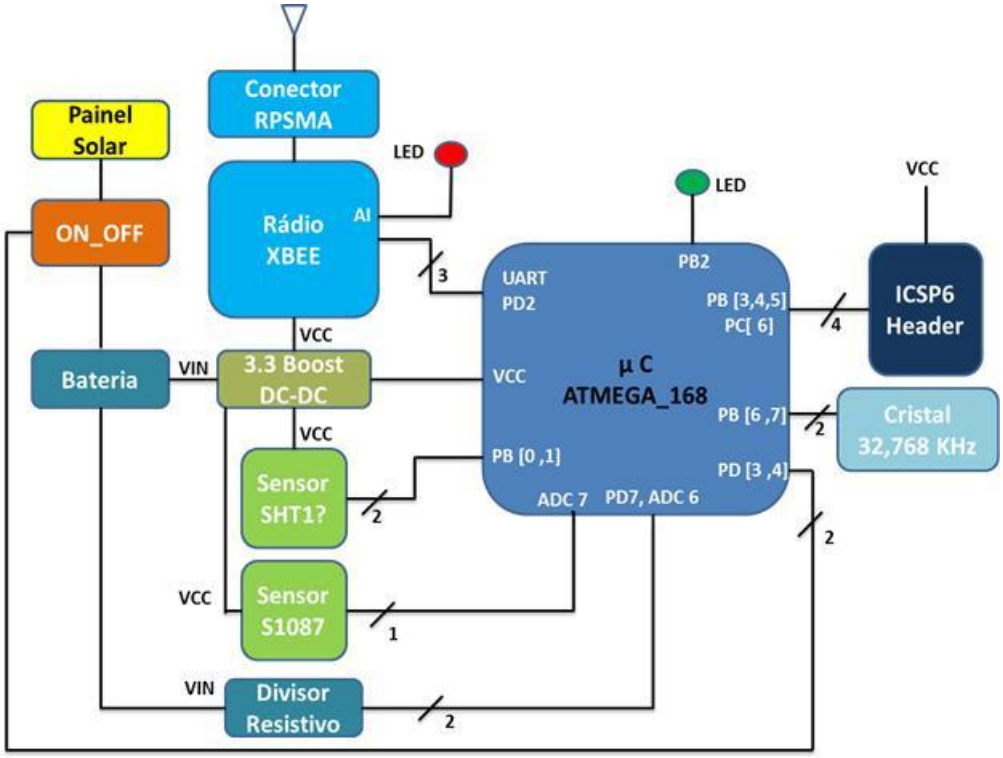


Figura 3.18 - Arquitectura do nó sensor.

Da arquitectura, para além de todos os componentes já mencionados, é de referir o bloco do **Divisor Resistivo** visível na Figura 3.19, que tem como função amostrar o valor da tensão das baterias com recurso a um MOSFET tipo P, o qual serve de interruptor para um divisor resistivo. A função do divisor é de reduzir o nível de tensão *Vin*, para um nível aceitável pelo microcontrolador.

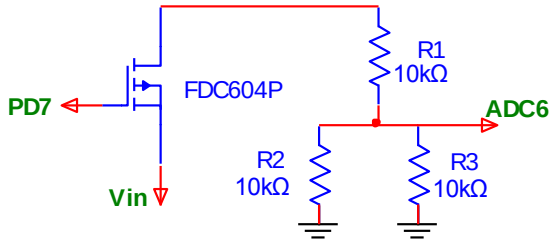


Figura 3.19 - Esquema electrónico do divisor resistivo.

O bloco mencionado como **ICSP6 header** consiste num *header* para poder carregar o *firmware* no microcontrolador com recurso a um programador para microcontroladores AVR designado por AVR-ISP500 [42]. O Anexo G contém informações relativas a este programador e ao circuito eléctrico de interligação entre o ATmega168 e o ICSP6.

3.4 Técnica de adormecimento dos *routers*

Antes de descrever no que consiste a técnica é preciso ter em conta os consumos do nó sensor neste projecto, atendendo a que os mesmos são colocados no meio da floresta sem recurso a fontes energéticas.

3.4.1 Consumos energéticos

A Tabela 3.2 mostra os consumos fornecidos pelos fabricantes de cada componente separadamente, quando é aplicada uma tensão de 3,3V a uma temperatura ambiente de 25ºC.

Tabela 3.2 – Consumos dos vários componentes

Componente	Estado		
	Activo (mA)	*Idle* (mA)	Dormindo (µA)
ATmega168	4	0,8	180
Rádio XBee	Tx – 40 Rx - 40	15	< 1
Sensor SHT15	0,55	-	0,3
AMPOP (circuito de transimpedância)	0,054	-	-
Total	44,60	15,8	181,3

Verifica-se que no estado activo o consumo total previsto para o nó é de 44,60 mA, no estado *idle* o consumo é de 15,8 mA e a dormir apresenta um valor de 181,3 µA. De mencionar ainda que aos totais obtidos falta a contribuição da corrente exigida pelo *Step_Up* para o seu correcto funcionamento.

Para obter uma noção do valor da corrente exigida pelo *Step_Up* foi necessário recorrer ao gráfico da eficiência em função da corrente de carga. Este gráfico é disponibilizado pelo fornecedor através do *datasheet*[28].

Sabendo que a eficiência é dada por:

$$\eta = \frac{P_{out}}{P_{in}} \tag{3.7}$$

e tendo em conta que a potência é dada pelo produto da tensão pela corrente vem que:

$$\eta = \frac{P_{out}}{P_{in}} = \frac{V_{out} * I_{out}}{V_{in} * I_{in}} \tag{3.8}$$

O valor a determinar é I_{in} de modo a saber qual a corrente total exigida pelo nó sensor. Reorganizando a fórmula 3.8 obtém-se:

$$I_{in} = \frac{V_{out} * I_{out}}{V_{in} * \eta} \tag{3.9}$$

A corrente I_{in} é variável, dependendo da tensão de entrada (V_{in}) e da eficiência do *Step_Up*. Consultando o *datasheet* verificou-se que a eficiência não se altera para 2,2 <V_{in}< 2,8, sendo esta de ≈92% no estado activo. No caso contrário (dormindo), *o Step_Up* apresenta uma eficiência na ordem dos 30%.

Admitindo uma tensão constante de saída (V_{out}=3,3V), uma I_{out} = 44,60 mA quando o nó está acordado e uma I_{out} = 181.3 µA dormindo, vem que o consumo total do nó (I_{in}) é dado pela média do valor obtido na Tabela 3.2.

Tabela 3.3 – Consumos energéticos do nó sensor em dois casos (acordado e dormindo).

	Vin (V)	I_{in} (acordado) (mA)	I_{in} (dormindo) (µA)
	2,80	56,42	854
	2.75	57,44	870
	2.70	58,51	886
	2.65	59,61	903
	2.60	60,76	920
	2.55	61,95	938
	2.50	63,19	957
	2.45	64,48	976
	2.40	65,82	997
	2.35	67,22	1018
	2.30	68,68	1040
	2.25	70,96	1063
	2.20	73,36	1087
Média	2.50	63.73	963

A partir do valor médio da corrente em ambos os estados, verifica-se que acordado o nó sensor necessita de 63,73 mA e a dormir é de 963 µA.

O nó sensor é alimentado por 2 baterias recarregáveis do tipo AA com uma carga máxima de 2500 mAh. A fórmula para saber a duração que um par de baterias leva a descarregar é dada pela expressão:

$$Duração\ (horas) = \frac{Carga\ das\ baterias}{Consumo\ do\ nó} \tag{3.10}$$

Para o caso do nó estar acordado e considerando I_{in} (acordado) médio vem:

$$Duração\ (h)_{acordado} = \frac{2500\ mAh}{63.73\ mA} = 39.22\ horas = 1.63\ dias$$

Para o caso do nó estar dormindo tendo em conta I_{in} (dormindo) médio vem:

$$Duração\ (h)_{dormindo} = \frac{2500\ mAh}{963\ \mu A} = 2596\ horas = 108\ dias$$

Os cálculos efectuados anteriormente servem apenas para indicar as durações das baterias em dois estados separados. O nó precisa de trabalhar nos dois estados, pelo que é necessário criar um ciclo em que o nó esteja acordado o tempo necessário para se associar à rede *ZigBee* e transmitir a mensagem com os valores das amostras e adormecer de seguida.

Numa rede *ZigBee* como mencionado no capítulo 2 existem três tipos de dispositivos lógicos: coordenador, *router* e *end-device*. Como o coordenador é alimentado por um cabo USB não é motivo de preocupação em termos de consumo energético, ao contrário do router e *end-device*.

Os *routers* servem para retransmitir ou mesmo transmitir mensagens caso estejam a efectuar algum tipo de amostragem. Partindo do princípio que não podem dormir, os routers apresentam uma duração de 1,63 dias, o que é grande desvantagem, implicando uma substituição de baterias com uma frequência quase diária.

Os *end-device*, já possuem a opção de serem colocados a dormir após a amostragem, o que faz com que a durabilidade das baterias seja superior.

O novo tempo de vida das baterias vai depender da duração de ambos os estados. As fórmulas 3.11, 3.12, 3.13 e 3.14 servem para estimar a nova duração das baterias quando estas trabalham sobre um ciclo. A expressão 3.11 é igual à expressão 3.10 apenas quando não existe uma alteração no estado do dispositivo (acordado, dormindo).

$$Duração\ (horas) = \frac{Carga\ das\ baterias}{Consumo\ médio\ do\ nó} \tag{3.11}$$

$$Consumo\ do\ nó_{acordado}\ (mA.s) = \overline{I_{in}}acordado * duração_{acordado} \tag{3.12}$$

$$Consumo\ do\ nó_{dormindo}(mA.s) = \overline{I_{in}}dormindo * duração_{dormindo} \tag{3.13}$$

$$Consumo\ médio\ do\ nó\ (mA) = \frac{\overline{I_{in}}acordado * duração_{acordado} + \overline{I_{in}}dormindo * duração_{dormindo}}{duração\ do\ ciclo\ (s)} \tag{3.14}$$

Como os tempos de amostragem podem variar dependendo do tipo de aplicação em questão, a Tabela 3.4 indica a possível duração do nó sensor para 1, 1.5, 2, 5, 20, 60 e 240 minutos sendo este último tempo, o valor máximo descrito pelo fabricante que um rádio XBee pode adormecer. O valor da duração do nó sensor acordado mantém-se fixa, nos 30 segundos, de modo a obter um consumo constante de 1911.9 mA.s

Tabela 3.4 - Duração do nó sensor com várias durações de ciclo

Duração do ciclo (s)	Duração dormindo (s)	$I_{in•dormindo}$ (mA)	$I_{in•médio}$ (mA)	Duração das baterias (horas | dias)
60	0	0	63.73	39.22 | 1.63
60	30	28.89	32.34	77.28 | 3.22
90	60	57.78	21.88	114.2 | 4.75
120	90	86.67	16.65	150.1 | 6.25
300	270	260.01	7.24	345.3 | 14.38
1200	1170	1126.70	2.53	987.2 | 41.13
3600	3570	3437.91	1.48	1682.3 | 70
14400	14370	13838.31	1.093	2285.6 | 95.23

Verifica-se a partir da Tabela 3.4, que à medida que a duração do nó a dormir aumenta, a duração das baterias por consequência também aumenta. Numa situação em que o nó não adormece, tem-se uma duração como já mencionada de 1,63 dias o que é muito pouco quando comparada com a duração máxima permitida, ou seja, 95,23 dias o que dá aproximadamente 3.17 meses de vida útil para o dispositivo *end-device*. Estes valores teóricos podem diferenciar da prática dependendo de vários factores tais como: a temperatura ambiente, tempo de vida que as baterias já possam possuir, efectuando recargas nas baterias com a utilização de carregadores inadequados, entre outros.

3.4.2 Técnica de adormecimento

O funcionamento normal de uma rede Zigbee implica que os *routers* estejam sempre ligados, ou seja, não possuem a facilidade de adormecimento. Como verificado anteriormente a estimativa da duração de um *router* acordado é de 1,63 dias, pelo que na maioria das aplicações estes encontram-se alimentados pela tensão da rede (230V AC) após ser convertida para 3,3V.

Para minimizar o consumo dos *routers ZigBee* foi necessário desenvolver uma técnica que efectuasse a comutação de *router* para *end-device* e vice-versa. Como já mencionado, os *end-devices* não possuem a capacidade de retransmitir mensagens ao contrário dos *routers*, mas conseguem adormecer enquanto que os *routers* não. Desta forma, será necessário utilizar um nó sensor com os dois tipos de dispositivos lógicos para garantir um correcto funcionamento da rede.

Ao utilizar os dois tipos lógicos, *router* e *end-device*, já é possível aplicar um ciclo em que, quando se necessita de enviar mensagens ou retransmiti-las, o rádio XBee encontra-se no estado *router*, após esta tarefa estar completa o rádio XBee comuta para *end-device* e vai dormir. Quando o *end-device* acorda, comuta de estado para *router* para poder formar a rede e começar a enviar mensagens novamente. De mencionar que esta comutação é efectuada pelo microcontrolador através de comandos específicos para o efeito.

3.5 Propagação do sinal rádio frequência

A propagação das ondas de rádio impõe perdas ao sinal, existindo diversas causas para a degradação do mesmo. Num ambiente sem obstáculos a atenuação é a do espaço livre. Tendo em conta que o sistema tem a sua principal aplicação ao ambiente florestal, tem-se que ter em conta uma atenuação adicional, devido à vegetação e solo, de modo que o sinal chega ao receptor por múltiplos percursos.

3.5.1 Modelo de atenuação em espaço livre

Este modelo deve ser utilizado quando existe uma linha de vista entre o emissor e o receptor em que ambos encontram-se livres de obstáculos circundantes.

O valor da perda no espaço livre pode ser calculado através da equação de *Friss*[43]*:*

$$PL_{FS}(dB) = 32.44 + 20log_{10}(f) + 20log_{10}(d) \quad (3.15)$$

em que a *f* é a frequência em GHz e *d* a distância entre emissor e receptor. Isto é um modelo teórico válido para longas distâncias na situação em que não existem obstáculos na primeira elipsóide de *Fresnel* [44].

3.5.2 Modelos de propagação em meios com vegetação

Para modelos empíricos, verificou-se que o modelo desenvolvido por *Weissberger* estima o excesso de atenuação produzida pela vegetação,

$$L_w(dB) = \begin{cases} 0{,}45f^{0{,}284}d & d \leq 14m \\ 1{,}33f^{0{,}284}d^{0{,}588} & 14m \leq d \leq 400m \end{cases} \quad (3.16)$$

com *f* em GHz e *d* em m. O modelo COST 235 considera a situação em que as árvores se encontram num meio com ou sem vegetação. O excesso de atenuação é dado por,

$$L_{COST}(dB) = \begin{cases} 15{,}6f^{-0{,}009}d^{0{,}26} & Meio\ com\ folhagem \\ 26{,}6f^{-0{,}2}d^{0{,}5} & Meio\ sem\ folhagem \end{cases} \quad (3.17)$$

com *f* em MHz e *d* em m. Considerando a recomendação ITU-R, Al-Nuaimi e Stephens desenvolveram um modelo utilizando os dados medidos na frequência 11,2 GHz e de 20 GHz (modelo FITU-R),

$$L_{FITU}(dB) = \begin{cases} 0{,}39f^{0{,}39}d^{0{,}25} & Meio\ com\ folhagem \\ 0{,}37f^{0{,}18}d^{0{,}59} & Meio\ sem\ folhagem \end{cases} \quad (3.18)$$

com *f* em MHz e *d* em m. Meng, Lee e Ng optimizaram estas expressões para as bandas VHF e UHF com os dados medidos a 240 MHz e 700 MHz. O modelo tem em conta a componente lateral onde o sinal percorre o topo das árvores. O modelo lateral ITU-R é definido por,

$$L_{LITU}(dB) = 0{,}48f^{0{,}43}d^{0{,}13} \quad (3.19)$$

válido para meios com folhagem, uma vez que o modelo foi desenvolvido com medições realizadas num meio com vegetação pouco densa.

A partir dos modelos apresentados, destaca-se que estes dependem da frequência *f* e da distância *d* não sendo considerados outro tipo de parâmetros relacionados com a floresta. A Figura 3.20 ilustra uma comparação das perdas de sinal para modelos empíricos.

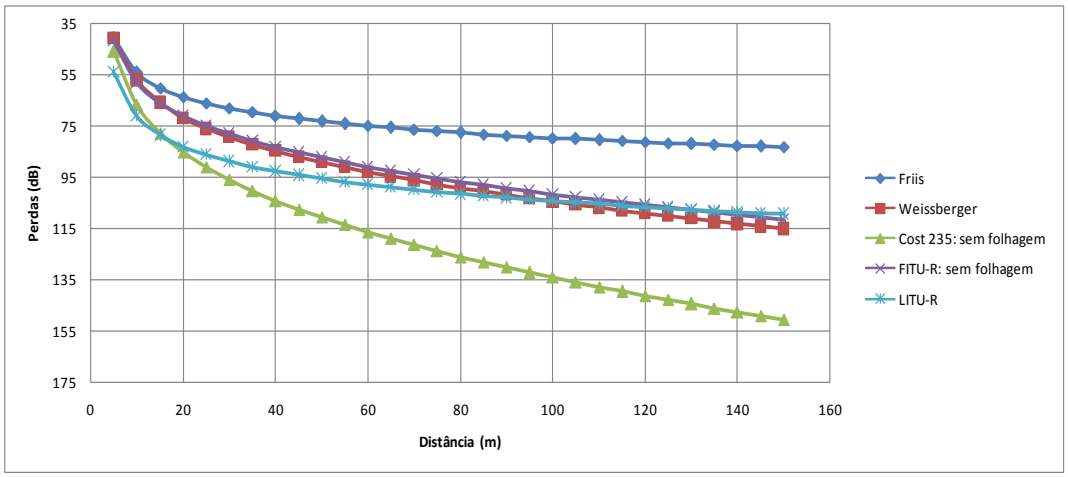


Figura 3.20 - Comparação entre modelos empíricos.

Outro modelo importante que é bastante utilizado é o *log-normal* com perdas de percurso dadas por,

$$PL_{LN}(dB) = PL(d_0) + 10n\log_{10}\left(\frac{d}{d_0}\right) + X_\sigma \tag{3.20}$$

com *n* sendo o indicador de perdas de percurso, que indica a gama de sinal atenuado face à distância, por exemplo n = 2 indica que estamos na presença do espaço livre, $PL(d_0)$ indica as perdas de percurso a uma distância conhecida d_0 no campo distante e X_σ é zero quando é considerada uma variável aleatória Gaussiana (em dB) com desvio padrão σ, e reflecte a variação da potência recebida em torno da média. As três variáveis de (3.21) podem ser utilizadas para se obter a relação entre as perdas de percurso dentro da floresta e os parâmetros das árvores[44].

4 Desenvolvimento do protótipo e do *software*

No quarto capítulo são mencionadas as várias faces do processo de desenvolvimento do protótipo. O segundo ponto consiste na colocação do protótipo dentro de uma caixa e aspectos relevantes sobre a posição dos sensores na caixa. É descrito no terceiro ponto a configuração de uma rede Zigbee, onde serão mencionados aspectos referentes à configuração dos rádio XBee. As ferramentas de software utilizadas, nomeadamente o *AVR Studio* juntamente com fluxogramas para uma melhor percepção do código perfazem o quarto ponto. Segue-se o tratamento, armazenamento e visualização dos dados recebidos pela rede Zigbee.. No último ponto é descrito com um maior pormenor a técnica de adormecimento dos routers.

4.1 Processos de fabrico do nó sensor

O *software* a ser utilizado na elaboração da placa de circuito impresso (PCI) referente ao fabrico do nó sensor foi o *NI Ultiboard 11* da empresa *National Instruments*. A colocação dos componentes na PCI é um processo complexo e moroso, visto que será necessário garantir um determinado espaço para os componentes e para as suas ligações, verificar se o componente existe em *stock* e qual a melhor posição para ser colocado. Nem todos os componentes necessários à elaboração da placa existem nas bibliotecas do *software*, sendo por vezes necessário recorrer à implementação dos componentes para satisfazer a necessidade pedida como é o caso do *socket* do rádio XBee.

O fabrico das placas passa essencialmente por seis processos: esboço, impressão, sensibilização por UV, revelação, perfuração e soldadura.

4.1.1 Esboço (*layout*)

Após a leitura do tutorial[45], começou-se a elaboração do esboço do nó sensor. A elaboração de uma PCI deverá ser efectuada de acordo com determinados protocolos[46]. O primeiro passo consiste em escolher as dimensões que a placa irá assumir, seguindo-se a escolha da tecnologia, que neste caso, é utilizado placas com pré-sensibilização de dupla-face. A tecnologia de placas pré-sensibilizadas de dupla-face permite minimizar as dimensões do nó, porque é possível colocar componentes em ambas as superfícies da placa. No Anexo H pode visualizar-se os vários esboços referentes à construção do nó sensor.

4.1.2 Impressão em papel de acetato

O próximo passo consiste em imprimir os esboços numa impressora a laser, utilizando uma folha de acetato. É necessário alinha-los, ou seja, sobrepô-los utilizando os alvos criados ao lado das placas como se pode verificar na Figura 4.1. Ao sobrepor os

esboços devemos deixar um espaço entre estas para que seja possível colocar uma placa pré-sensibilizada de dupla-face no meio.

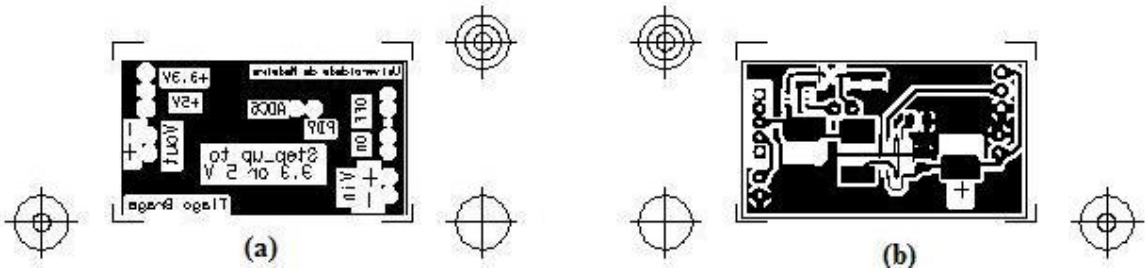


Figura 4.1 – Impressão do esboço: (a) – face superior, (b) – face inferior

De referir que a impressão do esboço referente à face superior está reflectida, devendo-se ao facto de que o lado em que é impressa deve estar em contacto directo com a placa pré-sensibilizada minimizando, desta forma, possíveis erros e falhas que possam ocorrer no momento da sensibilização por UV.

4.1.3 Método por sensibilização por UV

Para transferir a imagem contida na folha de acetato para a placa pré-sensibilizada é necessário utilizar lâmpadas ultravioletas. O tempo de exposição da placa às lâmpadas é variável, sendo indicado pelo fabricante um tempo de aproximadamente 120 segundos para o tipo de placas em questão. Os raios ultravioletas são perigosos para os olhos, sendo por isso utilizado um máquina, visível na Figura 4.2 onde as lâmpadas estão resguardadas.

Figura 4.2 – Máquina de sensibilização.

4.1.4 Revelação das pistas e remoção do cobre

Após a placa estar exposta durante 120 segundos aos raios ultravioletas, a placa é removida da máquina e colocada em banho em duas soluções corrosivas diferentes. A primeira consiste em água com soda cáustica para revelação do desenho que foi transferido, levando em média cerca de 1 minuto. A passagem desta primeira solução para a segunda é efectuada quando for possível visualizar o esboço na perfeição. A segunda solução consiste em água com percloreto de ferro, o qual tem como finalidade remover o cobre desnecessário, podendo levar até cerca de 1 hora a estar concluído.

4.1.5 Preparação da placa para soldar

Após remover a placa do banho de percloreto de ferro é necessário lava-la com água abundante para remover toda a solução corrosiva. O próximo passa consiste em perfurar a PCB nos locais onde levam componentes do tipo THT com a ajuda de um berbequim. Neste processo é preciso ter em conta os tamanhos dos alvéolos presentes na placa. São utilizadas brocas de 0,5, 0,8, 1,0 e 1,1 mm dependendo do componente em questão.

4.1.6 Montagem dos componentes

A montagem dos componentes é um processo delicado, porque é preciso ter em atenção vários aspectos: verificar a temperatura utilizada na estação de soldar, atendendo a que existem componentes mais susceptíveis de ficarem danificados com temperaturas mais elevadas; a correcta posição dos circuitos integrados; iniciar a soldadura sempre do interior da placa para a periferia; começar pelos componentes SMD e finalizar com os componentes de maior envergadura.

4.1.7 Componentes utilizados e custo do nó sensor

Antes de se proceder à compra dos componentes foi necessário averiguar se existiam componentes com preços mais acessíveis e que tivessem a mesma funcionalidade. A escolha da maioria dos componentes vai de encontro com o que é descrito nos *datasheets*. Outros custos adicionais ao nó sensor são a caixa com a norma IP65, a fita de velcro, o tubo de PVC, o filtro, junta tórica, silicone, mão-de-obra, placa PCB, montagem e instalação. Na Tabela 4.1 são apresentados os componentes com uma curta descrição, com o fabricante de cada componente e com o preço aplicado. Os preços datam de 20 Novembro de 2010 à empresa *Farnell*.

Tabela 4.1 - Descrição e preços dos componentes de um nó sensor

Qt.	Descrição produto	Nome e ref, fabricante	Preço/1u (€)
2	Resistor, 0805, 2,2kr, 5%	Multicomp - Mc 0,1w 0805 5% 2k2	0,03
6	Capacitor, 0805, 100nf, 50v, x7r	Kemet - c0805c104k5rac	0,04
1	Resistor, 1210, 5,5kr , 1%	Vishay draloric crcw121010k0fkea	0,05
1	Capacitor, ceramic disc, 220pf	Multicomp mcc11b221k3as6l-rh	0,097
5	Resistor, 1210, 10kr , 1%	Vishay draloric crcw121010k0fkea	0,124
1	Diode, schottky, 1a, 20v	Vishay Formerly I.R. Vs-1n5817	0,18
1	Crystal, Watch, 32,768khz, 12,5pf	Citizen America - Cfs308 32,768kdzf-Ub	0,25
1	Led, 3mm, Green	Led technology l02r3000f1	0,36
1	Mosfet, P, Supersot-6	Fairchild semiconductor - fdc604p	0,51
1	Slide Switch, Spdt	C & k - os102011ms2qn1	0,62
1	Choke, Smd 22uh, 0,22a	Wuerth elektronik - 744030220	0,95
1	Led, 5mm, Bright, Red	Led technology l07r5000q1	1,11
2	Capacitor, 47uf, 16v	Multicomp - Mc 0,1w 0805 5% 2k2	1,80
2	Socket, Vertical, 1row, 10way	Harwin M22-7131042	2,09
1	Holder, Battery, 2xaa, Wirel Leads	Pro Power - Bh322-1a	2,91
1	Ic Sm, Op Amp, Dual	Linear technology ltc6078hms8#pbf	3,10
1	Header, Straight, 2x3way	Tyco electronics / amp 826925-3	3,60
1	8-Bit Mcu, 16k La,2,7-5,5v, Tqfp32	Atmel-Atmega168p-20au	4,21
1	S1087, Photodiode, 560nm, Ceramic	Hamamatsu	5,16
1	Converter, Step Up Dc-Dc, Smd, 1675	Maxim integrated products max1675eua+	5,33
1	Antenna, 2.4ghz, 1/4 Dipole, Rpsma	Pulse engineering w1030	6,58
1	Relay, Smd, Dpdt, 3vdc, Latching	Omron Electronic Components g6ju-2fsy 3dc	8,34
1	Battery, Nimh 2650mah, Aa, Pk4	Duracell - 5000394065710	13,25

1	Solar Panel, 0.5w	Solarex Msx-005f	26,47
1	Sensor, Humidity & Temp, V4	Sensirion sht15	28,63
1	Module, Zigbee, XBee Znet 2.5, 1mw	Digi international xb24-bsit-004	34,03
		Total	154.43

Como se pode verificar, o preço total do nó sem contar com os custos adicionais, é de 154,43€. Como se pode averiguar a partir da tabela, os componentes considerados mais caros são os essenciais para o funcionamento do nó.

Quanto ao *router* com o circuito de *Step_Down*, atendendo a que não terá qualquer função de monitorização, o seu custo é bastante mais reduzido, se estivermos apenas a considerar os componentes que perfazem o nó, ou seja, sem contar com os elementos da alimentação.

Tabela 4.2 - Descrição e preços dos componentes do router com o circuito de *Step_Down*

Qt.	Descrição produto	Nome e ref, fabricante	Preço/1u. (€)
3	Capacitor, 0805, 100nf, 50v, x7r	Kemet - c0805c104k5rac	0,04
1	Resistor, 1210, 10kr , 1%	Vishay draloric crcw121010k0fkea	0,124
1	Diode, schottky, 1a, 20v	Vishay formerly i.r. vs-1n5817	0,18
1	Capacitor, 100uf, 16v	Rubycon 16zl3308x11.5	0,26
1	Terminal block, 2way	Camden ctb3051/2	0,34
1	Led, 3mm, green	Led technology l02r3000f1	0,36
1	Capacitor, 330uf, 16v	Rubycon 6zl3308x11.5	0,42
1	Choke, 330uh, 0.350ohm, 1.0a	Panasonic elc11d331f	1,49
1	Step-dwn sw v-reg 12v d2pak-5	Texas instruments tl2575-12ikttr	1,79
2	Socket, vertical, 1row, 10way	Harwin m22-7131042	2,09
1	Holder, battery, 2XAA, wirel leads	Pro power - bh322-1a	2,91
1	Header, straight, 2x3way	Tyco electronics / amp 826925-3	3,6
1	8-bit mcu, 16k la,2,7-5,5v, tqfp32	Atmel-atmega168p-20au	4,21
1	Antenna, 2.4ghz, 1/4 dipole, rpsma	Pulse engineering w1030	6,58
1	Module, zigbee, XBee znet 2.5, 1mw	Digi international xb24-bsit-004	34,03
		Total	60.59

4.2 Construção do nó sensor

Neste ponto serão descritos alguns dos processos efectuados na colocação dos vários módulos pertencentes a cada um dos nós sensores. Na Figura 4.3 pode visualizar-se um dos **oito** nós sensores construídos (protótipos).

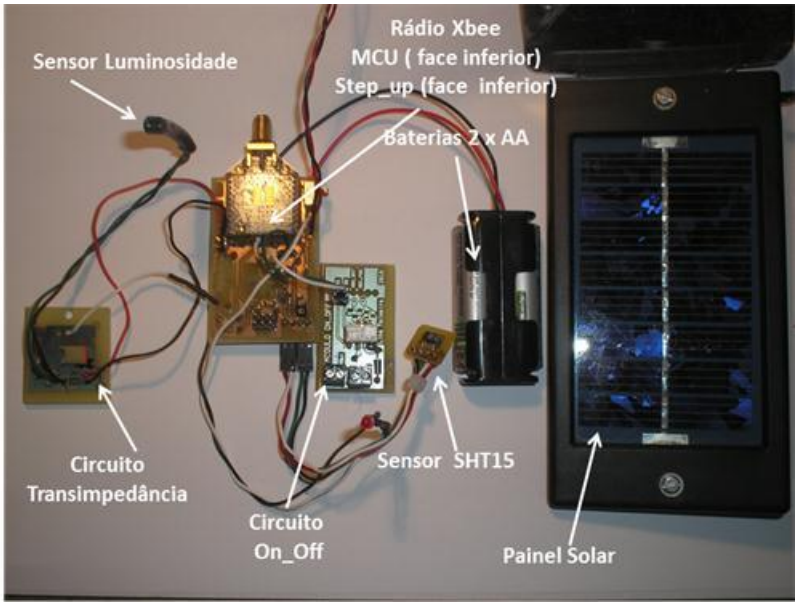


Figura 4.3 - Nó sensor (protótipo)

Atendendo a que os nós são colocados no ambiente exterior, sujeitos a condições atmosféricas diversas, optou-se por colocá-los dentro de caixas com a norma IP65[47]. O espaço da caixa é adequado para a colocação dos vários módulos referentes ao nó sensor como mostra a Figura 4.4.

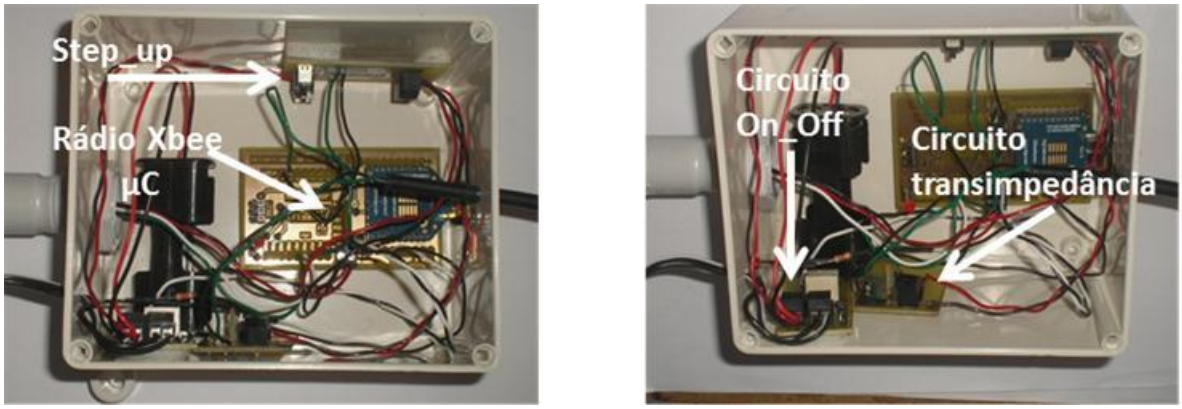


Figura 4.4 - Protótipo final colocado na caixa.

Ao utilizar módulos XBee com conectores RPSMA é de interesse que estas terminações fiquem no exterior da caixa para ser possível colocar uma antena. De modo a garantir que não entre água dentro da caixa foi colocado uma junta tórica na terminação RPSMA, como ilustra a Figura 4.5. A terminação RPSMA vem com uma porca de origem, a qual é enroscada no exterior na caixa permitindo desta feita que o rádio XBee fique fixo e que a junta tórica impermeabilize o furo.

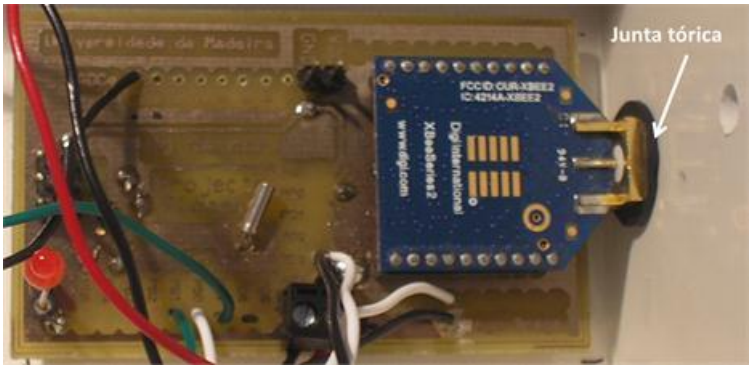


Figura 4.5 - Conector RPSMA com junta tórica

No fundo da caixa é efectuado outro furo para a colocação de um LED o qual fornece informação do estado do rádio XBee. Como as caixas serão fixas verticalmente em árvores ou postes, torna-se mais cómodo para o utilizador verificar o estado do rádio.

O sensor de luminosidade foi colocado no exterior da caixa, protegendo as ligações com recurso à utilização de um cabo visível na Figura 4.6. Na transição da lateral superior para o interior da caixa foi aplicado silicone para prevenir a entrada de água para o seu interior.

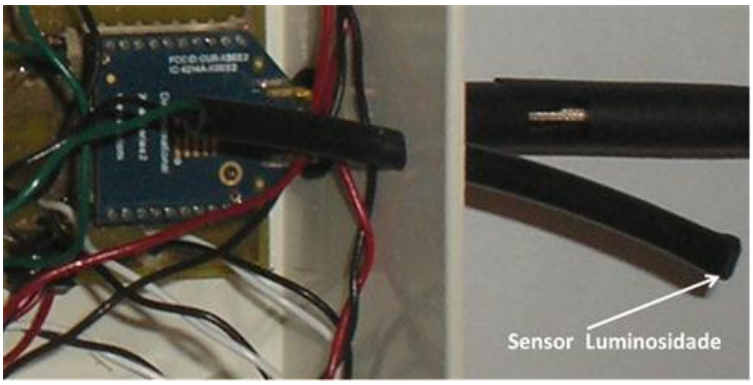


Figura 4.6 - Sensor de luminosidade

O circuito de transimpedância é fixo contra uma das laterais da caixa com recurso a fita de velcro, como se pode verificar na Figura 4.4.

Para que os valores obtidos pelo sensor de humidade e temperatura (SHT1x) fossem os mais fiáveis possível, foi necessário colocar o sensor na parte exterior da caixa. De modo a não ficar exposto ao sol como recomendado pelo fabricante, o sensor é colocado dentro de um tubo PVC o qual é enroscado na base da caixa. Outro aspecto relevante foi eliminar o calor proveniente da caixa devido à exposição solar a que este pode estar sujeito, sendo para isso colocado uma anilha de borracha, a qual fica encaixada no tubo de PVC no momento da montagem como ilustra o seguimento das fotografias da Figura 4.7. Na extremidade do tubo de PVC é colado um filtro para impedir o acesso de insectos e rastejantes ao interior da caixa permitindo desta forma que o sensor fique em contacto com o ar. Como as caixas serão colocadas verticalmente, o tubo de PVC estará sempre virado para o solo, não correndo o risco de entrada de água para o interior da caixa.

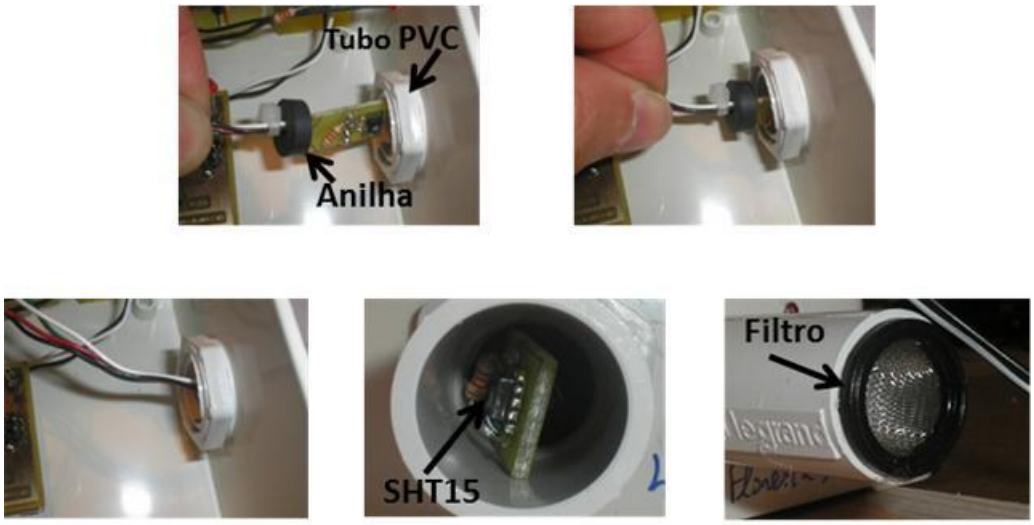


Figura 4.7 - Colocação do sensor SHT15 no tubo PVC

O circuito On-Off foi colocado o mais próximo do fundo da caixa atendendo a necessitamos de mais um furo para a entrada do cabo proveniente do painel solar. Sendo o furo efectuado no fundo da caixa, pode minimizar-se novamente a probabilidade de entrada de água sendo o mesmo protegido com silicone após o cabo estar ligado no terminal deixado para o efeito.

Como o circuito do *Step-Up* contém um interruptor na sua configuração foi necessário fixar o circuito de modo a que o interruptor esteja com um acesso facilitado para o utilizador.

Ambos os circuitos supracitados foram fixados às laterais da caixa com recurso a fita de velcro (Figura 4.4), não necessitando, desta forma, de efectuar um furo para fixação.

A ilustra o protótipo final embebido na caixa juntamente com o painel solar.

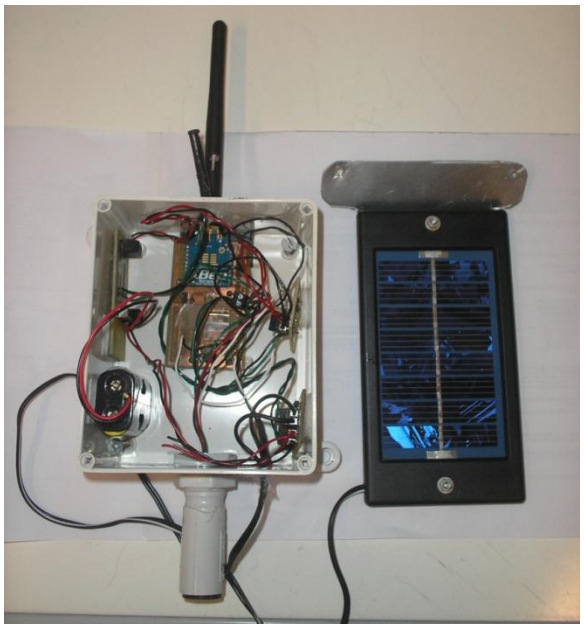


Figura 4.8 - Protótipo final (nó sensor)

Além dos oito nós construídos com as características indicados foi construído um outro, com funcionalidades de *router*, que será responsável por encaminhar todos os dados dos outros nós para a *gateway* existente no laboratório da Universidade. Este protótipo é diferente dos restantes porque não inclui sensores para além de que a sua alimentação é proveniente de uma bateria de 12V/70 Ah que está colocada no terraço. Na Figura 4.9 pode observar-se o interior da caixa, a qual contém apenas o rádio XBee juntamente com o microcontrolador e o circuito de *step-down*.

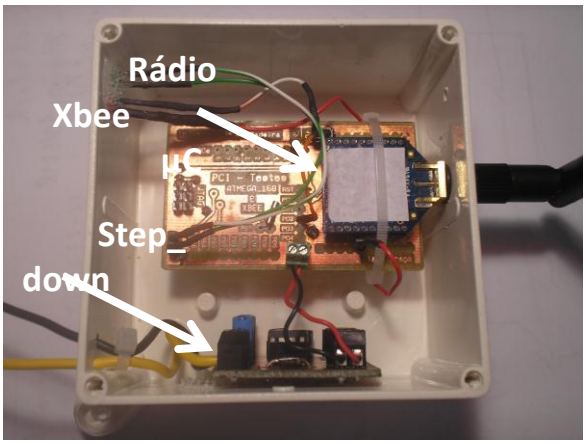


Figura 4.9 - Router com *step down*

4.3 Estudo das antenas utilizadas

Neste projecto foram utilizadas dois tipos de antenas, a antena monopolo (8 monopolos adquiridos pela Universidade) e a antena grelha (duas antenas construídas em laboratório). Para o cálculo do ganho destas antenas foram efectuadas um conjunto de medições, onde se tinha uma antena monopolo a emitir com uma potência de saída de 9 dBm. Com cada um das antenas utilizadas (monopolo e grelha) registaram-se os valores da potência recebida de metro a metro. Os valores obtidos encontram-se na Figura 4.10.

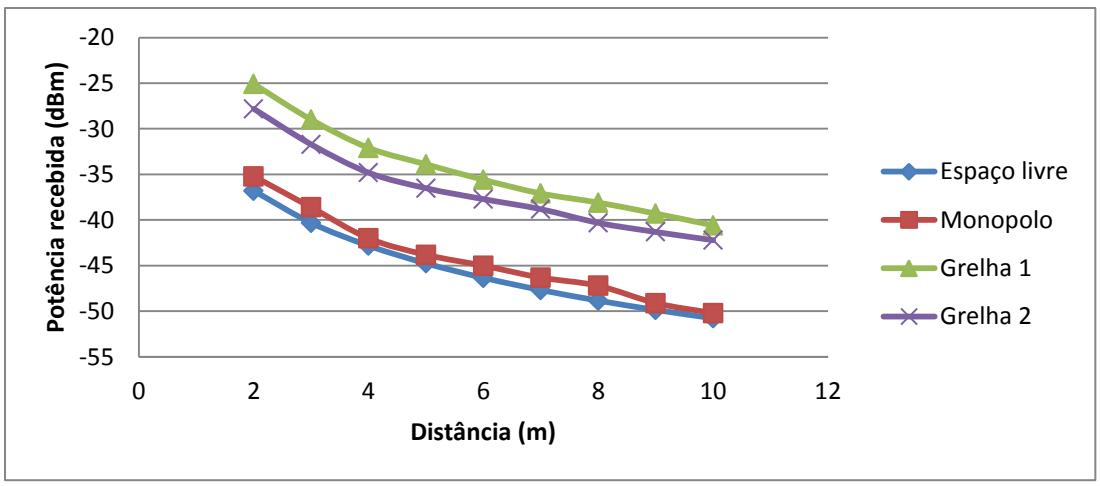


Figura 4.10 - Potência recebida em função da distância

Tendo como referência a curva do espaço livre, obtiveram-se os ganhos referentes para cada uma das antenas. A diferença do valor da potência recebida entre a curva do espaço livre e a antena em questão permite calcular o ganho.

Sendo assim obtiveram-se os seguintes ganhos: Monopolos ≈1,3 dBi, Grelha 1 ≈11 dBi, Grelha2 ≈9 dBi.

4.4 Configuração da rede *ZigBee*

Para a elaboração de uma rede 802.15.4/*ZigBee* são necessários dois rádios XBee. A configuração destes rádios é efectuada recorrendo a uma placa de circuito impresso fornecida pela empresa DIGI aquando da aquisição do material didáctico, a qual pode ser visualizada na Figura 4.11.

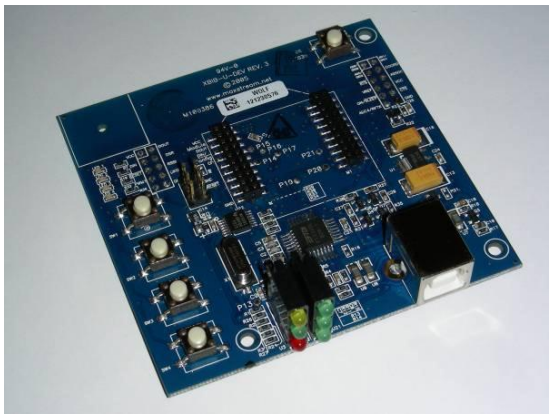


Figura 4.11- PCI-USB.

A placa de circuito impresso permite interligar o PC ao rádio XBee através de uma interface USB. A configuração do rádio é efectuada pelo programa X-CTU que pode ser adquirido gratuitamente em www.digi.com. Na Figura 4.12 pode visualizar-se a janela principal do programa.

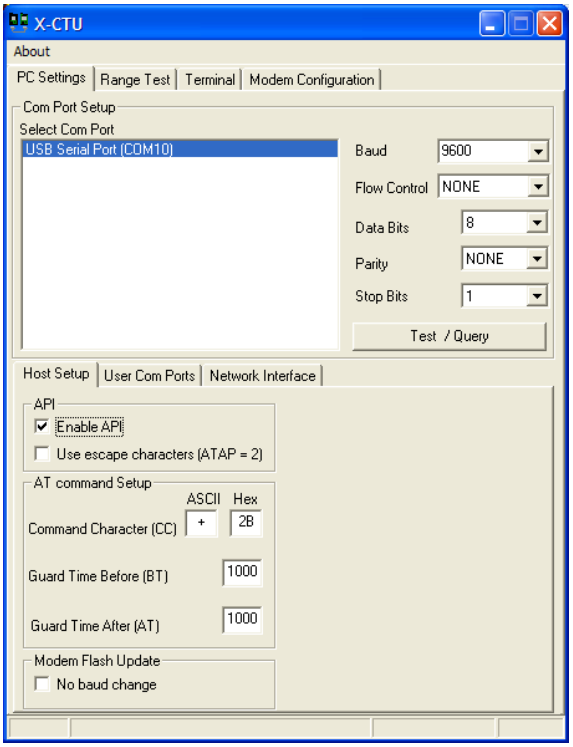


Figura 4.12- Janela principal do programa X-CTU.

Os rádios são providos de fábrica para trabalharem no modo AT, o que não convinha para o desenvolvimento do projecto, procedendo-se à alteração para modo API. Na janela principal do programa foi preciso seleccionar a porta USB, porque pode aparecer mais do que uma, e colocar um visto na opção *Enable* API. Os parâmetros de comunicação (*Baud, Flow Control Data Bits, Parity e Stop Bits*) vêm predefinidos de fábrica. Pressionou-se o botão *Test/Query* e obtém-se a Figura 4.13 a indicar o correcto acesso ao módulo.

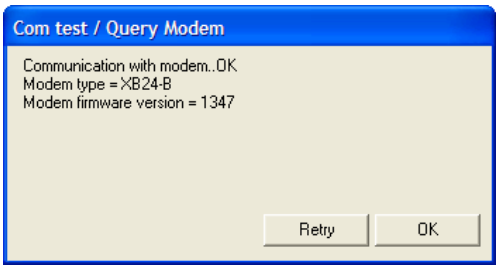


Figura 4.13- Indicação de acesso correcto ao módulo.

Após pressionar em OK, o passo seguinte foi seleccionar o menu *Modem Configuration* e ler qual o *firmware* instalado no XBee. Como são utilizados dois módulos XBee foi necessário configurar um de cada vez.

Para configurar um nó como terminal (*End Device),* foi escolhido, em primeiro lugar, no menu *Modem XBee* a versão *XB24-B* e em *Function Set* o *firmware ZNET 2.5 Router/End Device API,* como mostra a Figura 4.14.

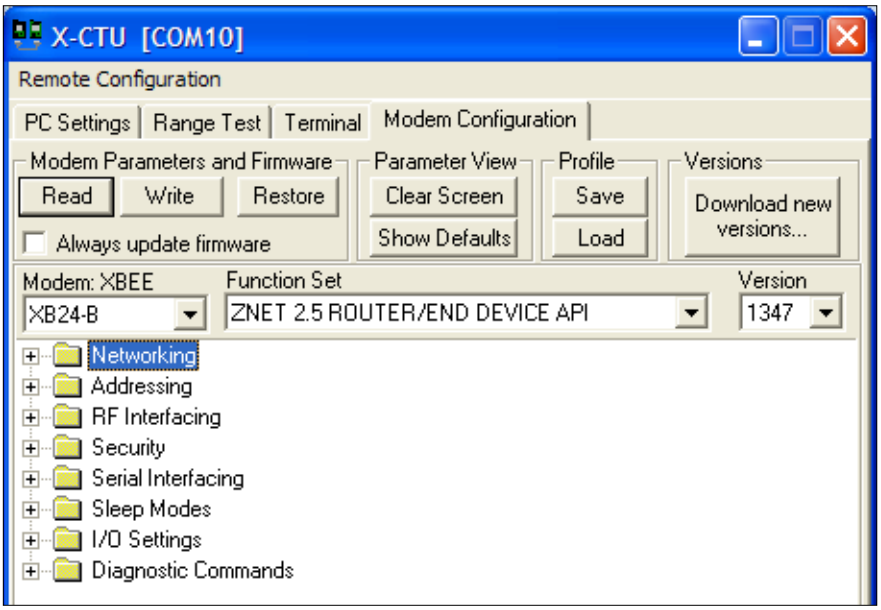


Figura 4.14- Janela com indicação de função.

Das pastas visualizadas na Figura 4.14 precisa-se da pasta *Networking*, *Addressing*, *RF Interfacing* e *Sleep Modes*. As restantes pastas têm funcionalidades que não foram utilizadas para o projecto.

Na pasta *Networking* configurou-se a rede com o campo *PAN ID* igual a 99, sendo um valor atribuído a gosto do projectista, e *Channel Verification* a 1. A *PAN ID* é a identificação do grupo de trabalho e *Channel Verification* serve para garantir que caso haja uma alteração no canal de funcionamento, *routers* e *end-devices* passam a trabalhar nesse novo canal.

Na pasta *Addressing* existe um campo denominado por *Node Identifier* o qual é configurado com um nome atribuído pelo projectista, facilitando desta feita o manuseamento com os nós.

A pasta *RF Interfacing* trata dos parâmetros de potência de emissão e sensibilidade do rádio. De referir que a potência de sinal (*Power Level*) está configurada para uma saída de +3dBm o que corresponde à opção 4, e o *Power Mode* está a 1 o que implica um melhoramento na sensibilidade em 1 dB e um aumento de 2 dB na potência de saída fazendo com que o alcance seja superior.

A distinção entre *router* e *end-device* é efectuada na pasta *Sleep Modes*, bastando para isso alterar o campo *SM* para 1 caso seja *end-device* ou deixar a 0 caso seja *router*.

As opções efectuadas são visíveis na Figura 4.15.

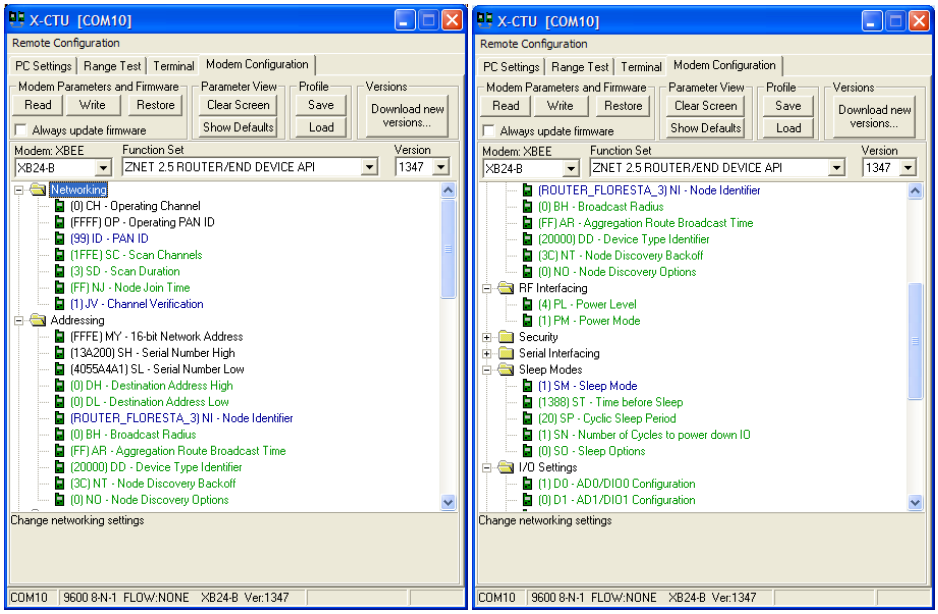


Figura 4.15- Janela de opções de configuração.

Após escolher os parâmetros, gravam-se as alterações pressionando o botão *Write.*

Para configurar o rádio como coordenador (*Coordinator)* foi escolhido no menu *Modem XBee* a versão *XB24-B* e em *Function Set* o *firmware ZNET 2.5 Coordinator API*. O coordenador apenas precisa de estar na mesma PAN ID e foi definido no campo *Node Identifier* como COORDENADOR. As alterações podem ser visualizadas na Figura 4.16.

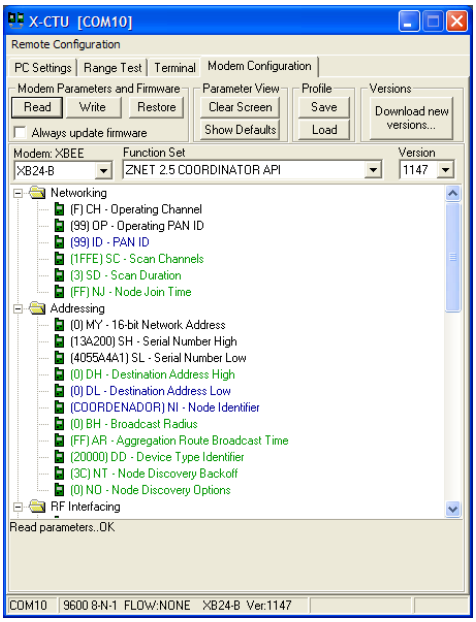


Figura 4.16 - Janela de configuração do Coordenador.

Após as configurações estarem efectuadas, é fundamental que o *end-device* ou *router* seja ligado após o coordenador, garantindo, desta forma, que fiquem associados ao canal em que o coordenador está a operar.

Para verificar se a rede está a funcionar basta fazer um *reboot* ao *end-device.* Quando este associa-se é apresentada uma mensagem de "*Hello*". Para visualizar esta

mensagem o programa X-CTU deve estar com o menu *Terminal* aberto e a opção *Show Hex* activa. Na Figura 4.17 é apresentada a mensagem de "*Hello*" do router Floresta_3.

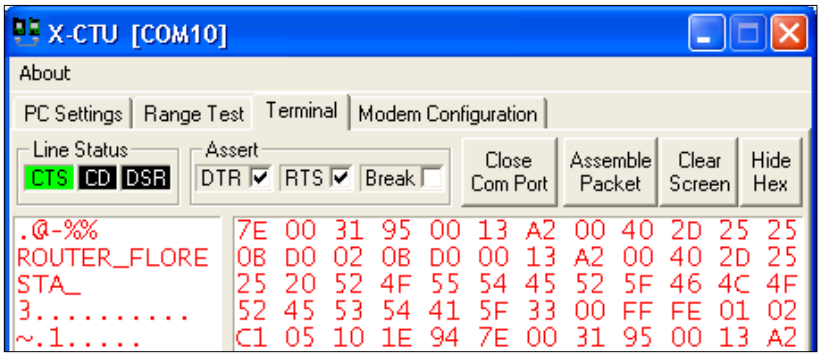


Figura 4.17 - Mensagem de "*Hello*" do router_floresta_3.

No Anexo C verifica-se facilmente que esta mensagem é do tipo *Node ID Indicator* porque é caracterizada pelo valor hexadecimal 95 no quarto byte.

Ainda no menu *Terminal* é possível o envio de comandos em formato API. Pressionando em *Assemble Packet* surge uma janela e é preciso seleccionar a opção *HEX*, o que implica que toda a mensagem é escrita em hexadecimal. Existem vários tipos de comandos que podem ser consultados na folha de características do rádio XBee, nomeadamente comandos especiais, endereçamento, rede, segurança, interface rádiofrequência, interface série (entrada/saída) e diagnóstico.

No exemplo da Figura 4.18 será enviado um comando de rede em formato hexadecimal que segue a estrutura de uma mensagem API.

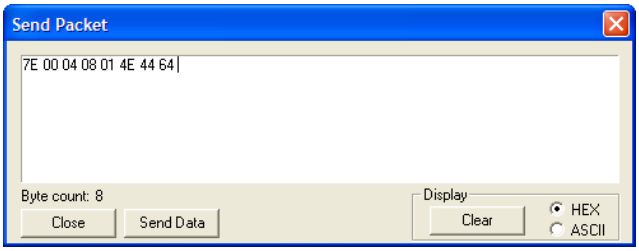


Figura 4.18 – Envio de mensagem com o comando ND

Inicializa-se por **7E,** seguindo-se os bytes referentes ao comprimento que neste caso é **04.**

O campo *API Ident*. identifica que tipo de mensagem será enviada. Na tabela em Anexo C pode visualizar-se que a mensagem do tipo **08** corresponde a *AT COMMAND*. O byte da *Frame ID* precisa de estar a **01** para que o coordenador receba um retorno da mensagem.

Em *Command Name* é necessário recorrer a uma tabela *ASCII* para converter o comando *ND*, em que N corresponde a **4E** e D corresponde a **44**. Este comando serve para descobrir quais os nós presentes na rede.

O valor do *CheckSum* é calculado por

$$CheckSum = [FF - (API\ Ident. + Frame\ ID + \cdots + Command\ Name)]$$
$$CheckSum = [FF - (08 + 01 + 4E + 44)] = \mathbf{64} \quad (4.1)$$

o que consiste em subtrair a FF a soma dos bytes a partir de *API Ident*. até *Command Name*.

Na Figura 4.19 afere-se o retorno do envio do comando ND, o qual contém duas mensagens provenientes dos *routers* Floresta_3 e Floresta_4.

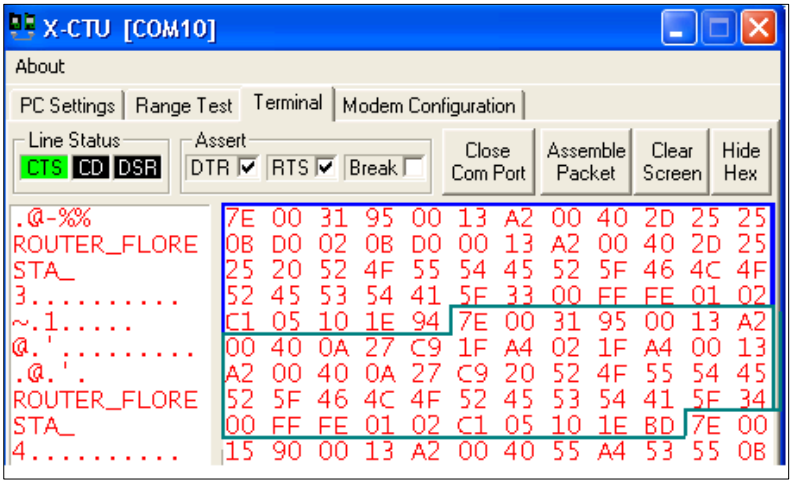
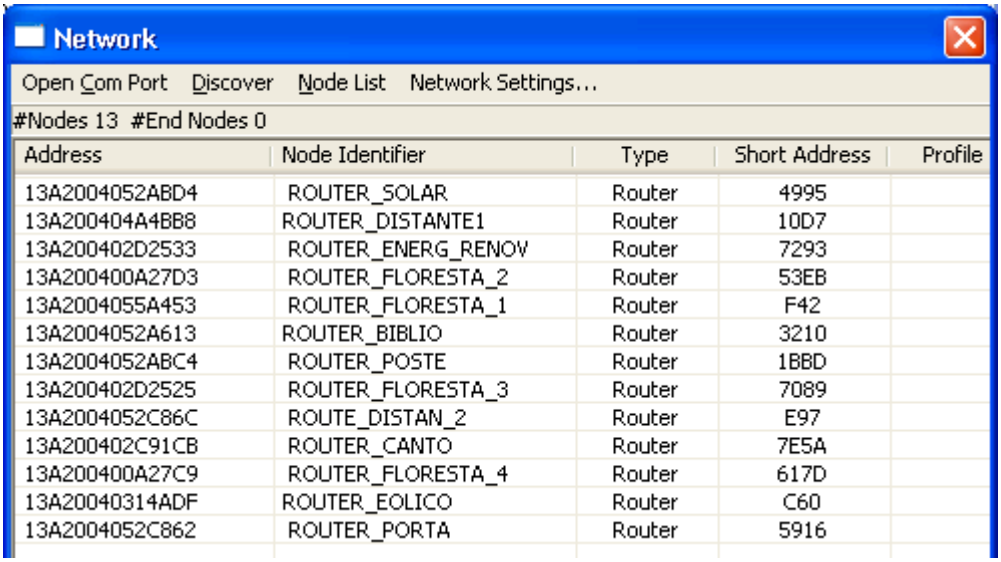


Figura 4.19 – Mensagens de retorno do comando ND

De uma forma mais cómodo, a versão 5.1.4.1 do *software* X-CTU contém uma facilidade que não só permite visualizar quais os nós presentes na rede como também uma possível configuração remota. Esta facilidade encontra-se no menu *Modem Configuration* com a designação *Remote Configuration*, visível na Figura 4.20. De mencionar que a configuração remota dos nós deve ser efectuada com precaução no que diz respeito aos parâmetros que sejam cruciais ao estabelecimento da rede, sendo um deles o *PAN ID*.



Address	Node Identifier	Type	Short Address	Profile
13A2004052ABD4	ROUTER_SOLAR	Router	4995	
13A200404A4BB8	ROUTER_DISTANTE1	Router	10D7	
13A200402D2533	ROUTER_ENERG_RENOV	Router	7293	
13A200400A27D3	ROUTER_FLORESTA_2	Router	53EB	
13A2004055A453	ROUTER_FLORESTA_1	Router	F42	
13A2004052A613	ROUTER_BIBLIO	Router	3210	
13A2004052ABC4	ROUTER_POSTE	Router	1BBD	
13A200402D2525	ROUTER_FLORESTA_3	Router	7089	
13A2004052C86C	ROUTE_DISTAN_2	Router	E97	
13A200402C91CB	ROUTER_CANTO	Router	7E5A	
13A200400A27C9	ROUTER_FLORESTA_4	Router	617D	
13A20040314ADF	ROUTER_EOLICO	Router	C60	
13A2004052C862	ROUTER_PORTA	Router	5916	

Figura 4.20 – Janela de configuração remota

4.4.1 RSSI

O RSSI é uma medida de desempenho do sistema de comunicações sem fios que serve para obter o valor da potência recebida.

Este valor é obtido a partir do nó coordenador enviando uma linha de comandos em estrutura API, utilizando o software X-CTU.

Antes de se proceder ao envio da mensagem é necessário determinar alguns parâmetros para que a mensagem seja correctamente enviada. Na Figura 4.21 visualiza-se o que tem de conter a estrutura da mensagem do comando a ser utilizado para obter o valor de RSSI, sendo neste caso (*Remote Command Request*).

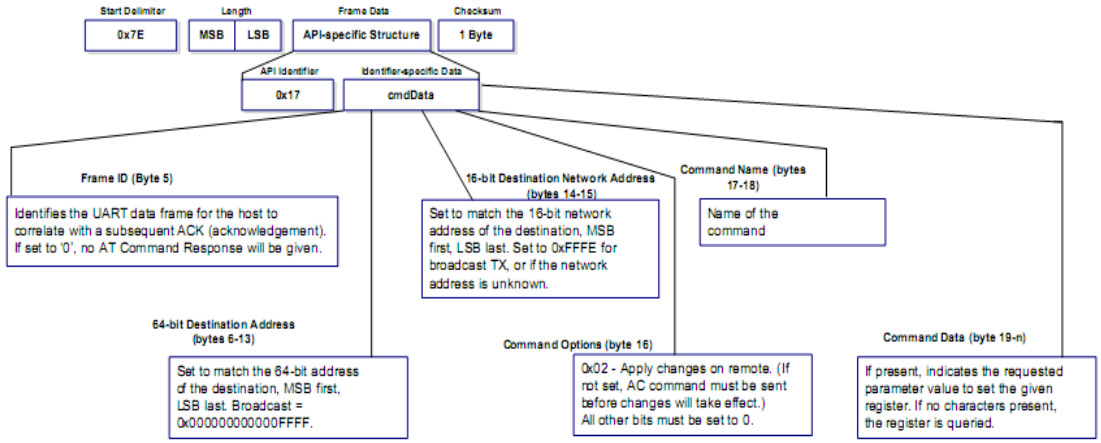


Figura 4.21- Estrutura específica do módulo API[48].

.Na tabela em Anexo C pode visualizar-se que a mensagem do tipo 17 corresponde a *Remote command Request*. O próximo campo após *Frame ID* é referente à identificação do nó em questão (numero de série), em que os primeiros 4 bytes correspondem ao SH (*Serial Number High*) e os restantes ao SL (*Serial Number Low*). Estes valores são obtidos à partida antes de se proceder ao envio da mensagem. Colocando o campo 16 Dest. Network a FF FE, implica fazer um *Broadcast* pela rede. Em *Command Name* é enviada um comando de interface radiofrequência com o atributo DB, o qual tem de ser convertido para hexadecimal, seguindo-se por fim o checksum. A Tabela **4.3** mostra a estrutura da mensagem enviada.

Tabela 4.3- Estrutura da mensagem enviada.

Start Delim.	Length	API Ident.	Frame ID	64 Dest. Address	16 Dest. Network	Options Command	Command name	Cheksum
7E	00 0F	17	01	00 13 A2 00 40 2C 91 CE	FF FE	01	44 42	E3

A leitura do RSSI, deve ser efectuada após o envio de várias mensagens de modo a estimar um valor mais aproximado. A mensagem de retorno tem uma forma semelhante à anterior, mas desta feita trata-se de uma mensagem do tipo *Remote Command Response*, porque é atribuído ao campo *API Identifier* o hexadecimal 0x97. A Tabela 4.4 mostra a estrutura da mensagem recebida.

Tabela 4.4- Estrutura da mensagem recebida.

Start Delim.	Length	API Ident.	Frame ID	64 Dest. Address	16 Dest. Network	Command name	Status	Request Value	Cheksum
7E	00 10	97	01	00 13 A2 00 40 2C 91 CE	7B C9	44 42	00	41	DC

Da mensagem recebida o campo de interesse é o *Request Value*. Este valor é convertido para decimal e indica o valor de RSSI em dBm. No exemplo em questão o valor obtido é de -65 dBm ($41_{16} \rightarrow 65_{10}$).

Esta facilidade é de extrema importância no momento da implantação dos dados, porque à medida que os nós sensores são colocados é possível obter o valor de RSSI referente ao seu vizinho.

4.5 Ferramentas de *software*

Após a construção do nó sensor e configuração da rede *ZigBee* procede-se à configuração do ATMega168. A programação do microcontrolador é efectuada utilizando-se o *software* AVR *Studio* juntamente com o compilador *Win AVR*. Durante o arranque do *software* é aberta uma janela onde é definido que tipo de projecto será desenvolvido, o nome do projecto e a localização das pastas subjacentes ao mesmo. Numa segunda janela é pedido para escolher qual a plataforma para *debugging* e o microcontrolador em questão.

A Figura 4.22 exemplifica os passos supracitados.

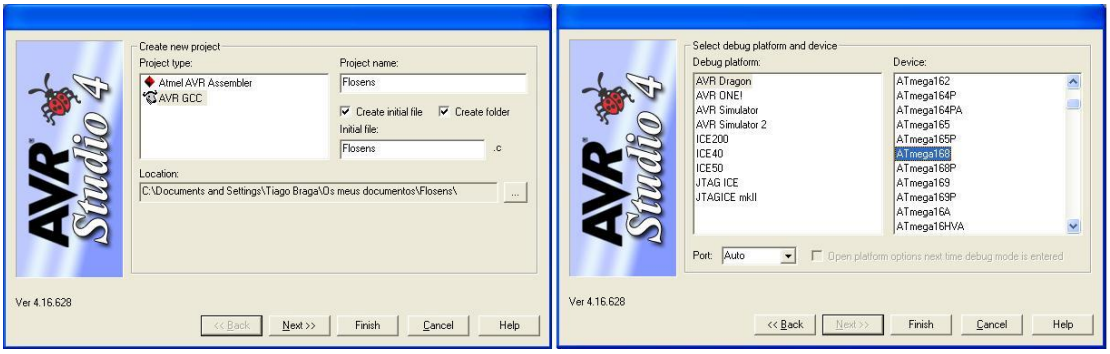


Figura 4.22 - Janelas iniciais do AVR Studio

Concluído este processo dá-se início à programação desenvolvida em linguagem C. A programação consiste em diversos aspectos. Inicialmente são carregadas as bibliotecas e as configurações das portas, funções auxiliares, variáveis globais e apontadores das funções. No programa principal são definidos os estados das portas, o arranque da UART e RTC. É inicializado um ciclo com várias secções as quais dependem de variáveis de tempo e de variáveis auxiliares.

A primeira secção consiste em adormecer o XBee e o microcontrolador. Na segunda secção tanto o XBee como o microcontrolador acordam após uma interrupção temporal e esperam um determinado tempo para que o XBee envie a mensagem de amostra. Após o envio, o XBee fica no estado *Idle* à espera de receber uma mensagem de *broadcast* que contém os tempos relativos para o próximo ciclo, sendo esta secção definida como a

terceira. Na última secção, o XBee após ter recebido o *broadcast* informa o microcontrolador dos novos tempos e o nó adormece.

O fluxograma da Figura 4.23 serve para uma melhor compreensão do programa desenvolvido.

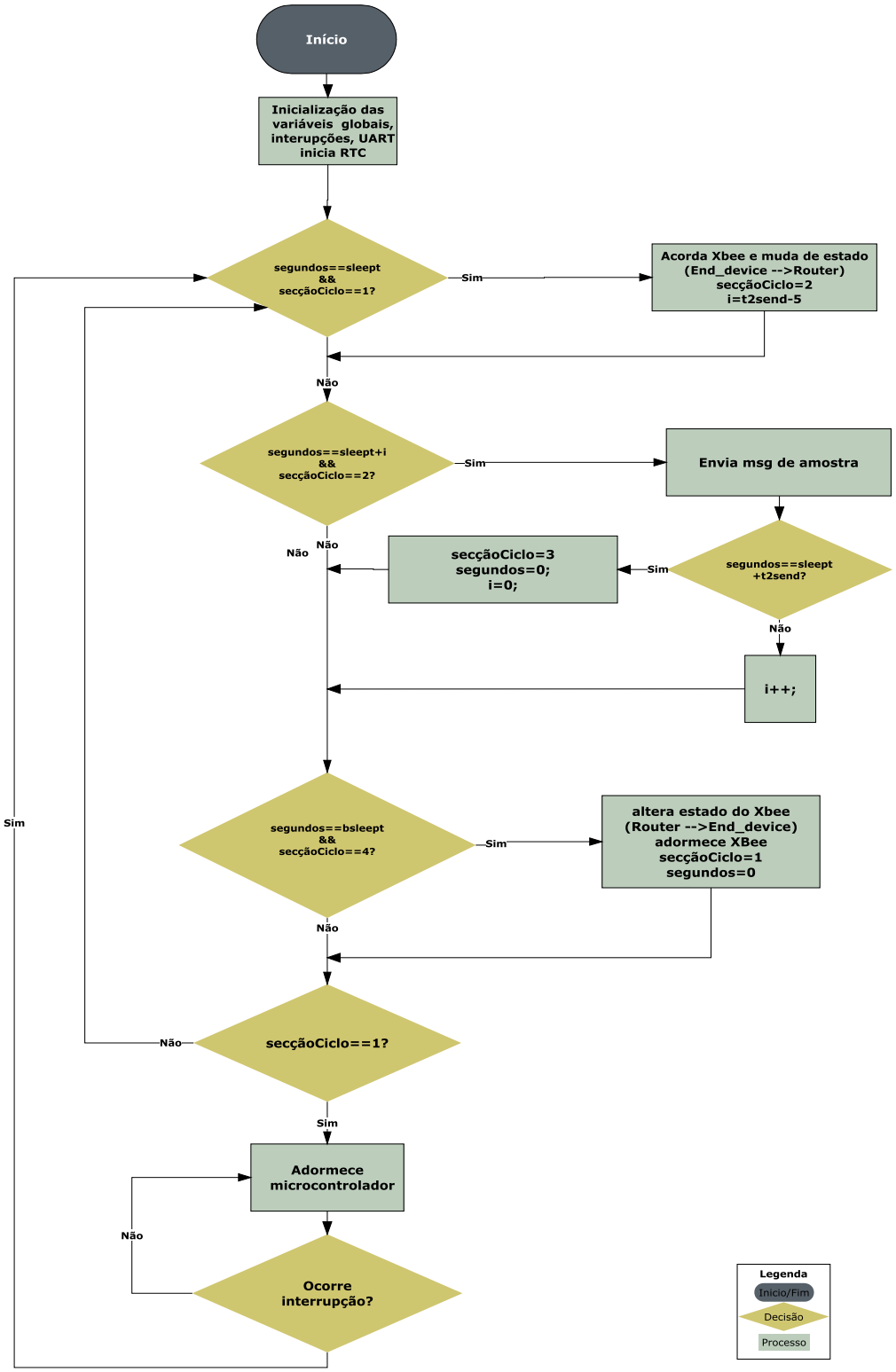


Figura 4.23 - Fluxograma do programa principal.

Na segunda secção é enviada a mensagem de amostra. Esta mensagem consiste em obter os vários valores dos sensores e o valor da tensão das baterias. O primeiro passo consiste em abrir o relé do circuito On_Off para que a tensão amostrada não seja a proveniente do painel solar. O MOSFET é activado para que haja corrente a percorrer o divisor resistivo e desta feita amostrar a tensão nas baterias. Para que não haja consumo de corrente desnecessário o MOSFET volta para o estado inicial, ou seja, aberto.

A tensão depois de ser amostrada passa por um circuito de decisão que consiste, basicamente, em comutar a posição do relé consoante o valor amostrado. Este processo serve para maximizar a vida útil da bateria evitando sobrecargas proveniente do painel solar.

Segue-se a amostragem dos sensores de luminosidade, temperatura e humidade. A mensagem é finalmente compilada e, após o cálculo do *checksum*, é enviada para a rede. O fluxograma da Figura 4.24 exibe os processos mencionados.

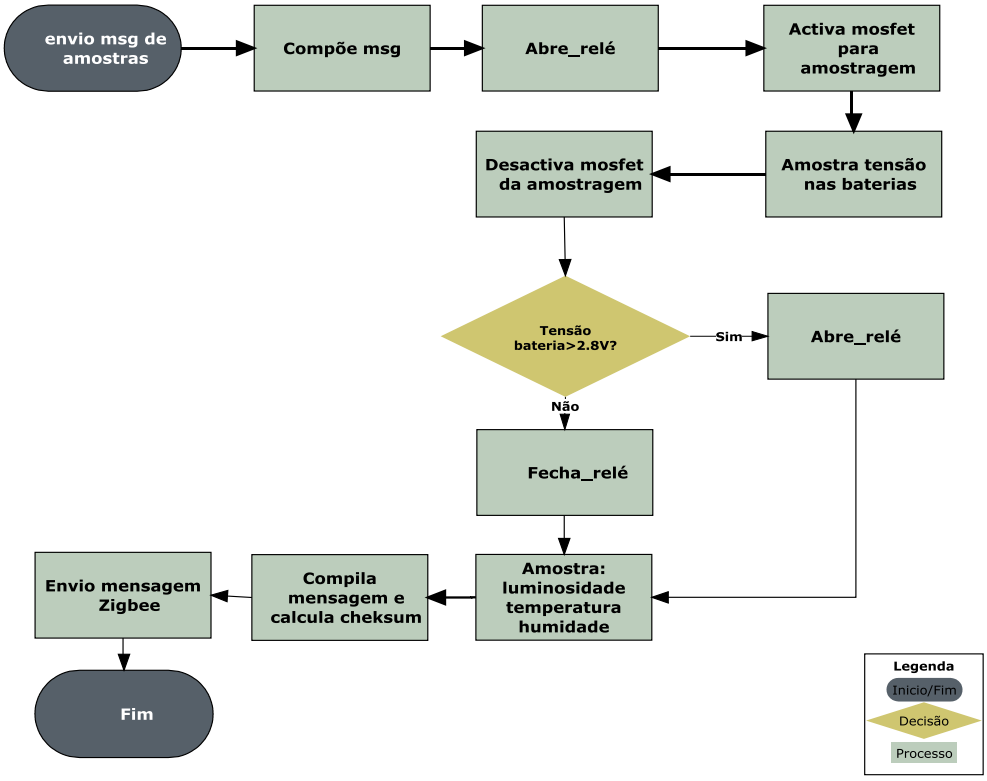


Figura 4.24 - Fluxograma da amostragem

Foram definidas inicialmente variáveis que servem para o ciclo de operação que são: *sleept*, *t2send* e *bsleept*. No fluxograma da Figura 4.23 verifica-se que estas variáveis estão entre os vários blocos de decisão. Para atribuir uma certa dinâmica aos tempos previamente definidos no microcontrolador, foi desenvolvido um tratamento para uma mensagem de *broadcast* proveniente do coordenador. O tratamento desta mensagem é efectuado *byte* por *byte*, ou seja, cada *byte* é tratado individualmente com recurso à variável *letraaux* e é-lhe atribuído uma posição utilizando a variável *xbeei*. À medida que a variável *xbeii* é incrementada, esta passa por várias decisões até que alcança a condição em que *syncx=3*. Quando o ciclo alcançar os *bytes* nas posições 17 18 e 19 são atribuídos

novos tempos para o ciclo de operação finalizando assim o tratamento da mensagem. O fluxograma que expõe o tratamento da mensagem encontra-se em Anexo I.

O código desenvolvido em linguagem C pode ser visualizado no Anexo J. O código após estar compilado é carregado no microcontrolador recorrendo ao emulador AVR-ISP500 como já mencionado.

4.6 Tratamento e Armazenamento dos dados

Cada nó envia as suas mensagens com destino ao coordenador, o qual encaminha pela UART até ao computador o que finalmente pode ser visualizado no X-CTU. Como este programa só apresenta as mensagens temporariamente foi necessário desenvolver uma aplicação responsável por receber, interpretar, condicionar e armazenar as mensagens provenientes dos vários nós numa base de dados.

O programa *Netbeans IDE 6.7* foi o programa de eleição para a elaboração desta aplicação. Foram definidas várias classes que têm variadas funções, desde comunicação com a porta série, manuseamento das mensagens com diferentes identificadores API, incluindo filtragem de mensagens indesejadas, envio de uma mensagem de *broadcast* não só para o envio dos tempos de adormecimento como também para o pedido de RSSI e armazenamento na base de dados. O Anexo K contém o código desenvolvido na criação das classes. Para que os dados sejam armazenados, é requisito fundamental que se active um serviço de base de dados, sendo neste projecto utilizado o *MySQL*[49]. O diagrama de entidade -relacão , visível na Figura 4.25, pretende ilustrar como está estruturada a base de dados. Esta apresenta três tabelas designadas por **amostras**, **valor_rssi** e **no.** Na tabela **amostras** ficam armazenadas os valores das amostras juntamente com a identificação do nó em questão, a data e hora de entrada. A tabela **valor_rssi** contém o valor de RSSI juntamente com a identificação do nó sensor e novamente com a data e hora. Na tabela **no** será guardado a identificação do nó através do campo 64address proveniente dos endereços SH (*Serial Number High*) e SL (*Serial Number Low*) do XBee. Como este endereço é extenso (5526146520361940) decidiu-se atribuir um campo towerID em que a cada nó sensor é atribuído um valor compreendido entre 15 a 24. Os restantes atributos na tabela **no** (lat, lng, cor, tipo) servem para o desenvolvimento da página de Internet.

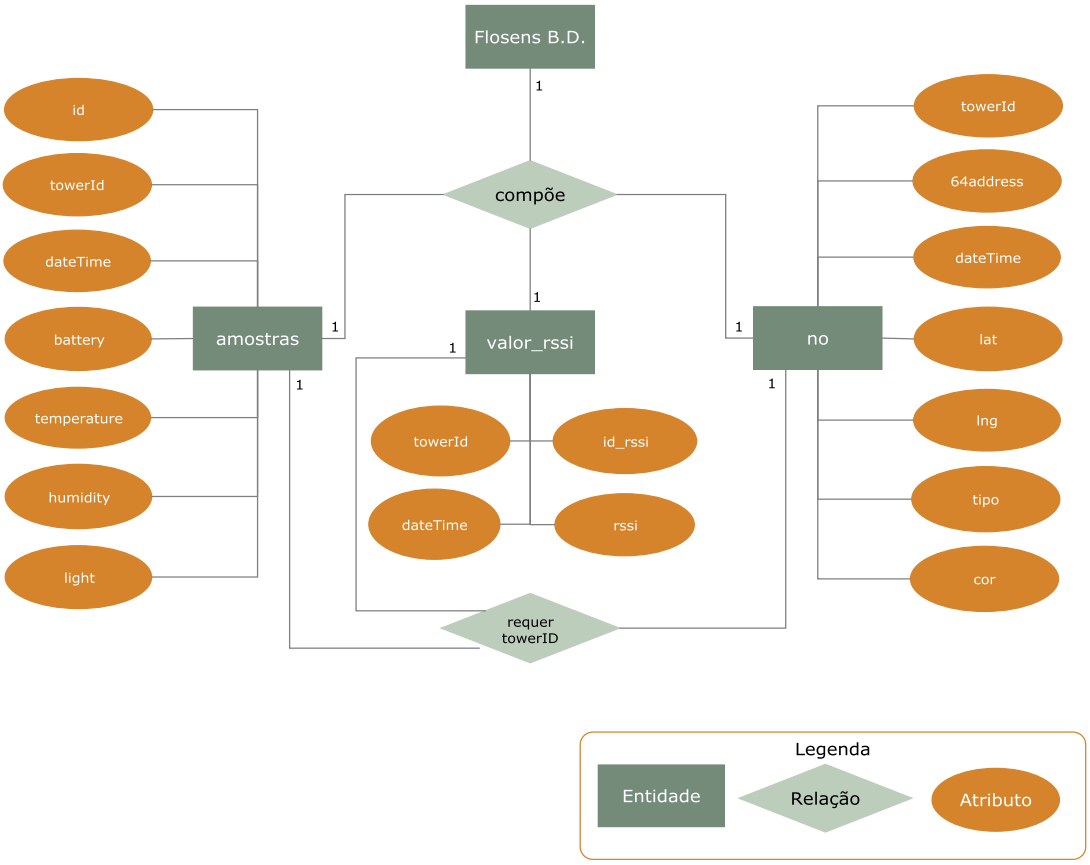


Figura 4.25 - DER presente na base de dados

4.7 Visualização dos dados

Um dos objectivos deste projecto também consistiu na apresentação dos valores obtidos pelos nós sensores numa interface gráfica em tempo real, utilizando uma página de *Internet*. Inicialmente foi instalado o pacote *XAMPP*[50], o qual contém um servidor *Apache* que serve como servidor de Internet, um motor *PHP* e um sistema de base de dados *MySQL*, os quais são fundamentais para a elaboração da página Web O diagrama de fluxos de dados referente à aplicação em questão pode ser visualizada na Figura 4.26.

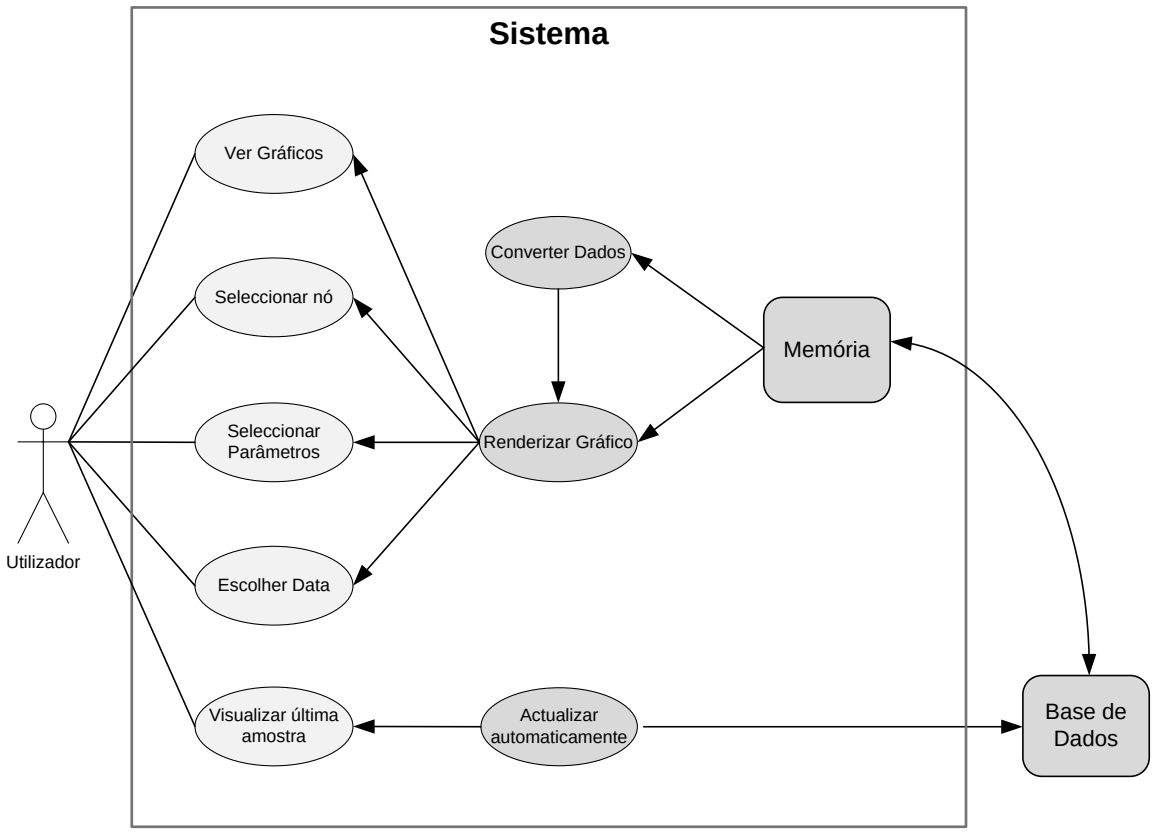


Figura 4.26 – Diagrama de fluxo de dados.

A aplicação foi desenvolvida em linguagem *PHP* utilizando o editor *Notepad*++[51]. Na Figura 4.27 visualiza-se o interface desenvolvido. Do lado esquerdo da página situa-se um mapa com a localização geográfica dos nós. Ao passar o rato em cima de um dos nós surge uma janela indicando os últimos valores obtidos da temperatura, humidade, luminosidade, tensão nas baterias, valor de *RSSI* e no fim do quadro a data e hora da última amostragem. Do lado direito pode-se efectuar uma pesquisa por data, seleccionar um ou mais nós usando a tecla *CTRL* e seleccionar o sensor pretendido. Os dados são fornecidos por meio de um gráfico, o que permite efectuar uma comparação entre os vários nós de uma forma mais cómoda. A página está disponível em http://alumni.dme.uma.pt/wsngroup/index.php?op=flosens. O código em php referente a esta aplicação encontra-se no Anexo L..

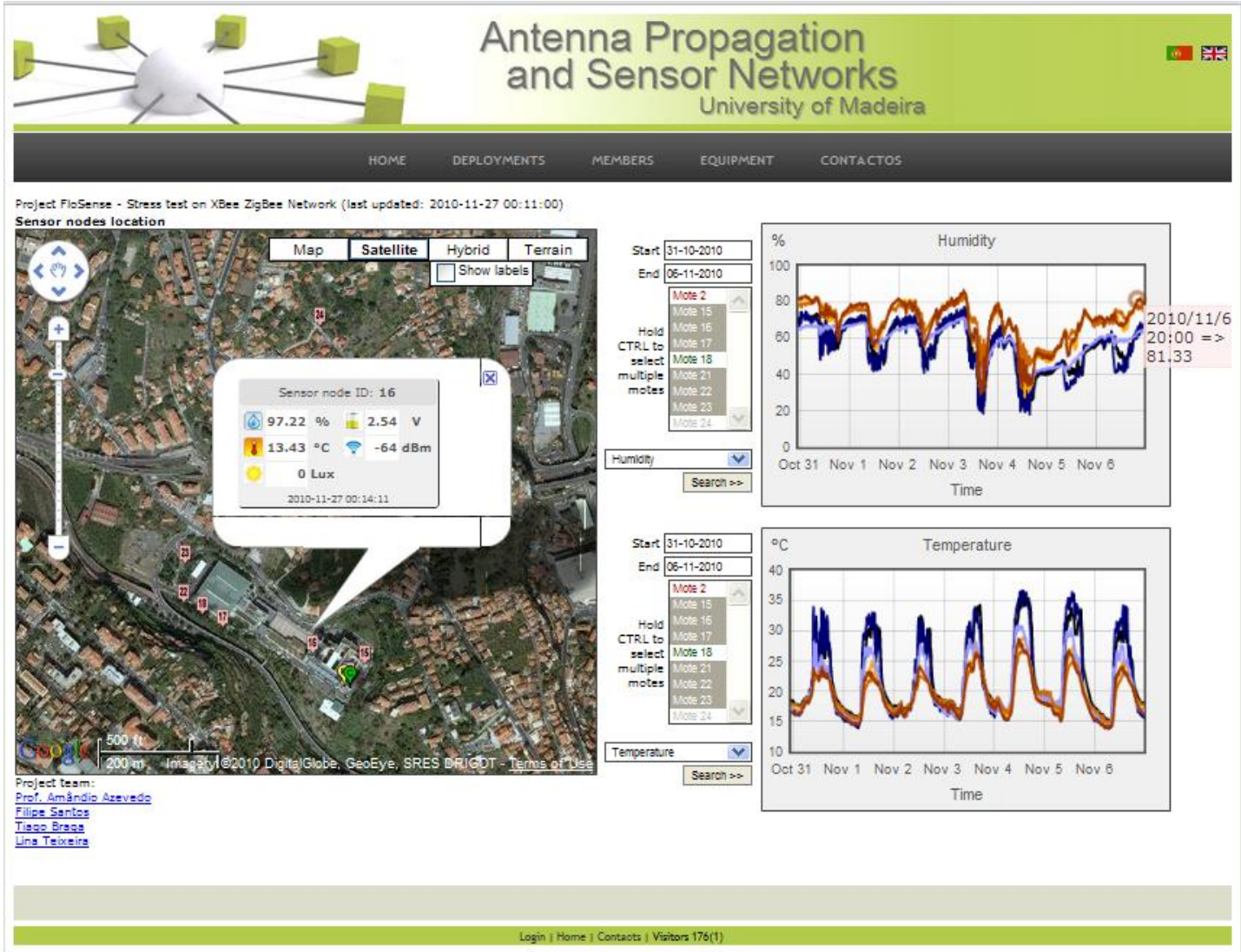


Figura 4.27 - Página Web.

4.8 Técnica de adormecimento dos routers

Como anteriormente referido esta técnica consiste na comutação entre dispositivos lógicos (*routers* e *end-device*). Para efectuar esta técnica é necessário utilizar o comando SM (*Sleep Mode*) presente no rádio XBee. Este comando é que permite distanciar um *router* de um *end-device*. O envio do comando em formato API é efectuado através da UART pelo microcontrolador e segue a seguinte estrutura. Na Tabela 4.5 e na Tabela 4.6 constam os dois tipos de mensagens que serão enviadas do microcontrolador, sendo a primeira para adormecer em que o campo *Parameter Value* está a 1 e a segunda para acordar em que o valor do *Parameter Value* passa a 0.

Tabela 4.5 - Estrutura do comando para adormecer.

Start Delim.	Length	API Ident.	Frame ID	AT Command	Parameter Value	Cheksum
7E	00 05	08	01	53 4D	01	55

Tabela 4.6 - Estrutura do comando para acordar.

Start Delim.	Length	API Ident.	Frame ID	AT Command	Parameter Value	Cheksum
7E	00 05	08	01	53 4D	00	56

A passagem de um estado para outro, ou seja, de *router* para *end-device* e vice-versa é no entanto efectuada pelo microcontrolador, o qual coloca o pino 9 (*Sleep*) no estado desejado, a *High* (1-para adormecer) e a Low (0-para acordar). Após confirmar o funcionamento correcto do algoritmo, procederam-se a testes para determinar qual o tempo mínimo necessário para o nó estabelecer associação ao coordenador e enviar a sua mensagem de amostra, concluindo-se que seriam necessários 35 segundos para efectuar todo este processo. Ainda no ATMega168 foram criadas três variáveis que correspondem a três tempos diferentes, sendo estes: *sleept*, *t2send*, *bsleept*. Estas três variáveis correspondem a tempos que serão enviados pelo coordenador através de uma mensagem de *broadcast*. O tempo *sleept* corresponde à duração em que o nó ficará a dormir, *t2send* corresponde à duração em que nó fica à espera que a rede se forme e envie mensagens e o *bsleept* é o tempo que o nó aguarda após o envio das mensagens para receber um novo *broadcast*, onde serão enviados os novos tempos do ciclo. A Figura 4.28 ilustra a duração do ciclo, onde são ilustradas as devidas comutações através de uma linha temporal.

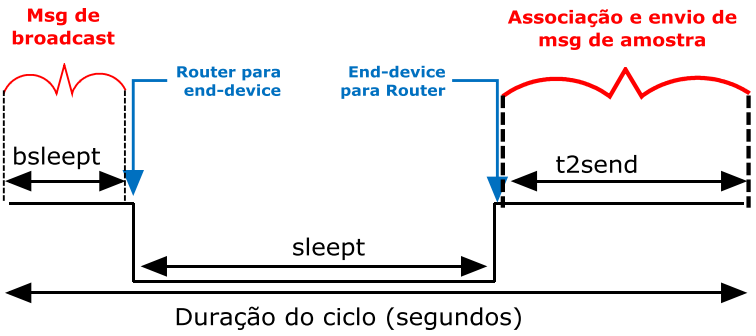


Figura 4.28 – Linha temporal de acontecimentos.

5 Testes e Resultados

Neste capítulo são apresentados os resultados obtidos pelos protótipos desenvolvidos e pela rede instalada. Recorrendo à ferramenta *Google Earth* procedeu-se à localização geográfica onde foram implantados os nós sensores.

É exposto um comparativo da temperatura, humidade, tensão na bateria, luminosidade e RSSI entre o nó sensor e o equipamento disponível em laboratório.

Seguem-se os resultados obtidos em dois ambientes distintos, ou seja, no terraço do edifício da Universidade e na num ambiente com vegetação.

Posteriormente são apresentados os resultados no que se refere à técnica de adormecimento dos *routers* em duas situações. A primeira consistiu num teste realizado à rede no interior das instalações da Universidade, em que a distância entre routers não ultrapassou os 15 metros. A segunda situação corresponde à implantação dos *routers* num ambiente exterior.

São ainda apresentados gráficos comparativos das amostras recebidas durante um período de 5 dias, nos dois ambientes. Sobre o RSSI são apresentados dois gráficos fazendo uma comparação entre os resultados obtidos dos modelos de propagação, o valor obtido pelo RSSI e pelo analisador de espectros para situações com diferentes antenas.

5.1 Distribuição geográfica dos nós sensores

A distribuição geográfica dos nós sensores, para realização de testes ao protótipo implementado, tornou-se numa tarefa árdua atendendo a que era crucial garantir que os nós comunicassem apenas com os nós vizinhos. Foram atribuídos nomes aos *routers* para uma melhor identificação ao longo do trabalho. Como o Coordenador está presente no Laboratório de Telecomunicações no piso -2 e o alcance mencionado na folha de características do rádio XBee no espaço livre em linha de vista pode alcançar 120m, fez com que o primeiro nó (Router_Canto) fosse implantado no topo do edifício, porque desta forma conseguia-se com que o sinal não alcançasse os restantes nós sem passar obrigatoriamente por este.

Como as persianas no Laboratório dificultem a propagação de sinais rádio-frequência optou-se por colar um monopolo no exterior do laboratório, para melhorar a qualidade de sinal entre o Coordenador e o Router_Canto. A Figura 5.1 ilustra a localização do monopolo no exterior das instalações juntamente com o Coordenador e a Gateway.

Figura 5.1 - Gateway conectada ao Coordenador com antena monopolo.

Na Figura 5.2 pode visualizar-se o monopolo colado no exterior do Laboratório e embora pouco visível encontra-se o Router_Canto no topo do edifício.

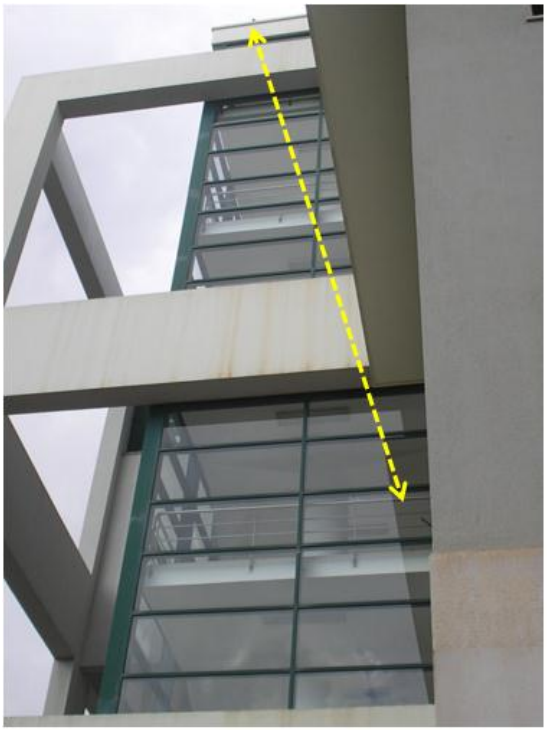


Figura 5.2 - Ligação entre Coordenador e Router_Canto.

Passando para o topo do edifício, procedeu-se da mesma forma para a colocação do Router_Porta e para os demais nó sensores. Antes de instalar cada nó sensor foram realizados testes, recorrendo à facilidade RSSI presente no rádio XBee, para verificar qual o nível de potência do sinal recebido. Este processo é efectuado através de um envio de uma mensagem em formato API como mencionado no ponto (4.4.1). Após verificar o nível do sinal, o nó é finalmente instalado e dá-se início à monitorização.

A distribuição dos nós sensores foi efectuada em redor da Universidade da Madeira, à excepção de dois nós sensores. Utilizando a ferramenta *Google Earth*, é possível visualizar na Figura 5.3 a distribuição geográfica dos nós juntamente com a indicação das distâncias entre estes. Foram atribuídos nomes aos *routers* para uma melhor identificação ao longo do trabalho. Embora não seja visível na Figura 5.3 a distância (altura) entre o Coordenador e o Router_Canto é de aproximadamente 25 metros.

Segue-se o Router_Porta a uma distância de 40 m do Router_Canto. Do Router_Porta para o Router_Biblioteca vai uma distância de 87 m. O Router_Distante1 encontra-se a uma distância de 356 m e a sua ligação é efectuada recorrendo a uma antena grelha. O Router_Distante2 encontra-se a 228 m do Router_Distante1 e é utilizada, também, uma antena grelha. Como a sua colocação foi efectuada em postos de electricidade presentes na rua, optou-se por remover o painel solar para não atrair a atenção de curiosos e colocar o nó a uma altura o menos acessível possível. Do Router_Biblioteca para o Router_Floresta1 vai uma distância de 177 m. A distância entre os "Routers_Floresta_n" é bastante mais reduzida devendo-se ao facto de não haver linha de vista entre eles e estarem implantados numa zona florestal.

Figura 5.3- Distribuição geográfica dos nós sensores

Outro dos aspectos relevantes foi a sua fixação, no que se refere aos nós presentes no ambiente florestal. De modo a não "ferir" as árvores optou-se por fixar inicialmente o nó sensor a uma régua de madeira e este posteriormente seria amarrado à árvore como se pode visualizar na Figura 5.4. Os *routers* com antenas grelha foram amarrados recorrendo a abraçadeiras plásticas.

Figura 5.4 - Nós sensores implantados

5.2 Testes comparativos do nó sensor

Atendendo a que foi necessário obter cinco parâmetros de interesse recorreu-se a cinco equipamentos diferentes, visíveis na Figura 5.5. O multímetro *Fluke 111* teve como função medir o valor da tensão nas baterias, o *Silva Pro ADC* serviu como higrómetro para leitura da humidade relativa do ar, o analisador de espectros *RS_FSH3* para obter o valor de RSSI, o luxímetro *Lutron LX-101* para obter o valor da luminosidade e por fim o multímetro *Fluke 87V* que teve a funcionalidade de trabalhar como termómetro associado a um termopar do tipo K.

Figura 5.5 – Equipamentos utilizados: (a) – Multimetro Fluke 111 (b) –Multi-Funcional Silva PRO_ADC, (c) –Analisador de espectros RS_FSH3, (d) Luxímetro Lx-101 da Lutron, (e) – Multimetro 87V ao qual é acoplado a sonda de temperatura 80BK-A [52][53][54][55][56].

A gama de operação, resolução e a precisão de cada instrumento são apresentadas na Tabela 5.1.

Tabela 5.1 - Características dos equipamentos

Equipamento	Gama de operação	Resolução	Precisão
Fluke 87	-200° C a 1090°C	0,1°C	2%
Fluke 111	0 a 600 V	1 mV máx	± (0,7 % + 2)
Silva ADC - Pro	20 a 80 %	0,1 %	20 a 80 → ± 3,5 % outros → ± 5%
Lutron LX - 101	2000 Lux 200 000 Lux 50 000 Lux	1 Lux 10 Lux 100 Lux	± 5% + 2d ± 5% + 2d ± 5% + 2d
RS_FSH3	100 kHz a 3 GHz	1 Hz	< 1,5 dB Típico: 0,5 dB ± 0,5

De modo a verificar o correcto funcionamento do nó sensor procedeu-se à comparação dos valores obtidos das amostras face aos equipamentos disponíveis em laboratório. Foram efectuados três tipos de testes.

O primeiro foi efectuado no interior do Laboratório onde é possível alterar com facilidade os parâmetros a serem monitorizados.

O segundo teste consistiu na confirmação dos valores dos sensores de humidade, temperatura e luminosidade e tensão nas baterias no terraço da Universidade. Este teste foi elaborado para confirmar se os valores recebidos diferem consideravelmente do amostrado.

O último teste de confirmação de valores foi efectuado no interior da vegetação.

Os testes efectuados em laboratório e no exterior tiveram a duração de uma hora e as amostras foram enviadas a cada dois minutos.

5.2.1 Testes em Laboratório

O primeiro campo a ser comparado foi o da tensão nas baterias. Como o módulo foi testado no laboratório optou-se por utilizar uma fonte de alimentação em vez de um par de baterias. Na Figura 5.6 apresenta-se o gráfico comparativo dos valores da tensão medidos. O valor da fonte de alimentação foi alterado à medida que se ia efectuado o teste, estas alterações decorrem aos 10, 20 e 50 minutos. Verifica-se que o valor obtido pelo nó sensor é ligeiramente diferente do obtido pelo voltímetro mas bastante aceitável.

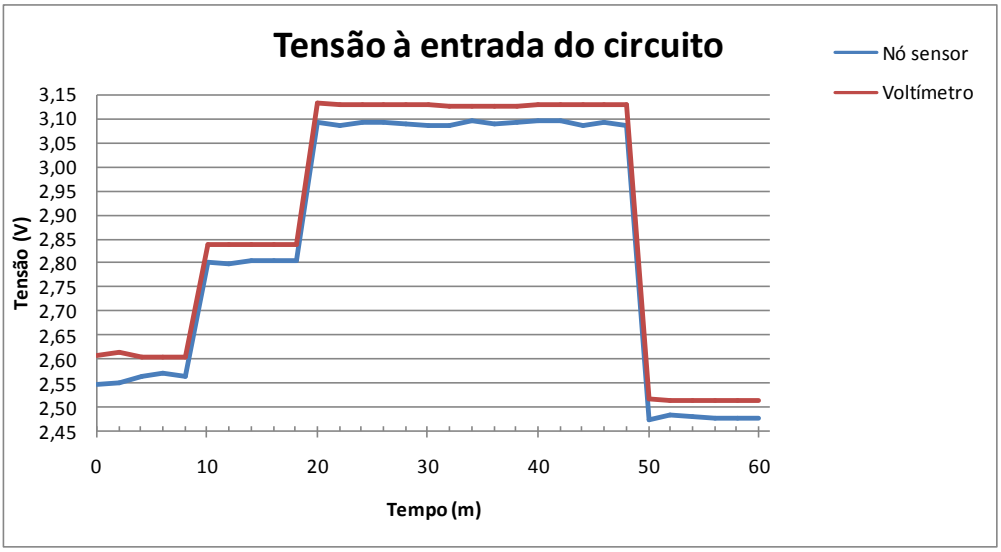


Figura 5.6 – Comparação do valor da tensão.

O segundo campo a ser comparado foi a da temperatura. Aproximando ao máximo possível a sonda de temperatura do multímetro *Fluke 87V* ao sensor SHT15 obtiveram-se os resultados obtidos na Figura 5.7. Nos primeiros 10 minutos do teste ambos registaram a temperatura ambiente presente na laboratório. Dos 10 aos 30 minutos manteve-se uma lâmpada incandescente acesa a uma distância de 10 cm do sensor SHT15 e da sonda registando desta feita um aumento na temperatura. Entre os 30 e os 50 minutos a lâmpada manteve-se apagada, registando uma diminuição no valor da temperatura. A partir dos 50 minutos até ao fim, a lâmpada foi mantida acesa na mesma posição e bem próxima (5 cm) do SHT15 e da sonda, verificando-se um aumento dos 23ºC para aproximadamente 50ºC em apenas 10 minutos. Outro facto constatado é que à medida que a temperatura ambiente aumenta começa a haver uma discrepância entre os valores obtidos pelo nó sensor e o termómetro. Esta diferença de temperatura ocorre porque o sensor SHT15 é exposto a uma fonte de calor, o que faz com que haja uma alteração no seu correcto funcionamento.

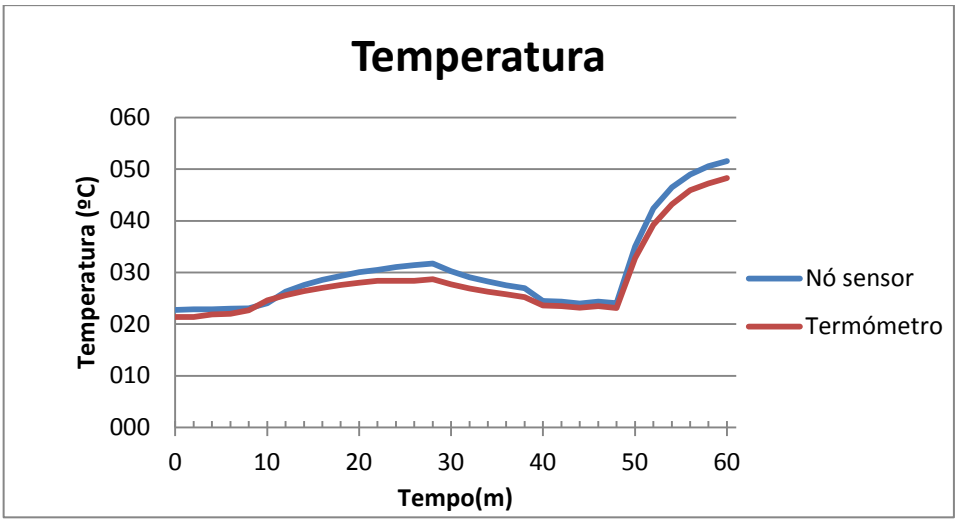


Figura 5.7 – Comparação da temperatura obtida pelo nó sensor com o termómetro.

Para comparar a humidade relativa do ar amostrada pelo sensor SHT15 foi utilizado um higrómetro presente no equipamento Silva ADC-PRO. A Figura 5.8 ilustra a comparação. O procedimento para alterar o valor da humidade foi o descrito anteriormente aquando da situação para a temperatura. Inicialmente ambos apresentam valores de humidade muito próximos, mas a partir do momento em que o sensor é exposto à fonte de calor (lâmpada) começam a surgir diferenças no momento de amostragem. Pelos gráficos presentes na Figura 5.8 nota-se que o gráfico referente ao higrómetro está ligeiramente desviado para à direita quando comparado com o do nó sensor. Ao comparar o gráfico da humidade com o da temperatura verifica-se que existe uma tendência contrária entre eles, ou seja, quando existe um aumento do valor da temperatura existe uma diminuição do valor da humidade e vice-versa.

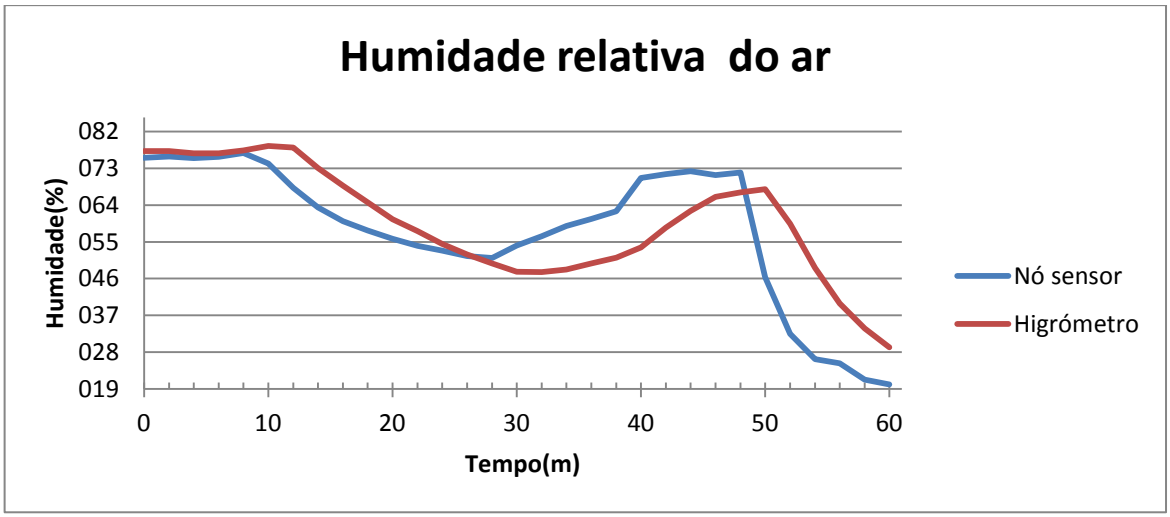


Figura 5.8 – Comparação da humidade obtida pelo nó sensor com o higrómetro.

Outro dos testes que foram efectuados é o da comparação do sensor s1087com o luxímetro Lx-101. Na Figura 5.9 apresenta-se o gráfico comparativo entre os dois. De referir que os valores das amostras do nó sensor aproximam-se dos valores obtidos pelo luxímetro. Para registar alterações no valor da luminosidade, tanto o sensor como o luxímetro estavam

paralelamente colocados entre si com uma lâmpada a servir de fonte luminosa. O teste iniciou-se com a lâmpada ligada a uma distância de 15 cm dos "sensores". Aos 10 minutos verifica-se uma alteração nos valores registados, momento em que a distância da lâmpada aos sensores foi reduzida para 10 cm. Aos 20 minutos voltou-se a colocar a lâmpada na situação inicial, mas registaram-se valores ligeiramente diferentes. Aos 40 minutos a distância da lâmpada aos sensores é reduzida para 5cm, verificando-se um aumento no valor da luminosidade.

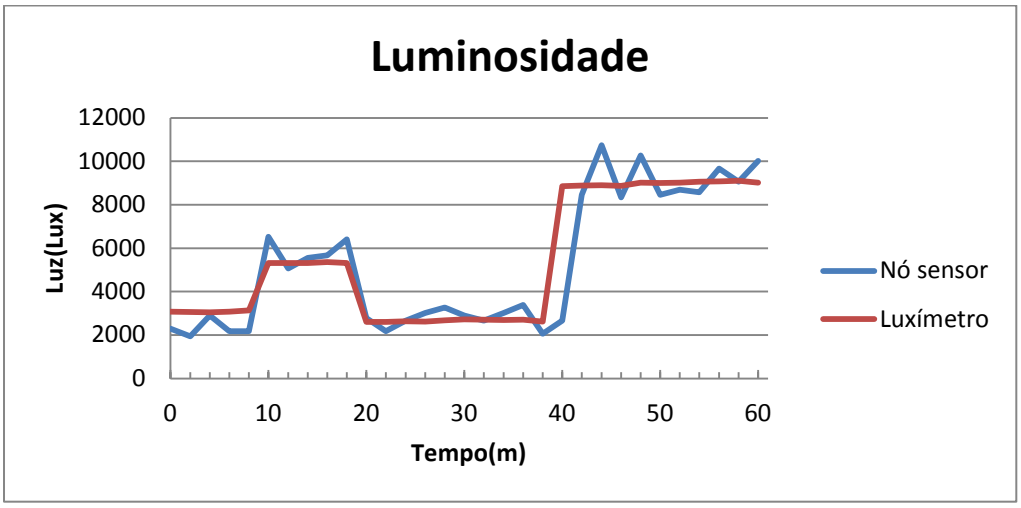


Figura 5.9 - Comparação da luminosidade obtida pelo nó sensor como luxímetro

Por fim, foi comparado o valor da potência do sinal recebido pelo nó sensor com o do analisador de espectros. O teste foi efectuado com antenas monopolo, seja na emissão e na recepção, a uma altura de 2,35 metros do solo como mostra a Figura 5.10.

Figura 5.10 - Equipamento utilizado no teste de RSSI

A potência transmitida em ambos os casos foi de 5 dBm, e registaram-se 26 pontos distanciados de metro a metro. O teste foi efectuado no terraço da Universidade, pois é o local mais perto das imediações com o mínimo de reflexões possível. A comparação dos valores obtidos é visível na Figura 5.11.

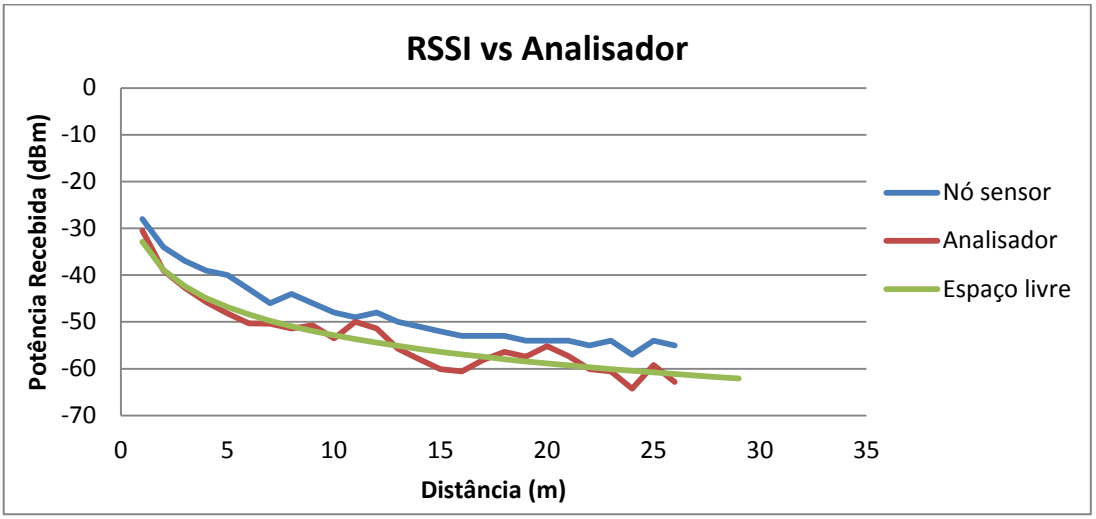


Figura 5.11 - Comparação da potência do sinal recebido com o nó e o analisador.

Verifica-se que o valor do sinal de RSSI obtido pelo nó sensor embora acompanhe o do analisador existe uma diferença média de 5 dB entre estes. Em comparação com a curva em espaço livre o analisador apresenta uma boa aproximação.

5.2.2 Testes no terraço e na vegetação

A preparação dos equipamentos para aferir os resultados na vegetação e no terraço pode ser visualizada na Figura 5.12.

Figura 5.12 - Equipamentos de teste.

Os testes tiveram a duração de uma hora e foram efectuados em horas completamente distintas. Os testes na **vegetação** iniciaram às 10:50 h. Primeiramente são apresentados os resultados referentes aos testes realizados na vegetação, ilustrados na Figura 5.13.

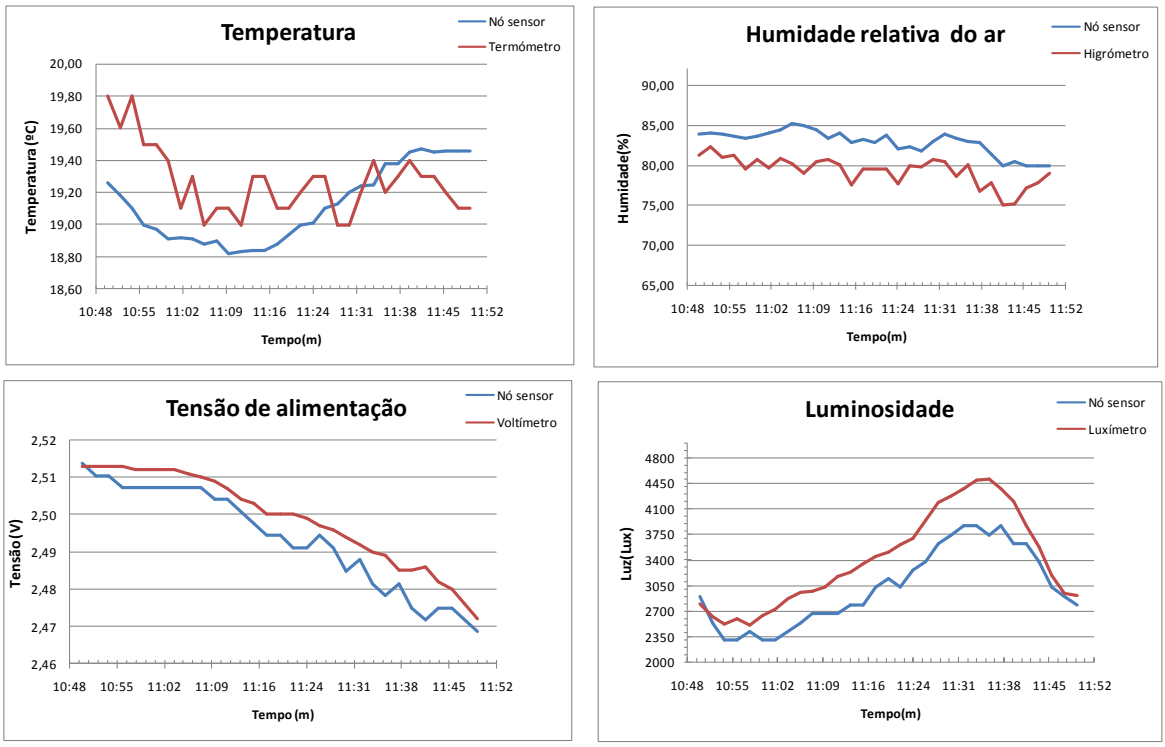


Figura 5.13 -Nó sensor *vs* equipamento no meio florestal.

Para os quatro parâmetros apresentados, observa-se que as amostras obtidas pelo nó sensor apresentam um comportamento idêntico ao medido pelo equipamento. A situação que exibe um maior erro recaiu sobre a luminosidade em que a diferença média entre o equipamento e o nó sensor é em torno dos 370 Lux. Esta discrepância poderá dever-se ao facto do luxímetro apresentar uma maior área de captação solar. Através da análise gráfica da humidade relativa do ar verifica-se que o nó sensor apresenta uma percentagem superior de humidade. Isto pode ser justificado com a diferença de posicionamento entre o equipamento e o nó sensor (colocado no interior de um tubo PVC, Figura 4.7). Em relação aos outros parâmetros não existe uma grande divergência entre os valores obtidos.

No que concerne aos testes realizados no **terraço** estes iniciaram-se às 17:05 h e os resultados obtidos podem ser visualizados na Figura 5.14.

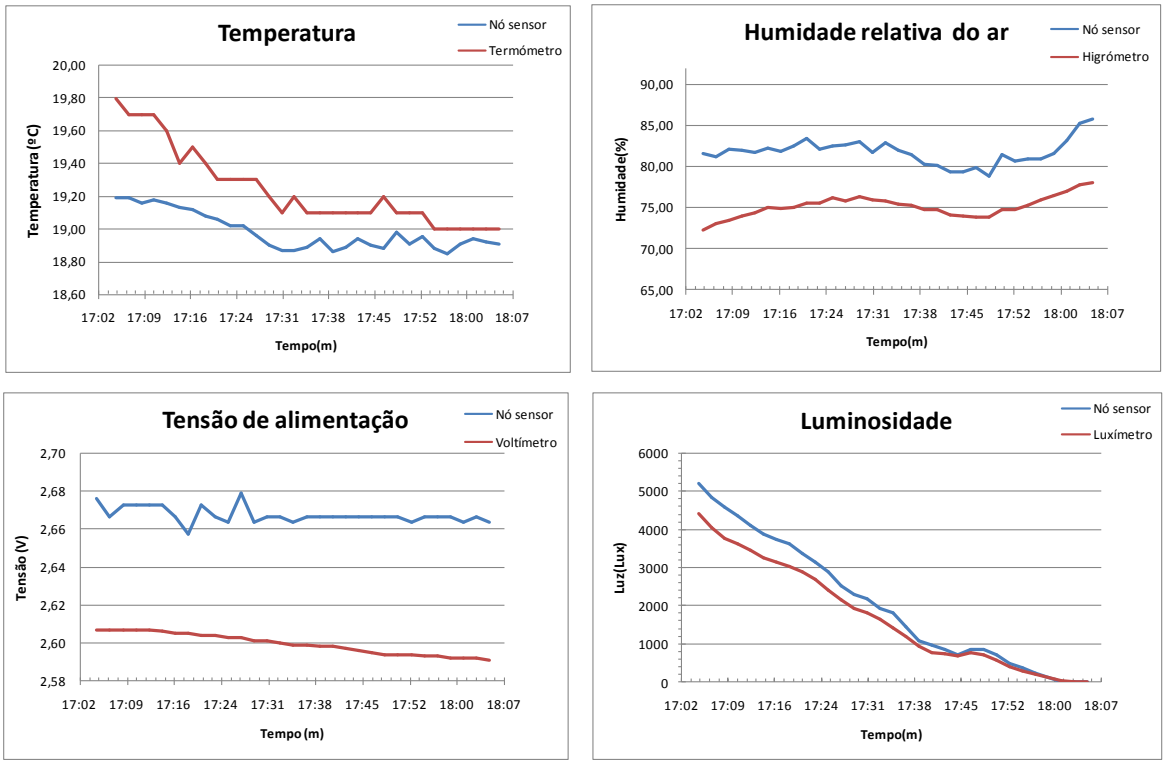


Figura 5.14 - Nó sensor *vs* equipamento no terraço.

Os valores obtidos seguem um comportamento análogo aos valores apresentados no ponto anterior. No que respeita à luminosidade constata-se que às 18:00 h tanto o nó sensor como o luxímetro não conseguiram detectar a presença de luminosidade.

Nos dois testes descritos verifica-se que o nó sensor apresenta resultados próximos do equipamento utilizado podendo aferir que os nós sensores implementados fornecem uma boa estimativa dos parâmetros em estudo.

5.3 Técnica de adormecimento dos routers

5.3.1 Teste em laboratório

Antes da instalação dos *routers* para implementação da rede de monitorização precederam-se a testes em laboratório com o intuito de averiguar o comportamento da técnica de adormecimento.

O primeiro teste foi realizado para averiguar qual a duração da bateria para o funcionamento do nó em duas situações. Foram utilizados dois nós sensores para efectuar a comparação dos níveis de bateria em laboratório. O primeiro nó foi configurado sem a técnica, pelo que não ficou habilitado a dormir, enquanto no segundo nó a técnica foi implementada estabelecendo um período de amostragem de 300 segundos, dos quais 265 segundos o nó fica a dormir e os restantes 35 segundos fica acordado para envio da mensagem de amostra e recepção da mensagem de *broadcast.* Para a realização do teste recorreu-se ao uso de baterias recarregáveis com carga máxima a 2,8 V. A Figura 5.15 ilustra a comparação da duração de funcionamento dos nós. Verifica-se que o nó sensor que não foi configurado

com a técnica, apresenta um tempo de operação de aproximadamente 1,45 dias, enquanto o nó com a respectiva técnica apresentou uma duração de 8 dias.

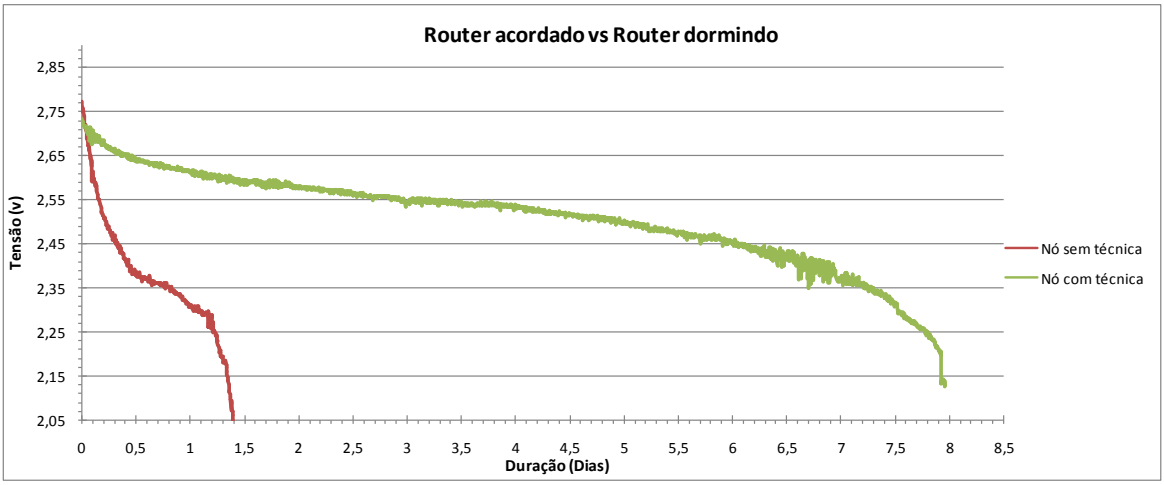


Figura 5.15 - Comparação da durabilidade do nó com e sem técnica de adormecimento

Para testar a técnica de adormecimento foram configurados 8 *routers* e colocados ao longo de um dos corredores da Universidade ficando um último nó no exterior como se pode visualizar na Figura 5.16. Para garantir um único salto (*one hop*) entre eles foi necessário efectuar alterações nas potências de emissão do rádio e ir testando com a funcionalidade do RSSI o nível de sinal do nó vizinho.

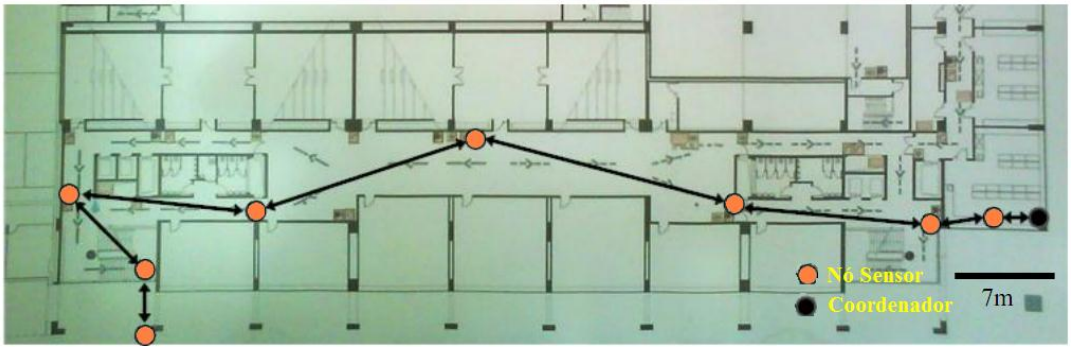


Figura 5.16 - Distribuição dos nós sensores no interior da Uma.

Este teste teve a duração de 3:15 h. Como o objectivo do teste é verificar o número de mensagens enviadas pela rede até atingir o coordenador optou-se por um ciclo de 75 segundos, dos quais 40 s correspondem ao nó sensor no estado de adormecimento e os restantes 35 s no modo activo como indica a linha temporal na Figura 5.17.

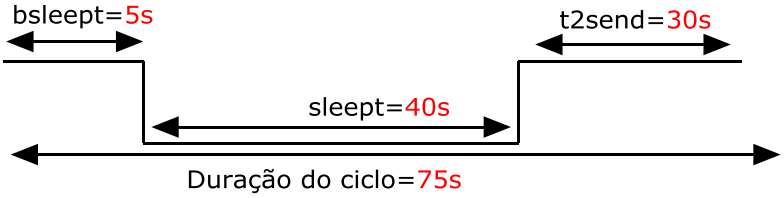


Figura 5.17 - Ciclo de amostragem com os tempos definidos.

Idealmente em cada ciclo são recebidas 8 mensagens distintas. Considera-se um ciclo bem sucedido quando são recebidas pelo menos 8 mensagens distintas. Todas as restantes situações são consideradas de insucessos.

Para obter uma taxa de sucesso de 100%, o número de ciclos terá de ser igual a 156 o que consiste em receber pelo menos 8 mensagens distintas durante o teste.

O gráfico da Figura 5.18 ilustra a comparação do número de ciclos com sucesso e insucesso obtidos no teste face à situação ideal.

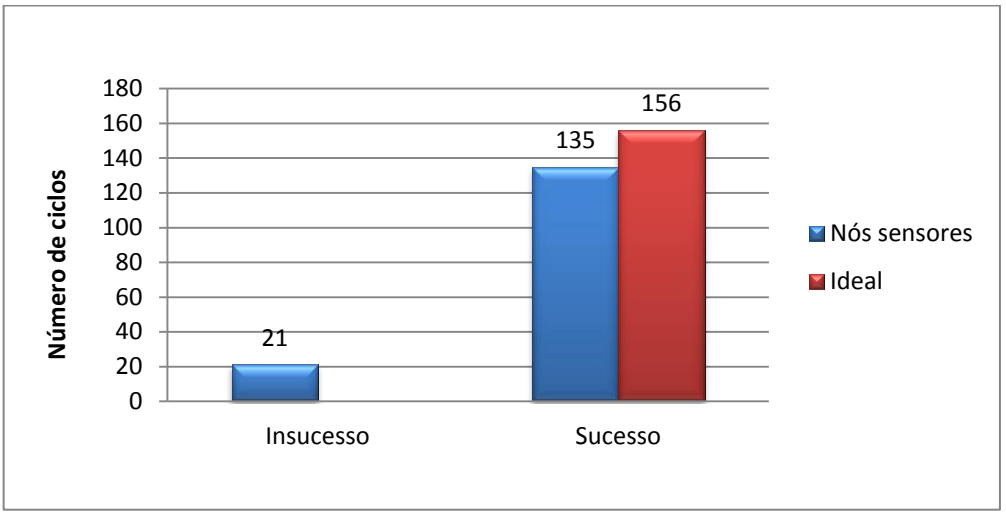


Figura 5.18 – Comparação entre ciclos com sucesso na situação ideal e obtidos no teste.

Verifica-se que a taxa de ciclos com sucesso é de 86.5% e a taxa de insucessos é 13.5%.

A fim de melhorar a taxa de sucesso foi implementado em cada nó sensor uma alteração no código do microcontrolador. Esta alteração faz com que haja uma repetição da mensagem a ser enviada pelo nó sensor, de pelo menos 5 vezes.

Isto acontece, no caso em que o rádio XBee não receba uma mensagem de reconhecimento (*acknowledge*) no momento após enviar a mensagem com as amostras. De seguida procedeu-se à actualização do *firmware* e efectuou-se novamente um teste nas mesmas condições verificando-se uma taxa de sucesso de 100%.

5.3.2 Na situação final de testes

Os testes realizados anteriormente serviram para estudar os principais parâmetros do sistema e para averiguar o comportamento da técnica antes de uma implementação final. Desta forma procedeu-se à colocação dos routers no exterior da Universidade com a distribuição mencionada no ponto 5.1 de modo a testar a viabilidade da técnica nesse ambiente. De referir que o teste teve a duração de cinco dias como uma frequência de amostragem de 5 minutos estimando um total de 1440 ciclos. Os gráficos da Figura 5.19 ilustram os resultados obtidos.

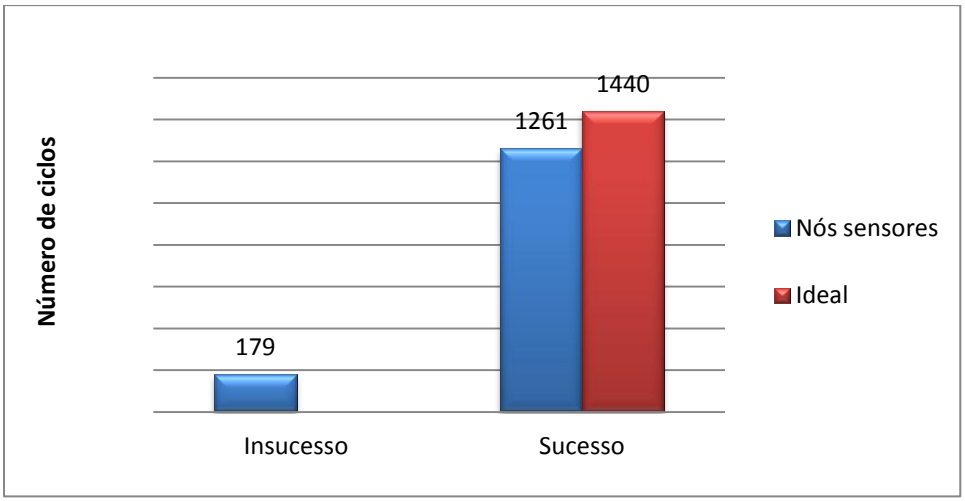


Figura 5.19 - Comparação entre ciclos com sucesso na situação ideal e obtidos no teste.

Considerando o raciocínio do ponto anterior, em termos de eficiência, verifica-se que para este caso a eficiência é de 87.5 %. A taxa de ciclos considerados como insucessos foi de 12,5 %. De mencionar que neste teste o artifício de repetição de mensagens descrito anteriormente já estava implementado.

A fim de verificar a duração das baterias foram analisadas as tensões das mesmas referentes aos dois nós sensores que não continham o painel solar. O resultado da análise é apresentado na Figura 5.20.

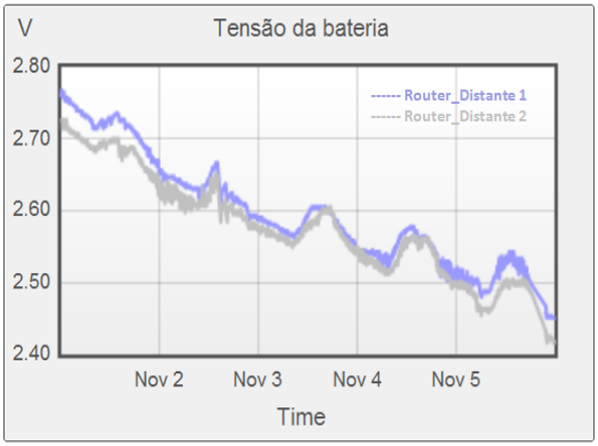


Figura 5.20 - Tensão nas baterias referentes aos routers "distantes".

O resultado apresentado sugere uma boa aproximação ao resultado obtido em laboratório, com a respectiva técnica. Note-se que os gráficos não apresentam uma curva como a obtida em laboratório. As pequenas elevações presentes poderão dever-se ao aumento da temperatura no interior das caixas atendendo a que as mesmas se encontram totalmente expostas ao sol.

Verifica-se que a durabilidade das baterias tende para o mesmo número de dias que foi apresentado na Figura 5.15, comprovando-se que esta técnica favorece uma maior autonomia aos nós sensores.

5.4 Resultados gerais dos nós sensores

A implantação dos nós sensores teve por principal objectivo realizar a monitorização ambiental. A rede foi submetida a testes, de forma averiguar o comportamento dos diferentes parâmetros ambientais, nomeadamente: luminosidade, humidade e temperatura, para além da tensão da bateria e valor de RSSI, durante um determinado período. Os resultados obtidos encontram-se apresentados na Figura 5.21.

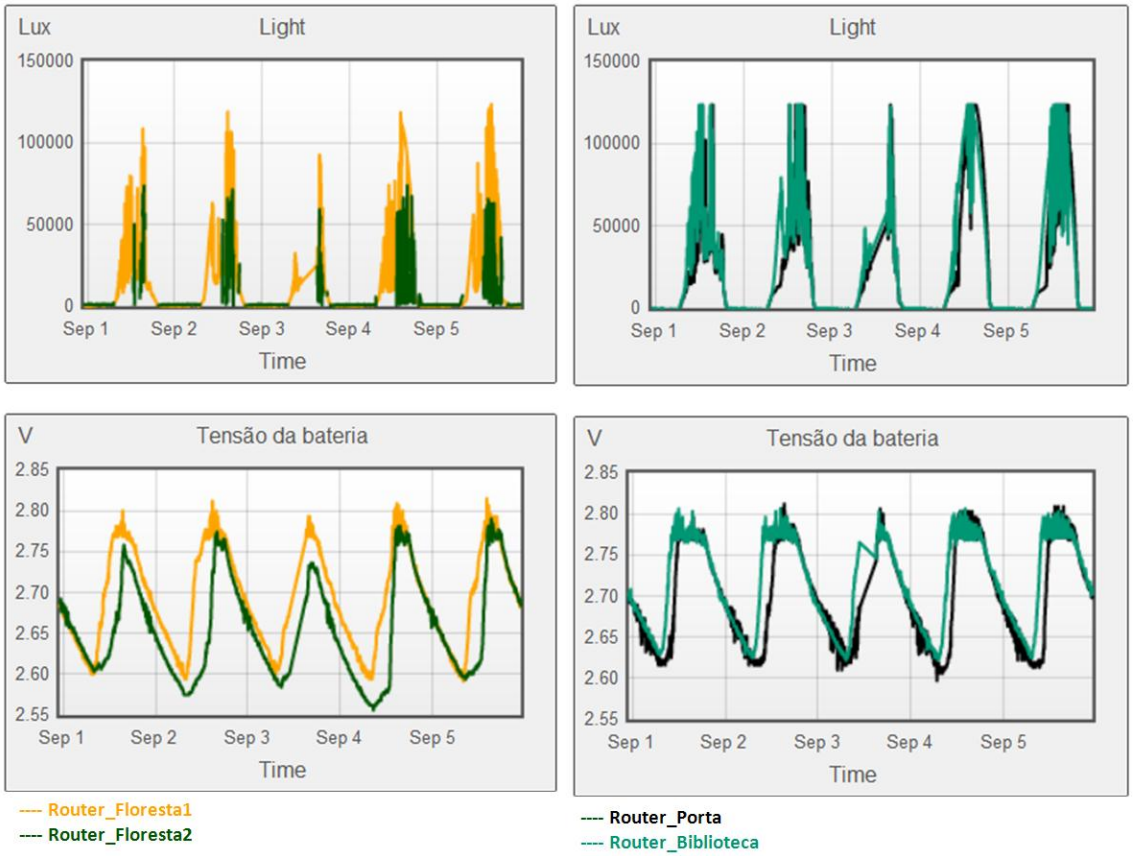


Figura 5.21 - Gráficos comparativos em dois meios utilizando o sensor de luminosidade e o nível de tensão nas baterias.

Os gráficos apresentam os valores obtidos para duas situações distintas: meio com vegetação e sem vegetação (terraço da Universidade). Os *routers* que identificam o meio com vegetação são Floresta1 e Floresta2. Relativamente aos que se encontram no terraço são denominados por Porta e Biblioteca.

No que concerne aos nós sensores presentes no meio florestal, verifica-se que o *router* Floresta2 apresenta níveis de luminosidade inferiores ao Floresta1, isto porque este posiciona-se num meio de vegetação mais denso, o que já era esperado. Este facto leva a que o painel solar não seja abrangido com tanta luz solar, fazendo com que as baterias deste nó não sejam carregadas da mesma forma que as do Floresta1. Verifica-se que, a tensão das baterias seguem um aumento do nível de tensão quando existe um aumento na luminosidade.

Relativamente aos nós posicionados no terraço da Universidade, observa-se que o comportamento dos parâmetros em estudo é análogo aos apresentados pelos nós implantados

na floresta. No entanto, estes não apresentam divergências a nível da exposição solar do painel, assim sendo, apresentam valores de luminosidade semelhantes e, consequentemente, em ambos os *routers* as baterias apresentam níveis idênticos de tensão.

Ao comparar-se os nós nos meios em que estão inseridos, pode-se constatar que no meio com vegetação existe uma maior variação de luminosidade, visto que há obstrução à passagem de luz solar devido às folhagens existentes no mesmo.

Tendo em conta que os níveis de luminosidade recebidos pelos painéis acoplados aos *routers* presentes no terraço são superiores e mais constantes, conseguem fornecer mais corrente para o recarregamento das baterias, fazendo com que a tensão aos terminais destas apresente valores mais elevados (em torno dos 2,78 V). Este facto leva a concluir que os nós presentes no terraço terão mais autonomia a nível de alimentação, face aos implementados na floresta.

Realizado o estudo da tensão das baterias e da luminosidade, procede-se à apresentação dos resultados obtidos face à humidade e temperatura, nos meios já supracitados. A Figura 5.22 apresenta os valores alcançados.

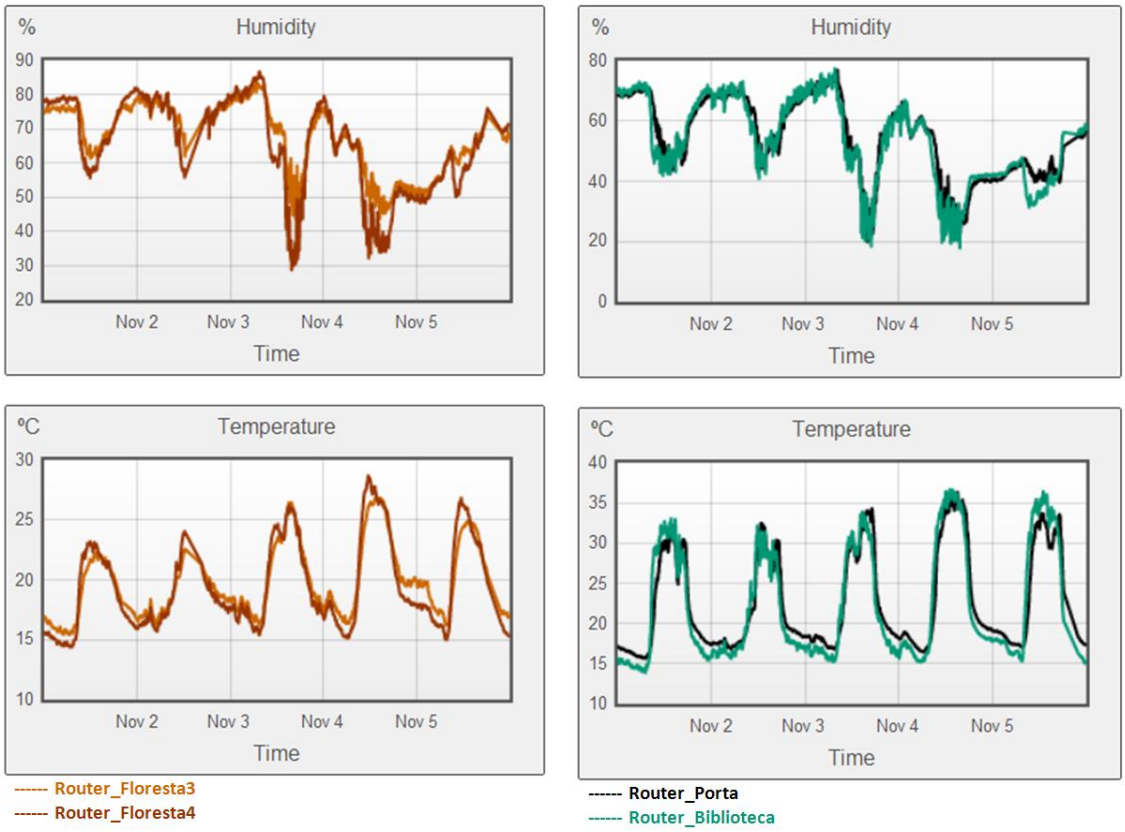


Figura 5.22 – Gráficos comparativos em dois meios utilizando o sensor SHT15

A análise da humidade e temperatura recai sobre os mesmos *routers* presentes no terraço e os outros dois routers colocados na floresta (Floresta3 e Floresta4), de modo a avaliar o desempenho de todos routers no meio em questão.

No que respeita aos nós sensores existentes no meio florestal, verifica-se que os dois nós apresentam o mesmo desempenho no que concerne a valores apresentados no gráfico

referente à humidade e temperatura. Isto deve-se ao facto de não existir qualquer divergência relativamente às características do meio no qual os *routers* se encontram. É de referir que para os nós presentes no terraço os resultados obtidos não apresentam alteração significativa no comportamento da temperatura e humidade, isto porque, as características do meio no qual se encontram são semelhantes.

Ao realizar a analogia do desempenho dos nós sensores nos dois meios, verifica-se que estes seguem um comportamento idêntico, mas os valores para humidade para o meio florestal são superiores face aos dados obtidos no terraço, o que era de esperar tendo em conta as características intrínsecas do mesmo. No que se refere à temperatura, observa-se que no terraço os valores obtidos são superiores, devido às razões já apresentadas. É de salientar que o sensor de humidade e temperatura encontra-se no interior do tubo PVC, fazendo com que os valores obtidos sejam superiores aos que eram esperados atendendo a que a amostragem foi efectuada no mês de Novembro.

Verifica-se a correcta monitorização do sensor de humidade e temperatura, tendo em conta que quando existem um aumento na humidade o valor da temperatura decresce e vice-versa.

Tendo em conta toda a análise efectuada face aos resultados obtidos, pode verificar-se o correcto funcionamento dos sensores independentemente do meio em que estão inseridos.

5.4.1 RSSI - Antenas monopolo

A Figura 5.23 ilustra a variação dos valores da potência recebida por cada nó sensor em relação ao nó adjacente, assim como os valores de potência recebida com o analisador de espectros. É de referir que a potência de saída de cada nó sensor é de aproximadamente 6,3 dBm ($P_{saída\_do\_rádio+}G_{antena\_monopolo}$).

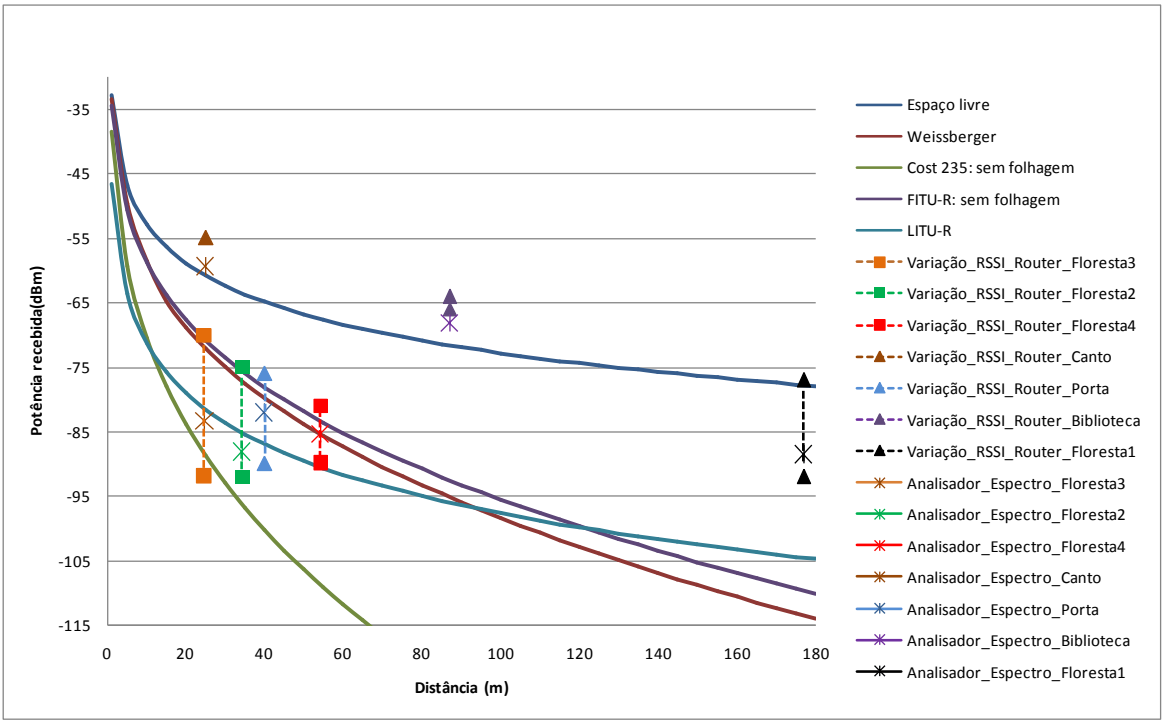


Figura 5.23 - Comparação dos valores do RSSI obtidos pelos nós sensores face aos modelos de propagação e ainda face ao valor obtido com o analisador de espectros.

Como foi apresentado na Figura 5.11, pode observar-se que os valores obtidos com o analisador de espectros para os nós num ambiente em que se pode utilizar o modelo do espaço livre encontram-se abaixo do valor médio de potência recebida por cada nó sensor.

Relativamente à variação de potência recebida pelos nós sensores presentes no interior da vegetação, verifica-se que estes encontram-se dentro dos valores de potência recebida descrita pelos modelos de propagação em meios com vegetação, à excepção do modelo COST 235 sem folhagem que apresenta um valor de potência recebida muito atenuado. Também se verifica que o valor obtido pelo analisador de espectros está dentro da gama indicada pelo RSSI.

Em relação ao Router_Floresta1, é de referir que, apesar deste encontrar-se no meio da vegetação, o seu nó adjacente (Router_Biblioteca) encontra-se num ponto mais elevado e fora da vegetação, o que faz com que o seu valor de potência recebida se encontre mais próximo do modelo de propagação em espaço livre.

Os valores de potência recebida pelos nós Router_Canto e Router_Biblioteca, pode observar-se que a variação do RSSI obtido encontra-se acima da curva característica do espaço livre, podendo isto dever-se a reflexões construtivas existentes no meio.

De constatar que o valor de RSSI referente ao Router_Porta, embora este se encontre num meio sem vegetação, o seu valor encontra-se muito abaixo da curva característica do espaço livre, podendo isto dever-se ao facto de que o seu nó adjacente (Router_Canto) não está em linha de vista, ou seja, existem obstáculos (parede) entre a sua ligação.

5.4.2 RSSI - Antenas grelha e monopolo

Atendendo a que os ganhos das antenas grelha são de 9 e 11 dBi respectivamente foi necessário alterar a potência de transmissão dos nós onde as grelhas se encontravam presentes

(1 dBm para a grelha de 9 dBi e -1 dBm para a grelha de 11 dBi) isto para que a potência total de saída não ultrapassasse o limite máximo permitido (10 dBm) por lei para a banda ISM.

Na Figura 5.24 apresenta-se a comparação entre a variação dos valores de potência recebida pelos nós sensores em relação à curva característica em espaço livre e aos resultados obtidos com o analisador de espectros. É de referir ainda que as curvas apresentadas nesta figura já têm em consideração o valor da potência emitida assim como os ganhos das antenas em questão.

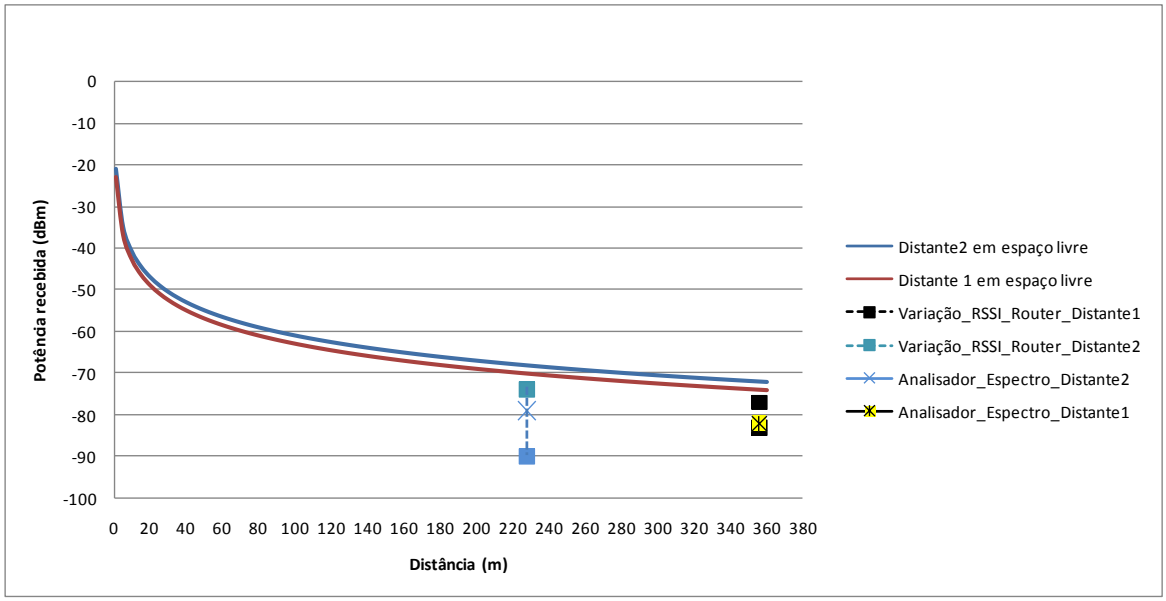


Figura 5.24 - Comparação dos valores do RSSI obtidos pelos nós sensores face ao modelo de propagação em espaço livre e ainda face ao valor obtido com o analisador de espectros

Analisando-se a Figura 5.24 pode observar-se mais uma vez que a variação do valor do RSSI amostrado por cada nó sensor vai de encontro com o valor obtido pelo analisador de espectros. Ainda verifica-se que o valor de RSSI obtido encontra-se ligeiramente abaixo do que era esperado no espaço livre, porque embora as antenas estivessem em linha de vista, existem sempre obstáculos que atenuam o seu sinal.

6 Conclusões e trabalhos futuros

Neste capítulo são abordadas as principais conclusões acerca do estudo desenvolvido na área de rede de sensores sem fios. Posteriormente é realizada uma pequena exposição relativamente aos aspectos que podem ser melhorados e desenvolvidos.

6.1 Conclusões

Com a realização deste trabalho verificou-se que apesar de existir uma vasta informação sobre as RSSF para monitorização ambiental, ainda não foi apresentada uma solução que seja completamente praticável para ser aplicada no meio florestal.

O protocolo utilizado para implementar a rede foi o Zigbee, concluindo-se que este foi suficiente para permitir a elaboração de uma rede de nós sensores sem fios com um baixo consumo energético.

Verificou-se que é fundamental a presença de quatro unidades para a construção de um nó sensor sendo estas: a unidade de controlo, alimentação, monitorização e comunicação. Outro ponto essencial consistiu na escolha mais adequada dos componentes que constituem as unidades supracitadas, visto que é um requisito fundamental o baixo consumo de energia.

No que concerne à técnica de adormecimento dos routers afere-se que esta permite uma maior autonomia do nó sensor, visto que possibilita o adormecimento do módulo, minimizando o consumo de energia durante um determinado período. Averigua-se que a taxa de sucesso para esta técnica é cerca de 80 %, para as condições impostas pelo respectivo teste.

Em relação aos nós sensores implantados no meio com vegetação conclui-se que o valor de RSSI obtido vai de encontro com o valor expectável caso fosse empregue os modelos de propagação, sendo que este resultado aproxima-se mais do modelo LITU-R.

Relativamente à monitorização ambiental, constata-se que é possível obter os parâmetros de interesse em tempo real e visualizar numa página de *Internet*.

6.2 Trabalhos futuros

Como trabalhos futuros, pode-se aumentar a cobertura dos nós sensores até ao Pico do Areeiro com o intuito de monitorizar as florestas nesse espaço.

Adicionar novos tipos de sensores (humidade do solo, acústico, oxigénio, entre outros) para obter-se um maior número de parâmetros referentes ao meio.

Aplicar outras fontes de energias renováveis (eólica, hídrica) aos nós sensores em meio florestal de modo a que a duração dos mesmos seja superior.

Referências

[1] Chee-Yee Chong and Srikanta P Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," 2003.

[2] Cooperative Engagement Capability (CEC). Disponivel em: <http://www.globalsecurity.org/military/systems/ship/systems/cec.htm>. Consultado em: 28 Setembro 2009.

[3] Fixed Distributed System (FDS). Disponivel em: <www.globalsecurity.org/intell/systems/fds.htm>. Consultado em: 26 Setembro 2009.

[4] Srikanta Kuma and David Sheperd, "SensIT: Sensor Information Technology For the Warfighter,".

[5] Kazem Sohraby, Daniel Minoli, and Taieb Znati, *Wireless Sensor Networks - Technology, Protocols, and Applications*. New Jersey: John Wiley & Sons, Inc., 2004.

[6] Guimarães De Freitas Germano, "Impacto da utilização de mecanismos de segurança em Redes de Sensores Sem Fio," Universidade Federal de Pernambuco, 2005.

[7] Jason Lester Hill, "System Architecture for Wireless Sensor Networks," Universidade da Califórnia, Berkeley, Tese de Doutoramento 2003.

[8] Crossbow Technology. Disponivel em: <http://www.xbow.com/Products/wproductsoverview.aspx>. Consultado em: 08 Outubro 2009.

[9] The Technology Environmental Monitoring. Crossbow Technology: Eko. Disponivel em: <http://www.xbow.com/Eko/eko_product1.aspx>. Consultado em: 07 Outubro 2009.

[10] Eko Nodes. datashhet. Disponivel em: <http://camalienetworks.com/eKo_Pro_System_datasheet.pdf>. Consultado em: 26 Abril 2010.

[11] Ivanovitch Medeiros Dantas Da Silva, "Redes de Sensores sem Fio aplicadas em Ambientes Industriais de Petróleo e Gás," UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE, 2006.

[12] Yong-sork Her, Byungrak Son, and Jung-Gyu Kim, "A Design and Implementation of Forest-Fires Surveillance System based on Wireless Sensor Networks for South Korea Mountains," *IJCSNS International Journal of Computer Science and Network Security*, vol. 6B, pp. 124-130, Setembro 2006.

[13] Sensor TIP50CM. Disponivel em: <http://www.tradekorea.com/product-detail/P00045960/TIP50CM.html>. Consultado em: 15 Junho 2010.

[14] FireBug. Disponivel em: <http://firebug.sourceforge.net/index.php>. Consultado em: 04 Novembro 2009.

[15] David M. Doolin and Nicholas Sitar, "Wireless sensors for wildfire monitoring," University of California, Berkeley,CA, 2004.

[16] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, and David Culler, "Wireless Sensor Networks for Habitat Monitoring," Intel Research Laboratory, Berkeley,.

[17] Great Duck sensor. Disponivel em: <http://www.wired.com/wired/archive/11.12/network_pr.html>. Consultado em: 26 Junho 2010.

[18] Jukka Suhonen, Mikko Kohvakka, Marko Hännikäinen, and Timo D. Hämäläinen, "Design, Implementation, and Experiments on Outdoor Deployment ofWireless Sensor Network for Environmental Monitoring," Tampere University of Technology/Institute of Computer and Digital Systems, 2006.

[19] José A. Gutiérrez, Edgar H. Callaway Jr, and Raymond L. Barret, *Enabling Wireless Sensors with 802.15.4*. New York, USA: standards Information Network, 2004.

[20] André Teixeira da Silva, "Módulos de Comunicação Wireless para Sensores," Universidade do Porto, Porto, Projecto Final de Curso 2007.

[21] Pilha Protocolar. Disponivel em: <http://chipdesignmag.com/sld/files/2009/06/zigbee_wirelesshealth1.jpg>. Consultado em: 29 Abril 2010.

[22] S Pavan Kumar. MAC for 802.15.4, 2005. Disponivel em: <http://www.cse.iitk.ac.in/users/cs698t/Lecture_notes/Y1306_scribenotes.pdf>. Consultado em: 29 Junho 2010.

[23] Disponivel em: <http://www.ieee802.org/15/pub/TG4.html>. Consultado em: 25 Abril 2010.

[24] MeshNetics eLearning. Disponivel em: <http://www.meshnetics.com/learning/index.php?show=22>. Consultado em: 7 Julho 2010.

[25] Woo-Jin Choi and Sirin Tekinay, "An Efficient Table Driven Routing Algorithm for Wireless Ad Hoc Networks," New Jersey Institute of Technology, New Jersey Center for Wireless Telecommunications , 2001.

[26] Wikipedia, The Free Encyclopedia. Disponivel em: <http://en.wikipedia.org/wiki/Comparison_of_802.15.4_radio_modules>. Consultado em: 24 Agosto 2010.

[27] ENERGIZER NH15-2500. datasheet. Disponivel em: <http://data.energizer.com/PDFs/nh15-2500.pdf>. Consultado em: 29 Aug. 2010.

[28] Max1675. datasheet. Disponivel em: <http://pdfserv.maxim-ic.com/en/ds/MAX1674-MAX1676.pdf>. Consultado em: 29 Aug. 2010.

[29] Apresentação_Aula02. Disponivel em: <http://www.scribd.com/doc/13928773/ApresentacaoAula02>. Consultado em: 25 Novembro 2010.

[30] MSX-005F. Datasheet. Disponivel em: <http://www.farnell.com/datasheets/54658.pdf>. Consultado em: 10 Outubro 2010.

[31] Lina Teixeira, "Sistema de Alimentação Autónomo para Monitorização Ambiental,"

Universidade da Madeira, Madeira, 2010.

[32] Tl2575. datasheet. Disponivel em: <http://focus.ti.com/lit/ds/symlink/tl2575-05.pdf>. Consultado em: 25 Feb. 2010.

[33] XBee/XBee-PRO OEM RF Modules. datasheet. Disponivel em: <http://elmicro.com/files/sparkfun/xbee-25-manual.pdf>. Consultado em: 29 Aug. 2010.

[34] The Major Differences in the XBee Series 1 vs. the XBee Series 2. Disponivel em: <http://www.digi.com/support/kbase/kbaseresultdetl.jsp?id=2213>. Consultado em: 07 Sep. 2010.

[35] datasheet. Disponivel em: <http://www.e-lab.de/downloads/DOCs/mega48_88_168.pdf>. Consultado em: 08 Sep. 2010.

[36] Atmega168. Disponivel em: <fonte: http://parts.digikey.com/1/parts/1561038-mcu-avr-16k-flash-15mhz-32-tqfp-atmega168-15az.html>. Consultado em: 28 Julho 2010.

[37] Sensirion datasheet. Disponivel em: <http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf>. Consultado em: 12 Sep. 2010.

[38] Uilian Lucas de Souza and Thiago Ramos Pereira. Fotodiodos e Fototransistores. Disponivel em: <http://www.etb.com.br/professor/materialdeapoio/transistor.pdf>.

[39] Sensor S1087. Disponivel em: <http://jp.hamamatsu.com/products/sensor-ssd/pd041/pd050/pd051/S1087/index_en.html >. Consultado em: 1 Novembro 2010.

[40] Disponivel em: <http://electronicdesign.com/article/analog-and-mixed-signal/what-s-all-this-transimpedance-amplifier-stuff-any.aspx>. Consultado em: 15 Sep. 2010.

[41] Application circuit examples of Si photodiode. Disponivel em: <http://sales.hamamatsu.com/assets/applications/SSD/si_pd_circuit_examples.pdf>. Consultado em: 14 Sep. 2010.

[42] AVR-ISP500. Disponivel em: <http://www.olimex.com/dev/pdf/AVR/AVR-ISP500.pdf>. Consultado em: 22 Setembro 2010.

[43] Antena Theory. Disponivel em: <http://www.antenna-theory.com/basics/friis.php>. Consultado em: 20 Novembro 2010.

[44] Joaquim A.R. Azevedo and Filipe E.S. Santos, "An empirical propation model for forest environments at tree trunk level,".

[45] Manual. Disponivel em: <http://www.ni.com/pdf/manuals/371585a.pdf>. Consultado em: 27 Setembro 2009.

[46] PC-D-275 Task Group (D-31b), "Generic Standard on Printed Board Design," The Institute for Interconnecting and Packaging Electronic Circuits, 1998.

[47] 'Tabela IP'. Disponivel em: <http://www.aquatext.com/tables/ip_ratings.htm>. Consultado em: 05 Outubro 2010.

[48] Manual Xbee. Disponivel em:

<http://elmicro.com/files/sparkfun/XBee-25-manual.pdf>. Consultado em: 16 Fevereiro 2010.

[49] MySQL: The The world's most popular open source database. Disponivel em: <http://www.mysql.com/>. Consultado em: 15 Maio 2010.

[50] apache friends. Disponivel em: <http://www.apachefriends.org/en/xampp.html>. Consultado em: 16 Maio 2010.

[51] Notepad++|5.8.3. Disponivel em: <http://notepad-plus-plus.org/download>. Consultado em: 01 Junho 2010.

[52] Analisador de Expectros. Disponivel em: <http://www2.rohde-schwarz.com/file_6913/FSH_Technical_Information_en_v11.pdf>. Consultado em: 20 Novembro 2010.

[53] Fluke 111. Disponivel em: <http://www.fluke.pt/comx/show_product.aspx?pid=29980&product=HMA&type=3&locale=ptpt>. Consultado em: 20 Novembro 2010.

[54] Luximetro. Disponivel em: <http://www.pdeal.nl/Lutron/pdf/LX-101.pdf>. Consultado em: 20 Novembro 2010.

[55] Fluke 87V. Disponivel em: < http://fluke.informationstore.net/efulfillment.asp?publication=10826-eng◊>. Consultado em: 20 Novembro 2010.

[56] Silva PRO-ADC. Disponivel em: <http://www.retrevo.com/d/ds/progress?doc=f715a2f9030e63d4b778a028bad3372d&rk=0.46468500094488263>. Consultado em: 20 Novembro 2010.

[57] Zigbee Alliance. Disponivel em: <http://www.zigbee.org/home.aspx>. Consultado em: 27 Apr. 2010.

[58] Patrick Kinney, "Wireless Control that simply Works," 2 Outubro de 2003.

[59] *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. New york, USA: Institute of Electrical and Electronics Engineers, Inc., 2003.

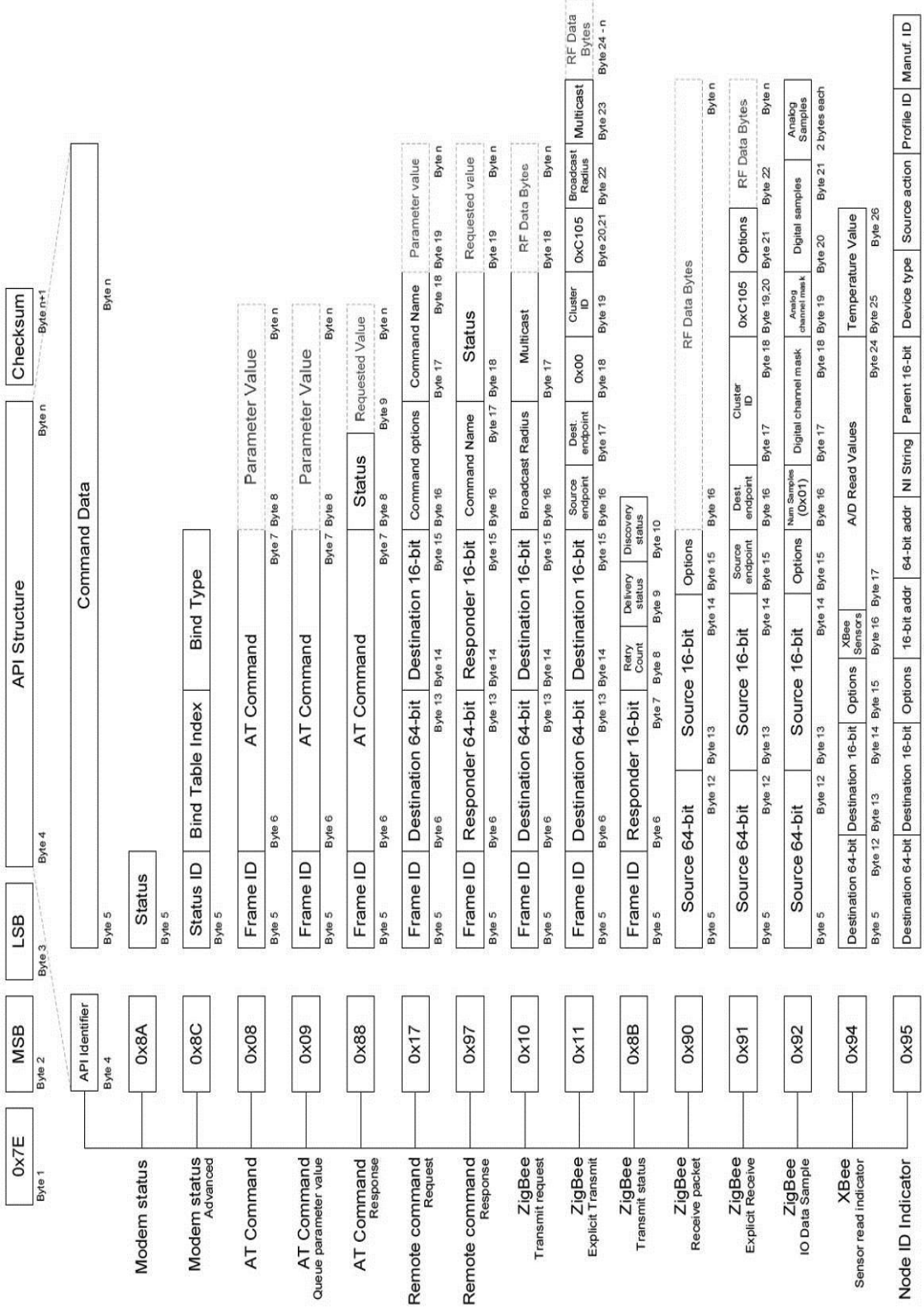
Anexo A - Especificações dos módulos XBee e XBee Pro[48].

Specification	XBee ZNet 2.5	XBee PRO ZNet 2.5
Performance		
Indoor/Urban Range	up to 133 ft. (40 m)	up to 300 ft. (100 m)
Outdoor RF line-of-sight Range	up to 400 ft. (120 m)	up to 1 mile (1.6 km)
Transmit Power Output	2mW (+3dBm), boost mode enabled 1.25mW (+1dBm), boost mode disabled	63mW (+18 dBm) 10mW (+10 dBm) for International variant
RF Data Rate	250,000 bps	250,000 bps
Serial Interface Data Rate (software selectable)	1200 - 230400 bps (non-standard baud rates also supported)	1200 - 230400 bps (non-standard baud rates also supported)
Receiver Sensitivity	-96 dBm, boost mode enabled -95 dBm, boost mode disabled	-102 dBm
Power Requirements		
Supply Voltage	2.1 - 3.6 V	3.0 - 3.4 V
Operating Current (Transmit, max output power)	40mA (@ 3.3 V, boost mode enabled) 35mA (@ 3.3 V, boost mode disabled)	295mA (@3.3 V)
Operating Current (Receive))	40mA (@ 3.3 V, boost mode enabled) 38mA (@ 3.3 V, boost mode disabled)	45 mA (@3.3 V)
Idle Current (Receiver off)	15mA	15mA
Power-down Current	< 1 uA @ 25°C	< 1 uA @ 25°C
General		
Operating Frequency Band	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960 x 1.297 (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)	-40 to 85° C (industrial)
Antenna Options	Integrated Whip, Chip, RPSMA, or U.FL Connector*	Integrated Whip, Chip, RPSMA, or U.FL Connector*
Networking & Security		
Supported Network Topologies	Point-to-point, Point-to-multipoint, Peer-to-peer, and Mesh	Point-to-point, Point-to-multipoint, Peer-to-peer, and Mesh
Number of Channels	16 Direct Sequence Channels	13 Direct Sequence Channels
Addressing Options	PAN ID and Addresses, Cluster IDs and Endpoints (optional)	PAN ID and Addresses, Cluster IDs and Endpoints (optional)
Agency Approvals		
United States (FCC Part 15.247)	OUR-XBEE2	MCQ-XBEEPRO2
Industry Canada (IC)	4214A-XBEE2	1846A-XBEEPRO2
Europe (CE)	ETSI	ETSI
RoHS	Compliant	Compliant

Anexo B - Configuração dos Pinos do módulo XBEE[48].

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / **CONFIG**	Input	UART Data In
4	DIO12	Either	Digital I/O 12
5	**RESET**	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI / DIO10	Either	PWM Output 0 / RX Signal Strength Indicator / Digital IO
7	PWM / DIO11	Either	Digital I/O 11
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ/ DIO8	Either	Pin Sleep Control Line or Digital IO 8
10	GND	-	Ground
11	DIO4	Either	Digital I/O 4
12	**CTS** / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / **SLEEP** / **DIO9**	Output	Module Status Indicator or Digital I/O 9
14	[reserved]	-	Do not connect
15	Associate / DIO5	Either	Associated Indicator, Digital I/O 5
16	**RTS** / DIO6	Either	Request-to-Send Flow Control, Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0 / Commissioning Button	Either	Analog Input 0, Digital IO 0, or Commissioning Button

Anexo C - Estrutura específica do modo API



Anexo D - Identificação dos pinos do ATMega168

Pino Físico/ Descrição do Pino	Componente/Ligação	Função	
1. PD3 - PCINT19, OC2B, INT1	**Comutação entre bateria e pSolar**	Interruptor	
2. PD4 - PCINT20, XCK, T0	**Comutação entre bateria e pSolar**	Interruptor	
3. GND		Massa	
4. VCC	Pino 1 do XBee	Alimentação	
5. GND	Pino 10 do XBee	Massa	
6. VCC		Alimentação	
7. PB6 - PCINT6, XTAL1, TOSC1	Oscilador 32,768 KHz	Clock Externo	
8. PB7 - PCINT7, XTAL2, TOSC2	Oscilador 32,768 KHz	Clock Externo	
9. PD5 - PCINT21, OCB0, T1			Livre
10. PD6 - PCINT22, OC0A, AIN0			Livre
11. PD7 - PCINT23, OC0B,	**Gate do Mosfet DivResistivo**	Interruptor	
12. PB0 - PCINT0, CLKO, ICP1	**Sensor de temperatura e Humidade**	DATA	
13. PB1 - PCINT1, OC1A	**Sensor de temperatura e Humidade**	SCK	
14. PB2 - PCINT2, OC1B, SS\	**LED+Resistência 2,2K**	Indicador	
15. PB3 - PCINT3, OC2A, MOSI	Pino 4 do JTAG	MOSI	
16. PB4 - PCINT4, MISO	Pino 1 do JTAG	MISO	
17. PB5 - PCINT5, SCK	Pino 3 do JTAG	SCK	
18. AVCC	//Condensador - 100 nF	Alimentação	
19. ADC6			Livre
20. AREF	Condensador. - 100 nF+massa	Tensão REF	
21. GND		Massa	
22. ADC7			Livre
23. PC0 - PCINT8, ADC0	**Leitura de bateria**	ADC-Data	
24. PC1 - PCINT9, ADC1	**Leitura de luminosidade**	ADC-Data	
25. PC2 - PCINT10, ADC2			Livre
26. PC3 - PCINT11, ADC3			Livre
27. PC4 - PCINT12, ADC4, SDA			Livre
28. PC5 - PCINT13, ADC5, SCL			Livre
29. PC6 - PCINT15, RESET	Pino 5 do JTAG//Resist 1k+Vcc	RESET	
30. PD0 - PCINT16, RXD	Pino 2 do XBee	TX	
31. PD1 - PCINT17, TXD	Pino 3 do XBee	RX	
32. PD2 - PCINT18, INT0	Pino 9 do XBee	Sleep	

Anexo E - Características do sensor SHT15

Parâmetros	Condição	Mínimo	Típico	Máximo	Unidade
Humidade					
Gama de operação		0		100	% RH
Precisão			±0,2	0,4	% RH
Resolução		0,4	0,05	0,05	RH
		8	12	12	Bit
Temperatura					
Gama de operação		-40		123,8	°C
Precisão		±1,5	±0,3	±1,7	°C
Resolução		0,04	0,01	0,01	°C
		12	14	14	Bit
Características gerais					
Tensão de alimentação		2.4	3.3	5.5	V
Corrente	a dormir		0.3	1,5	µA
	a medir		0.55	1	mA
	média	2	28		µA
Comunicação	Digital, utilizando dois fios (*DATA* e *SCK*)				
Armazenamento	10 – 50°C (0 – 125°C pico), 20 – 60%RH				

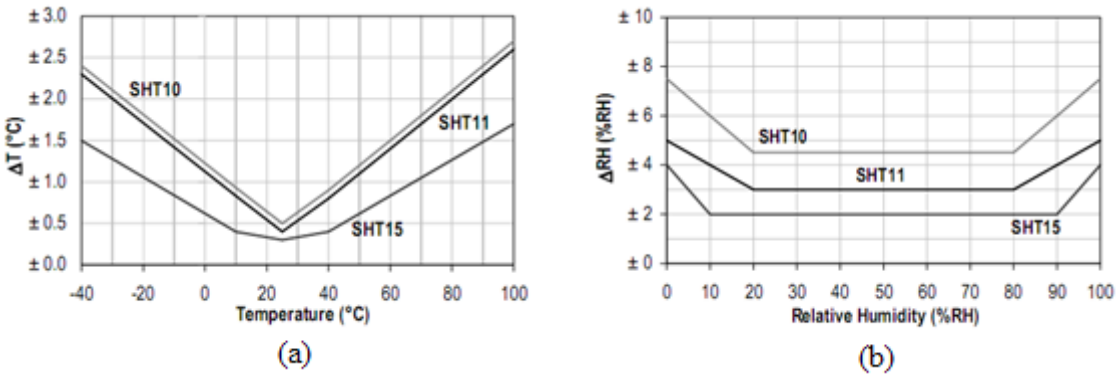
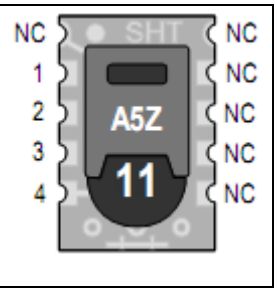


Figura E.1 - Variação da humidade relativa (a) e temperatura (b) para 3 tipos de sensores

Pino	Nome	Descrição
1	GND	Massa
2	DATA	*Serial Data*, bidireccional
3	SCK	*Serial Clock*,



4	VDD	Tensão de alimentação	
Nc	Nc	Não ligar	

Anexo F - Características do sensor S1087

■ Electrical and optical characteristics (Typ. Ta=25 °C, unless otherwise noted)

Type No.	Spectral response range λ (nm)	Peak sensitivity wavelength λp (nm)	Photo sensitivity S (A/W) λp	GaP LED 560 nm	He-Ne laser 633 nm	Infrared sensitivity ratio (%)	Short circuit current Isc 100 lx (μA)	Temp. coefficient of Isc (%/°C)	Dark current ID VR=1 V Max. (pA)	Temp. coefficient of ID TCID (times/°C)	Rise time tr VR=0 V RL=1 kΩ (μs)	Terminal capacitance Ct VR=0 V f=10 kHz (pF)	Shunt resistance Rsh VR=10 mV Min. (GΩ)	Shunt resistance Rsh VR=10 mV Typ. (GΩ)
S1087	320 to 730	560	0.3	0.3	0.19	10	0.16	-0.01	10	1.12	0.5	200	10	250
S1087-01	320 to 1100	960	0.58	0.33	0.38	-	1.3	0.1						
S1133	320 to 730	560	0.3	0.3	0.19	10	0.65	-0.01			2.5	700		100
S1133-01	320 to 1100	960	0.58	0.33	0.38	-	5.6	0.1						
S1133-14	320 to 1000	720	0.4		0.37	-	3.4		20		0.5	200		50

■ Short circuit current linearity

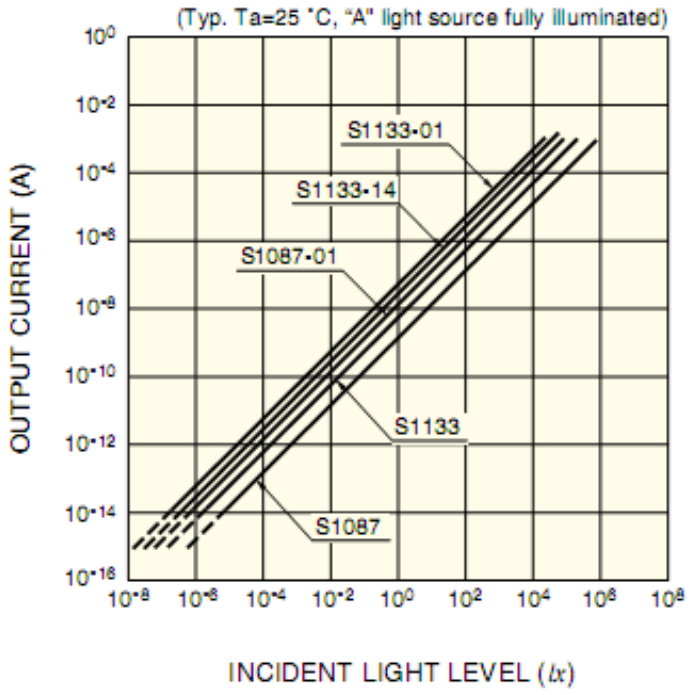


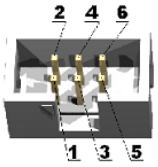
Figura F.1 - Recta de conversão da luz vs corrente em curto-circuito

Anexo G - Programador AVR-ISP500

O AVR-ISP500 é um programador de baixo custo com entrada USB que permite programar microcontroladores AVR. Definido pela *Atmel* suporta protocolo STK500v2 a qual é compatível com várias ferramentas para interacção com o microcontrolador. A ferramenta utilizada no projecto foi o *AVRStudio*. Na Figura G.1 pode-se visualizar o programador utilizado para efectuar o *upload* das configurações.

Figura G.1 – Programador AVR-ISP500

Em termos de pinos este programador contém dois tipos de interligação o ICSP10 e o ICSP6. A Figura G.2 apresenta o *header* com a descrição dos pinos.



ICSP6		
pin	Abbrev.	description
1	MISO	Serial Input
2	V_TAR	Target VCC
3	SCK	Serial Clock
4	MOSI	Serial Output
5	TRST	Target RESET
6	GND	Ground

Figura G.2 – Pinos ICSP6

A Figura G.3 refere-se a um esquema electrónico de como devem ser efectuadas as ligações entre o ATMega168 e o *header* ICSP6.

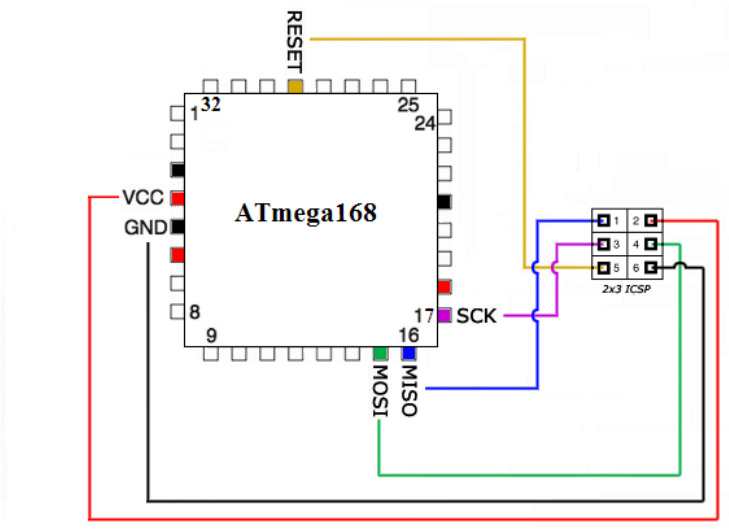


Figura G.3 – Interligação entre ATmega168 e ICSP6

Anexo H - Esboços dos módulos referentes ao nó sensor

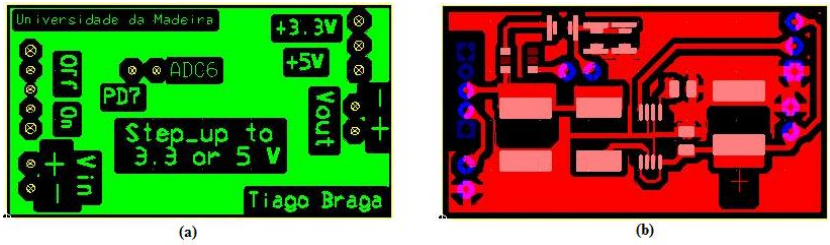


Figura H.1 - Esboço do módulo de alimentação: (a) – face superior; (b) – face inferior.

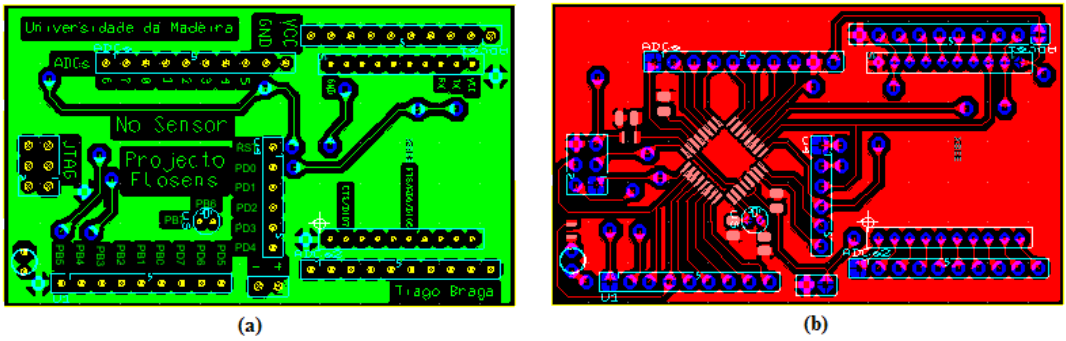


Figura H.2 - Esboço do módulo de controlo e comunicação: (a) – face superior; (b) – face inferior.

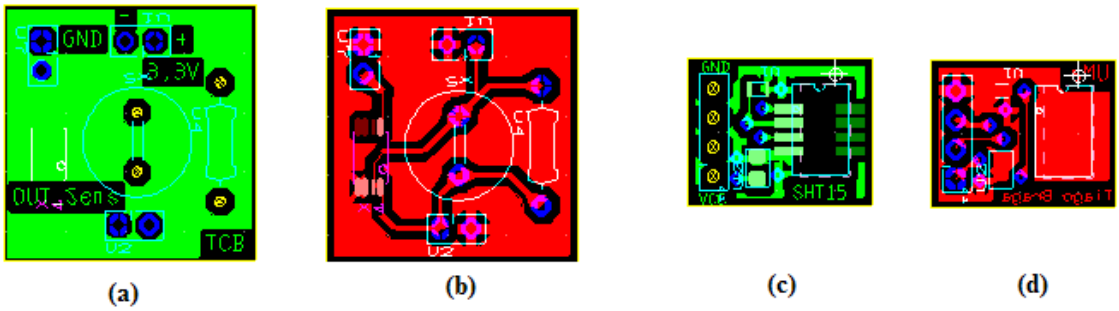


Figura H.3 - Esboços dos módulos de monitorização: (a) e (c) – face superior; (b) e (d) – face inferior.

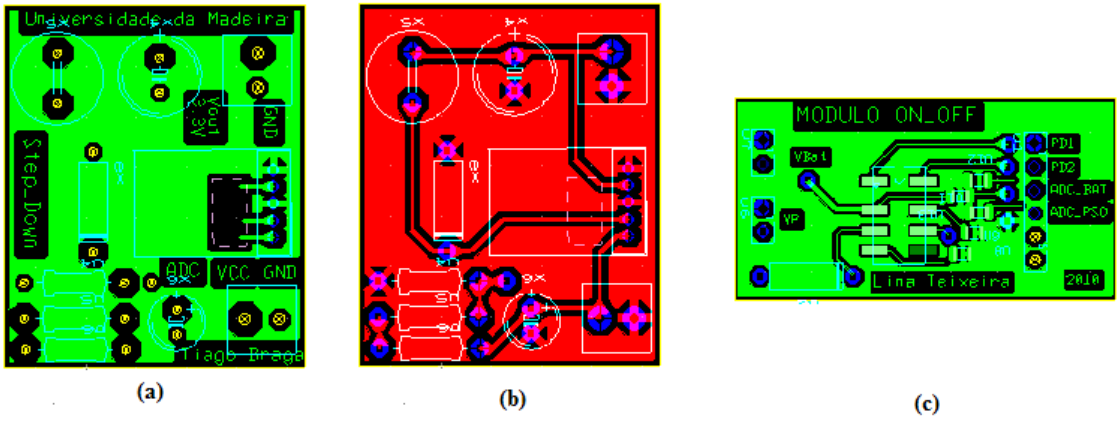
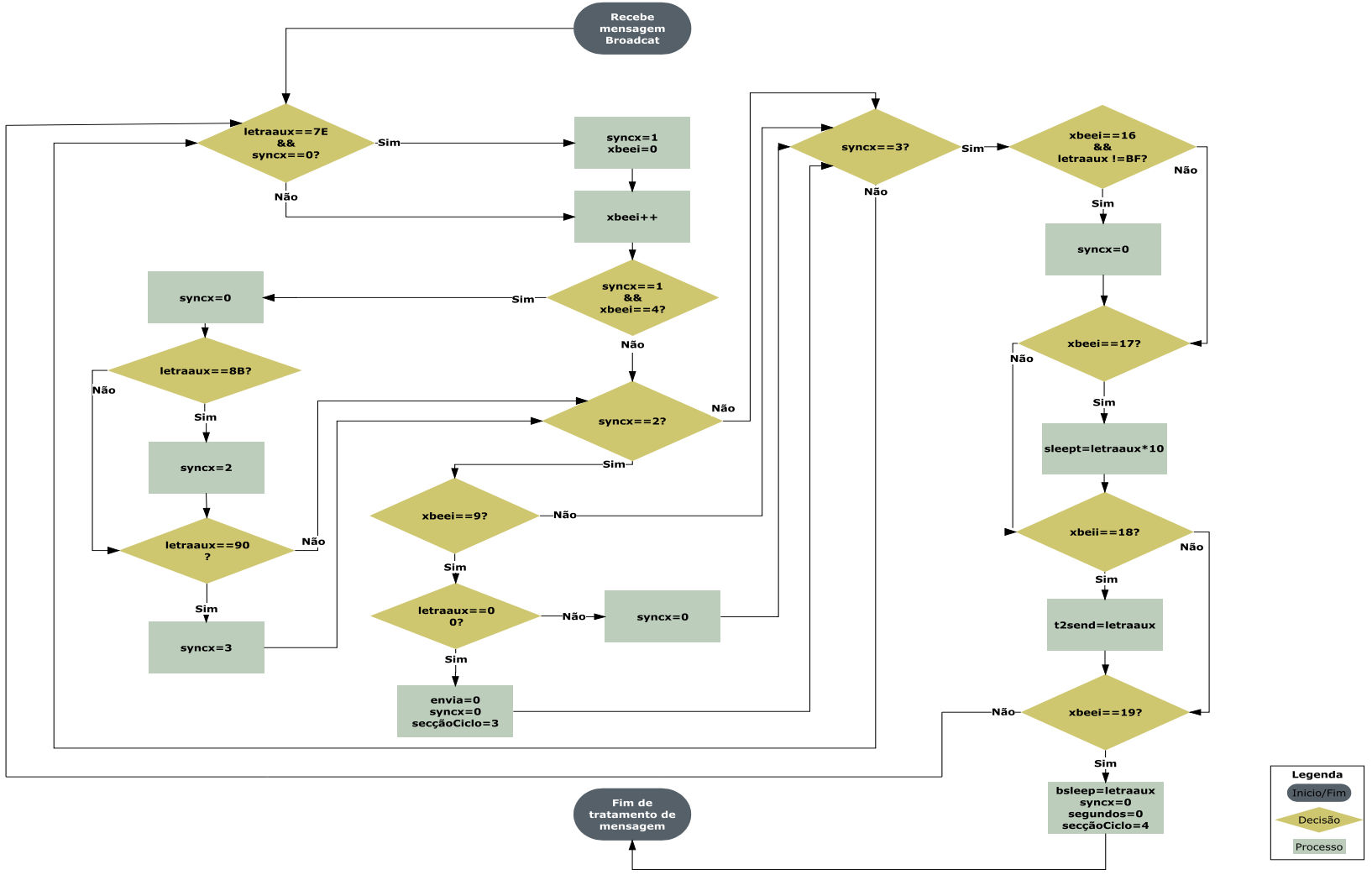


Figura H.4 - Esboços do módulo *Step_Down* (3.3V) e On_Off: (a) e (c) – face superior; (b) – face inferior.

Anexo I - Fluxograma para tratamento da mensagem de *broadcast*



Anexo J - Programação em linguagem C

```
/*****************************************************************************************
* File Name          : routerA1.c
* Author             : Filipe Santos, Tiago Braga
* Date First Issued  : 03/12/2009
* Description        : This file is used for the  ATMega168 to sense SHT1x and Battery voltage and communicate
            With the  XBee ZigBee module
*****************************************************************************************/
#define F_CPU 8000000UL
/********************************** LIVRARIAS ********************************************************/
#include <stdlib.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <avr/interrupt.h>
#include <avr/io.h>
#include <avr/sfr_defs.h>
#include <avr/power.h>
#include <avr/pgmspace.h>
#include <avr/sleep.h>
#include <util/delay.h>

/****************************** NOMENCLATURA E CONSTANTES ********************************/
#define XBEESLEEP          PD2       // Pino 9 sleep do XBee
#define LED                PB2       // Led de informação
#define MOSFET   PD7                 // Mosfet do divisor de tensão
#define    DATA     PB0              // Pino usado como linha DATA para o SHT15 (dados)
#define    SCK              PB1      // Pino usado como linha SCK para o SHT15 (relogio)
#define RESETINT           PD4
#define RELPIN1  PD3                 // Pino para comutar relé
#define RELPIN2  PD4                 // Pino para comutar relé

enum {TEMP,HUMI};                    // Tipo de amostra para sensor SHT1x

#define noACK 0                              // Nomenclatura para recepção de nível zero
#define ACK   1                              // Nomenclatura para recepção de nível um

// Comandos do sensor STH15
#define STATUS_REG_W 0x06                    //000   0011    0
#define STATUS_REG_R 0x07                    //000   0011    1
#define MEASURE_TEMP 0x03                    //000   0001    1
#define MEASURE_HUMI 0x05                    //000   0010    1
#define RESET1       0x1e                    //000   1111    0

/********************************** FUNÇÕES AUXILIARES ***************************************/
#define HI(x)                (x>>8)                  // Utilizado para obter o MSB de número de 16 bit
#define LO(x)                (x&0xff)
#define BSET(p,b) ((p) |=  (1<<b))                   // Set bit
#define BCLR(p,b)            ((p) &= ~(1<<b))        // Clear bit


/********************************** VARIÁVEIS GLOBAIS ****************************************/
volatile int segundos = 0, syncx = 0,XBeei=0,XBeeit=0,i=0;
volatile int sleept = 30, t2send = 10, bsleept = 30;          // Para ciclo de operação (segundos)
volatile char letraaux2 = 0x00;
volatile char msgXBEE[36];
volatile int seccaoCiclo = 1;
volatile int envia = 1;
volatile int aux = 1;
volatile int releState = 0;                                    // 0 => C. Aberto, 1=> C. Fechado
```

```
/****************************** APONTADORES DAS FUNÇÕES ************************************/
char s_write_byte(unsigned char value);
char s_read_byte(unsigned char ack);
void s_transstart(void);
void s_connectionreset(void);
char s_softreset(void);
char s_read_statusreg(unsigned char *p_value, unsigned char *p_checksum);
char s_write_statusreg(unsigned char *p_value);
char s_measure(unsigned char *p_value, unsigned char *p_checksum, unsigned char mode);
void USART_Init( unsigned int baud );
void USART_Transmit( unsigned char data );
unsigned char USART_Receive( void );
int measureADC(char adcN);
void sampleSend();
void configRTC(void);
unsigned char       checksumCalc (unsigned char * data_in,unsigned short      number_of_bytes_to_read, short
        array_index_counter);
void sendMsg (unsigned char * data_in,unsigned short      number_of_bytes_to_send);
void abreRele(void);
void fechaRele(void);


/************************************ PROGRAMA PRINCIPAL ****************************************/
int main()
{
        char msgXBSleepON[9] = {0x7E,0x00,0x05,0x08,0x01,0x53,0x4D,0x01,0x55};   // SM = 1
        char msgXBSleepOFF[9] = {0x7E,0x00,0x05,0x08,0x01,0x53,0x4D,0x00,0x56}; // SM = 0

        BSET(PORTD,RESETINT);                   // Pino de interrupção de reset para sincronismo
        BCLR(DDRD,RESETINT);                    // Coloca pino como input a interrupção de reset
        BSET(DDRB,LED);                         // Coloca o pino LED como output
        BCLR(PORTB,LED);                // Liga LED
        BSET(DDRD,XBEESLEEP);                   // Coloca o pino Sleep de XBee como output
        BSET(PORTD,XBEESLEEP);                  // Activa XBee = 0 (Entrar em Sleep = 1)
        BSET(DDRD,MOSFET);              // Coloca o pino GATE do Mosfet como output Divisor de Tensão
        BSET(PORTD,MOSFET);                     // Desactiva GATE = 1 (Activa = 0)
        BSET(DDRB,SCK);                         // Coloca o pino SCK como output
        BSET(DDRB,DATA);                        // coloca o pino DATA como output
        BCLR(PORTB,SCK);                        // Define nível 0 no SCK

        clock_prescale_set(0);                  // Define prescaler do relógio a 1 => uC a 8MHz
        ADMUX  = 0xC0;                          // Configura conversor ADC
        ADCSRA = 0x86;
        DIDR0  = 0xFF;                          // Poupança de energia

        // Incializacao dos pinos que controlam o relé

        BSET(DDRD,RELPIN1);             // Coloca o pino RELPIN1 como output
        BCLR(PORTD,RELPIN1);            // 0
        BSET(DDRD,RELPIN2);             // Coloca o pino RELPIN1 como output
        BCLR(PORTD,RELPIN2);            // 0
        fechaRele();                    // Coloca relé na posicão por defeito

        USART_Init(51);                 // 9600 8bit + 1 stopbit XBEE
        configRTC();

        _delay_ms(1);                           // Espera que cristal estabilize
        set_sleep_mode(SLEEP_MODE_PWR_SAVE);// Modo de adormecimento
        _delay_ms(1);

        BSET(PORTB,LED);                        // Acabou inicialização, desliga LED
        sei();                          // Activa interrupções

        /*
                SECCOES DO CICLO
                1 - XBEE e uC dormem
                2 - XBEE e uC acordam e esperam T2send, ao fim  este tempo enviam msg
```

```
                3 - Espera por broadcast
                4 - Conta T2sleep e dormem
        */
        for(;;)
        {
                if(segundos == sleept && seccaoCiclo == 1)
                {
                        envia = 1;
                                // Acorda XBEE: Tira o sleep e passa a router
                        BCLR(PORTD,XBEESLEEP);          // Sleep = 0
                        _delay_ms(13);
                        sendMsg((unsigned char*) &msgXBSleepOFF,9);
                        seccaoCiclo = 2;
                        i=t2send-5;
                }
                if((segundos == sleept+i) && (seccaoCiclo == 2) && (envia == 1))
                {
                        sampleSend();       // Amostra tensão de bat., temperat e humid, luminosidade
                                if(segundos == (sleept+t2send))
                                {
                                        seccaoCiclo = 3;
                                        segundos = 0;
                                        i = 0;
                                }
                                else
                                {
                                        i++;
                                }
                }

                if(segundos == bsleept && seccaoCiclo == 4)
                {
                                // Adormece XBEE: Troca para folha e activa o sleep
                        sendMsg((unsigned char*) &msgXBSleepON,9);
                        BSET(PORTD,XBEESLEEP);          // Sleep = 1
                        seccaoCiclo = 1;
                        segundos = 0;
                }

                if(seccaoCiclo == 1)
                {
                        cli();
                        sleep_enable();
                        sei();
                        _delay_us(10);
                }
        }
}

void fechaRele()
{
        BCLR(PORTD,RELPIN1);  // Activa bobine para colocar rele na posicao pretendida
        BSET(PORTD,RELPIN2);
        _delay_ms(10);
        BCLR(PORTD,RELPIN1);  // DESLIGA bobine do rele
        BCLR(PORTD,RELPIN2);
        //releState = 1;
}
void abreRele()
{
        BSET(PORTD,RELPIN1);  // Activa bobine para colocar rele na posicao pretendida
        BCLR(PORTD,RELPIN2);
        _delay_ms(10);
        BCLR(PORTD,RELPIN1);  // DESLIGA bobine do rele
        BCLR(PORTD,RELPIN2);
        releState = 0;
}
```

```
void sampleSend()
{
	BCLR(PORTB,LED);		// Liga LED

	unsigned char error=0,checksum=0;

	int j = 0, battery = 0, humidity = 0, temperature = 0, light = 0;

	char msgAPIXBee[27] ={0x7E,0x00,0x17,0x10,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFE,
				0x00,0x00,0xD4,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};


	abreRele();				// para medir a tensão real na bateria
	_delay_ms(50);

	// Mede sensores
	BCLR(PORTD,MOSFET);		// Activa mosfet
	battery = measureADC(0);	// Amostra tensão da BATERIA
	BSET(PORTD,MOSFET);		// Desactiva mosfet

	if(battery > 875) // fecha relé qd tensão baterias for inferior a 2.80V
	{
		abreRele();
	}
	else
	{
		if(releState == 0)
		{
			fechaRele();
		}

	}

	light = measureADC(1);	// Amostra tensão do sensor de luz

	error+=s_measure((unsigned char*) &humidity,&checksum,HUMI);		// Mede HUMIDADE
	error+=s_measure((unsigned char*) &temperature,&checksum,TEMP);	//Mede TEMPERATURA

	if(error!=0) s_connectionreset();		// Em caso de erro realiza reset ao sensor SHT1x

	msgAPIXBee[18] = HI(battery);
	msgAPIXBee[19] = LO(battery);
	msgAPIXBee[20] = HI(temperature);
	msgAPIXBee[21] = LO(temperature);
	msgAPIXBee[22] = HI(humidity);
	msgAPIXBee[23] = LO(humidity);
	msgAPIXBee[24] = HI(light);
	msgAPIXBee[25] = LO(light);

	msgAPIXBee[26] = checksumCalc((unsigned char*) &msgAPIXBee,23,3);

	for(j=0;j<27;j++)
	{
		USART_Transmit(msgAPIXBee[j]);
	}
	_delay_ms(2);			// Aguarda que a comunicação termine
	BSET(PORTB,LED);
}
/****************************************** INTERRUPÇÕES ******************************************/
// Interrupção do RTC
ISR(TIMER2_OVF_vect)
{
	segundos++;
	_delay_ms(1);
}
```

```
ISR(PCINT2_vect)
{
	_delay_ms(50);
	if(!bit_is_set(PIND,RESETINT))
	{
		BCLR(PCICR,PCIE2);		//  enable PCMSK2 pin change interrupts
		BCLR(PCMSK2,PD4);		//  tell pin change mask to listen to PD4 PCINT20
		configRTC();
		segundos = sleept;
		seccaoCiclo = 1;
	}
}


// Recebe msg de brodcast e actualiza sleept t2send bsleept
ISR(USART_RX_vect)
{
	letraaux2 = USART_Receive();

	if((letraaux2 == 0x7E) && (syncx == 0))
	{
		syncx = 1;
		XBeei = 0;
	}
	XBeei++;
	if((syncx == 1) && (XBeei == 4))
	{
		syncx=0;
		if(letraaux2 == 0x8B)
		{
			syncx=2;
		}
		if(letraaux2 == 0x90)
		{
			syncx=3;
		}
	}
	if(syncx == 2)
	{
		if(XBeei == 9)
		{			// Lê msg de feedback
			if(letraaux2 == 0x00)
			{
				envia = 0;
				syncx = 0;
				seccaoCiclo = 3;
			}
			else
			{
				syncx = 0;
			}
		}
	}


	if(syncx == 3)	// Lê msg com os novos tempos
	{
		if((XBeei == 16) && (letraaux2 != 0xBF) )
		{
			syncx = 0;
		}
		if(XBeei == 17)
		{
			sleept = letraaux2*10;
		}
		if(XBeei == 18)
```

```
                {
                        t2send = letraaux2;
                }
                if(XBeei == 19)
                {
                        bsleept = letraaux2;
                        syncx = 0;
                        segundos = 0;
                        seccaoCiclo = 4;
                }
        }
}

/************************************** Comunicação USART ************************************/
void USART_Init( unsigned int ubrr )
{
        UBRR0H = (unsigned char)(ubrr>>8);                      // Define velocidade comunicação
        UBRR0L = (unsigned char)ubrr;                           // Configura 8 bit + 1 de stop
        //UCSR0B = ((1<<RXEN0)|(1<<TXEN0));                     // Activa RX e TX
        UCSR0B = (1<<RXCIE0)|(1<<RXEN0)|(1<<TXEN0);             // Rx Tx enable (1<<RXCIE1)|
        UCSR0C = (3<<UCSZ00);
}

void USART_Transmit( unsigned char data )
{
        while ( !( UCSR0A & (1<<UDRE0)) );              // Espera que o buffer de transmissão esvazie
        UDR0 = data;                                    // Envia o byte
}

unsigned char USART_Receive( void )
{
        while ( !(UCSR0A & (1<<RXC0)) );                // Espera que o byte tenha sido recebido
        return UDR0;                                    // Devolve byte armazenado no buffer
}

/*************************** FUNÇÕES AUXILIARES ************************************/
int measureADC( char adcN )
{
        int aux1=0,aux2=0,auxt = 0;
        int i;
        ADMUX=0xC0+adcN;                                        // Selecciona o sensor a medir
        _delay_us(40);
        for(i=0;i<16;i++)
        {
                ADCSRA |= (1<<ADSC);                            // Inicia a conversão
                loop_until_bit_is_set(ADCSRA, ADIF);            // Espera que a conversão termine
                aux1 = ADCL;
                aux2 = ADCH;
                aux1 = (aux1);
                aux2 = (aux2<<8);
                aux1 = aux2+aux1;// ADCL + ADCH = 10bit's
                auxt = auxt+aux1;
        }
        auxt = (auxt>>4);                                       // Igual a / 16 sem arredondamento
        return auxt;
}

//checksumCalc -   Calcula o checksum para a msg API do XBee
unsigned char checksumCalc (unsigned char * data_in,unsigned short number_of_bytes_to_read,short array_index_counter)
{
        unsigned char       checksum;
        checksum = 0;
        number_of_bytes_to_read+=array_index_counter;
                while (array_index_counter != number_of_bytes_to_read)
                        {
                        array_index_counter++;
                        checksum += data_in[array_index_counter - 1];
```

```
                                }
            checksum = 0xFF-checksum;
              return checksum;
}


//sendMsg -    Envia vector de caracteres via UART
void sendMsg (unsigned char * data_in,unsigned short number_of_bytes_to_send)
{
            int j=0;
            for(j=0;j<number_of_bytes_to_send;j++)
            {
                        USART_Transmit(data_in[j]);
            }
            _delay_ms(1);
}



/*          configRTC -         Configura o timer 2 de 8 bit para contar segundo um cristal externo
                                Configura interrupção de overflow (1/32768 * 256 * 256 = 2)
 */
void configRTC(void)
{
            ASSR  =0x20;        // Activa modo assíncrono (recebe impulsos externos)
            TIMSK2=0x00;        // Desactiva todas a interrupções para alteração correcta dos registos
            OCR2A =0x00;        // Inicializa registo de comparacao
            OCR2B =0x00;        // Inicializa registo de comparacao
            TCCR2A=0x00;        // Configura modo normal
            TCCR2B=0x05;        // Inicia o timer com prescaler 05 => 128 cristal 32768 Overflow 128 => T = 1s
            TIMSK2=0x01;        // Activa interrupção de overflow
}

/*********************  Funções auxiliares do sensor SHT1x  ******************************/
// Writes a byte on the Sensibus and checks the acknowledge
char s_write_byte(unsigned char value){
  unsigned char i,error=0;
  for (i=0x80;i>0;i/=2){        // Shift bit for masking
            DDRB|=(1<<DATA);                    // DATA como output
            if (i & value){
                        PORTB|=(1<<DATA);   // Masking value with i , write to SENSI-BUS
            }else{
                        PORTB&=~(_BV(DATA));
            }
            _delay_us(1);
    PORTB|=(1<<SCK);        // Clk for SENSI-BUS
            _delay_us(5);                       // Pulsewidth approx. 5 us
    PORTB&=~(_BV(SCK));
  }
  DDRB|=(0<<DATA);                              // Release DATA-line
  PORTB|=(1<<DATA);
  DDRB&=~(_BV(DATA));
  PORTB|=(1<<SCK);                              // Clk #9 for ack
  if (bit_is_set(PINB,DATA)) error=3;
  else error=0;             // Check ack (DATA will be pulled down by SHT11)
  PORTB&=~(_BV(SCK));
  return error;             // Error=1 in case of no acknowledge
}

// Reads a byte form the Sensibus and gives an acknowledge in case of "ack=1"
char s_read_byte(unsigned char ack){
  unsigned char i,val=0;

  PORTB|=(1<<DATA);                     // release DATA-line
  DDRB&=~(_BV(DATA));       // release DATA-line
  for (i=0x80;i>0;i/=2)     // shift bit for masking
  { PORTB|=(1<<SCK);
```

```
            _delay_us(2);          // clk for SENSI-BUS
    if(bit_is_set(PINB,DATA)) val=(val | i);// read bit
    PORTB&=~(_BV(SCK));
            _delay_us(2);
  }
  DDRB|=(1<<DATA);
  if (ack){
            DDRB|=(1<<DATA);
            PORTB&=~(_BV(DATA));    //in case of "ack==1" pull down DATA-Line
  }

  PORTB|=(1<<SCK);          //clk #9 for ack
  _delay_us(5);
  PORTB&=~(_BV(SCK));
  PORTB|=(1<<DATA);                    //release DATA-line
  DDRB&=~(_BV(DATA));
  return val;
}

// Generates a transmission start
//       _____         ________
// DATA:      |_______|
//           ___     ___
// SCK : ___|   |___|   |______
void s_transstart(void){
            DDRB|=(1<<DATA);                        // DATA pin as output
            PORTB|=(1<<DATA); PORTB&=~(_BV(SCK));//Initial state
            _delay_us(2); // or _nop_();_nop_();
            PORTB|=(1<<SCK);
            _delay_us(2);
            PORTB&=~(_BV(DATA));
            _delay_us(2);
            PORTB&=~(_BV(SCK));
            _delay_us(5);
            PORTB|=(1<<SCK);
            _delay_us(2);
            PORTB|=(1<<DATA);
            _delay_us(2);
            PORTB&=~(_BV(SCK));
}

// Communication reset: DATA-line=1 and at least 9 SCK cycles followed by transstart
//       _____________________________________________________         ________
// DATA:                                                      |_______|
//           _   _   _   _   _   _   _   _   _       ___     ___
// SCK : __| |__| |__| |__| |__| |__| |__| |__| |_____|   |___|   |______
void s_connectionreset(void){
  unsigned char i;
  DDRB|=(1<<DATA);                                                  // DATA pin as output
  PORTB|=(1<<DATA);
  PORTB&=~(_BV(SCK));             // Initial state
  for(i=0;i<9;i++)                // 9 SCK cycles
  { PORTB|=(1<<SCK);
            _delay_us(3);
    PORTB&=~(_BV(SCK));
            _delay_us(1);
  }
  s_transstart();                 // transmission start
}

// Resets the sensor by a softreset
char s_softreset(void){
  unsigned char error=0;
  s_connectionreset();            // reset communication
  error+=s_write_byte(RESET1);      // send RESET-command to sensor
  return error;                   // error=1 in case of no response from the sensor
}
```

```
// Reads the status register with checksum (8-bit)
char s_read_statusreg(unsigned char *p_value, unsigned char *p_checksum){
  unsigned char error=0;
  s_connectionreset(); //s_transstart();                //transmission start
  error=s_write_byte(STATUS_REG_R); // send command to sensor
  *p_value=s_read_byte(ACK);        // read status register (8-bit)
  *p_checksum=s_read_byte(noACK);   // read checksum (8-bit)
  return error;                     // error=1 in case of no response from the sensor
}

// Writes the status register with checksum (8-bit)
char s_write_statusreg(unsigned char *p_value){
  unsigned char error=0;
  s_transstart();                   // transmission start
  error+=s_write_byte(STATUS_REG_W);// send command to sensor
  error+=s_write_byte(*p_value);    // send value of status register
  return error;                     // error>=1 in case of no response form the sensor
}

// Makes a measurement (humidity/temperature) with checksum

char s_measure(unsigned char *p_value, unsigned char *p_checksum, unsigned char mode){
  unsigned error=0;
  unsigned int i;

  s_transstart();                   // transmission start
  switch(mode){                     // send command to sensor
    case TEMP       : error+=s_write_byte(MEASURE_TEMP); break;
    case HUMI       : error+=s_write_byte(MEASURE_HUMI); break;
    default     : break;
  }

  for (i=0;i<20000;i++)
  {
            _delay_us(100);
            if(bit_is_clear(PINB,DATA)) break;
  } // Wait until sensor has finished the measurement
  if(bit_is_set(PINB,DATA)) error+=7;// or timeout (~2 sec.) is reached
  *(p_value+1)  = s_read_byte(ACK);  //read the first byte (MSB)
  *(p_value)    = s_read_byte(ACK);  //read the second byte (LSB)
  *p_checksum   = s_read_byte(noACK);//read checksum
  return error;
}

/*******************************  FIM DE PROGRAMA  ************************************/
```

Anexo K - Classes desenvolvidas no PARSER

Program.java

```
package fsparser;

import java.util.logging.Level;
import java.util.logging.Logger;

public class Program extends Thread {

    public static void run(int sessionId) {
        int i = 0;
        boolean deviceFound;
        System.out.println("A procurar dispositivo . . . ");
        while (!(deviceFound = Device.Search()) && i < 20) {
            i++;
            try {
                Thread.sleep(500);
            } catch (InterruptedException ex) {
                Logger.getLogger(Program.class.getName()).log(Level.SEVERE, null, ex);
            }
            System.out.print(" . ");
        }
        if (deviceFound){
            System.out.println("Dispositivo encontrado na porta: " + Device.GetPortName());
            Device.StartReading(sessionId);
            System.out.println("Recebendo dados...");

        }
        else{
            System.out.println("Dispositivo nao encontrado, a terminar tarefa!!!");
        }
    }

    public static void main(String[] args) {
        run(1);
    }
}
```

ComReader.java

```
package fsparser;

import java.io.*;
import java.util.*;
import gnu.io.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ComReader implements Runnable, SerialPortEventListener {

    private final static String DB = "flosense";
    InputStream inputStream;
    OutputStream outputStream;
```

```
    SerialPort serialPort;
    Thread readThread;
    Message msg;
    WelcomeMsg wmsg;
    int msg_len;
    int msg_type;
    MySQL conect;
    int broadcast;
    private int sessionId;
    private Vector<AccDataType> accs = new Vector<AccDataType>();

    public ComReader(SerialPort sp, int sessionId) {
        this.sessionId = sessionId;
        try {
            serialPort = sp;
            inputStream = sp.getInputStream();
            outputStream = sp.getOutputStream();
            serialPort.addEventListener(this);
//
            new Thread(new SyncThread(outputStream)).start();
            System.out.println("start");
        } catch (TooManyListenersException e) {
            System.out.println(e + " - " + e.getCause() + " - " + e.getStackTrace());
        } catch (IOException e) {
            System.out.println(e + " - " + e.getCause() + " - " + e.getStackTrace());
        }

        readThread = new Thread(this);
        msg = new Message();
        wmsg = new WelcomeMsg();
        conect = new MySQL();
        if (conect.Connect() == true) {
            System.out.println("Ligado ao servidor de BD.");
        } else {
            System.out.println("Erro ao ligar a base de dados.");
        }

        if (conect.EnterDatabase(DB) == false) {
            System.out.println("ERRO: Nao foi possivel ligar-se a Base de Dados");
            return;
        }

//        conect.CreateTables();
        msg_len = 0;
        msg_type = 0;
        broadcast = 0;
        readThread.start();
    }

    public void Stop() {
        System.out.println("ComReader Stoped!!!");
        serialPort.close();
    }

    public void run() {
            //System.out.println("ComReader running...");
        boolean bc = false;
        while (bc) {
            try {
                Thread.sleep(100);
```

```
            if (broadcast++ == 10) {
               throw new Exception("Broadcast Exception");
            }
         } catch (InterruptedException e) {
         } catch (Exception e) {
//             System.out.println("Ex: " + e + " - " + e.getCause());
            try {
               if (e.getMessage().equals("Broadcast Exception")) {
                  broadcast = 0;
                  System.out.println("      ---- BROADCASTING ----");
                  int[] b = {0x7E, 00, 0x10, 0x10, 00, 00, 00, 00, 00, 00, 00, 0xFF, 0xFF, 0xFF, 0xFE, 0x01, 00,
0xB1, 0x01, 0x41};


                  for (int i = 0; i < b.length; i++) {
                     outputStream.write(b[i]);
                  }
                  outputStream.flush();
               }
            } catch (IOException ioe) {
               System.out.println("IOex: " + ioe + " - " + ioe.getCause());
            }
         }
      }
   }

   public void serialEvent(SerialPortEvent event) {
      switch (event.getEventType()) {
         case SerialPortEvent.BI:
         case SerialPortEvent.OE:
         case SerialPortEvent.PE:
         case SerialPortEvent.CD:
         case SerialPortEvent.CTS:
         case SerialPortEvent.DSR:
         case SerialPortEvent.RI:
         case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
            break;
         case SerialPortEvent.DATA_AVAILABLE:

            try {

               while (inputStream.available() > 0) {
/
                  int data = inputStream.read();

                  if (data != 0x7E || msg_len > 0)
                  {
                                                      if (msg_len == 0) {
                        msg_len += data;

                     } else if (msg_type == 0) {
                        msg_type = data;

                        if (msg_type == 0x95) { // LIGOU-SE COM SUCESSO
                           wmsg.Process(0);
                           wmsg.Process(msg_len);
                           wmsg.Process(msg_type);
                        } else if (msg_type == 0x90) { // DADOS
                           msg.Process(0);
                           msg.Process(msg_len);
```

```
            msg.Process(msg_type);

        }else if (msg_type == 0x97) { // DADOS de RSSI
            msg.Process(0);
            msg.Process(msg_len);
            msg.Process(msg_type);

        } else if (msg_type == 0x88) { // Response ND command
            wmsg.Process(0);
            wmsg.Process(msg_len);
            wmsg.Process(msg_type);
            System.out.println("\nRespondeu ao comando ND o ");
        }

    } else if (msg_type == 0x92) { // Welcome Message

    } else if (msg_type == 0x95) { // Welcome Message

        wmsg.Process(data);

        if (wmsg.IsComplete() == true) {
            System.out.println("Dispositivo :" +
    wmsg.Convert64BitAddress(wmsg.Get64BitAddress()) + " conectou!");
            wmsg = new WelcomeMsg();
            msg_len = 0;
            msg_type = 0;
        }
    } else if (msg_type == 0x88) { // Welcome Message

        wmsg.Process(data);


        if (wmsg.IsComplete() == true) {
                        wmsg = new WelcomeMsg();
            msg_len = 0;
            msg_type = 0;
        }
    } else if (msg_type == 0x90) { // DADOS
        // System.out.println("Msg de dados recebida");

        msg.Process(data);
        if (msg.IsComplete() == true) {


            if msg.GetData()[0] == 0xD0) {
                msg = new AmbientD0Data(msg);
            }

            System.out.println(msg.toString());
            try {

                msg.AddToDatabase(conect, sessionId, accs);

            } catch (Exception ex) {
                Logger.getLogger(ComReader.class.getName()).log(Level.SEVERE, null, ex);
                System.out.println(ex + " - " + ex.getCause() + " - " + ex.getStackTrace());
            }
            //System.out.println("Msg de dados recebida");

            msg = new Message();
```

```
                    msg_len = 0;
                    msg_type = 0;
                }
            }else if (msg_type == 0x97) { // RSSI response

                msg.Process(data);
                if (msg.IsComplete() == true) {
                    msg = new rssiMsg(msg);
                    //System.out.println(msg.toString());
                    try {

                        msg.AddToDatabase(conect, sessionId, accs);

                    } catch (Exception ex) {
                        Logger.getLogger(ComReader.class.getName()).log(Level.SEVERE, null, ex);
                        System.out.println(ex + " - " + ex.getCause() + " - " + ex.getStackTrace());
                    }
                    //System.out.println("Msg de dados recebida");

                    msg = new Message();
                    msg_len = 0;
                    msg_type = 0;
                }
            }
        }
    }
} catch (IOException e) {
    System.out.println("IOex: " + e + " - " + e.getCause());
} catch (Exception e) {
    System.out.println("Ex: " + e + " - " + e.getCause());
}
break;
}
}
}
```

AmbientD0.java

```
package fsparser;

import java.text.DecimalFormat;
import java.util.Vector;

public class AmbientD0Data extends Message{

    public AmbientD0Data(Message msg) {
        super(msg);
    }

    public int getTemperature() {
        int temp = this.DATA[3] << 8;
        temp += this.DATA[4];
        return temp;
    }

    public int getHumidity() {
        int hum = this.DATA[5] << 8;
        hum += this.DATA[6];
```

```
        return hum;
    }

    public int getBattery() {
        int bat = this.DATA[1] << 8;
        bat += this.DATA[2];
        return bat;
    }
    @Override
    public void AddToDatabase(MySQL db, int sessionId, Vector<AccDataType> accs) throws Exception{
        try {
            String towerId = towerExists(db, sessionId, this.Get64BitAddress());
            String[] ident = new String[5];
            ident[0] = "towerId";
            ident[1] = "dateTime";
            ident[2] = "battery";
            ident[3] = "temperature";
            ident[4] = "humidity";

            String[] vals = new String[5];
            vals[0] = "" + towerId;
            vals[1] = DateUtils.GetCurrentDateTime();
            vals[2] = "" + this.getBattery();
            vals[3] = "" + this.getTemperature();
            vals[4] = "" + this.getHumidity();

            if (!db.InsertTableValue("ambientd0", ident, vals)){
                throw new Exception("Failed to insert Ambient measurement into the database");
            }
        } catch (Exception e) {
            throw new Exception("Exception: Relative to an interaction with the database - ", e);
        }
    }

    @Override
    public String toString() {
        int t = this.getTemperature(),
                h = this.getHumidity(),
                b = this.getBattery();
                                    double temp = t * 0.01 - 40,
                hum = (temp - 25) * (0.01 + 0.00008 * t) + (-4 + 0.0405 * h - 0.0000028 * h * h),
                bat = (b * 1.1 / 1024) * 3;

        DecimalFormat df = new DecimalFormat("0.00");
        String result = "Length: " + this.LENGTH;
        result += " Code: " + this.CODE;
        result += " 64BitAddress: " + this.SIXTY_FOUR_BITADDRESS;
        result += " 16BitAddress: " + this.SIXTEEN_BITADDRESS;
        result += " Option: " + this.OPTION;
        result += " Temperature: " + df.format(temp);//temp;
        result += " Humidity: " + df.format(hum);
        result += " Battery: " + df.format(bat);
        result += " CheckSum: " + this.CHECK_SUM;
        return result;
    }
}
```

Device.java

```
package fsparser;

import java.util.Enumeration;
import gnu.io.CommPortIdentifier;
import gnu.io.PortInUseException;
import gnu.io.SerialPort;
import gnu.io.UnsupportedCommOperationException;


public class Device {
        private static CommPortIdentifier  portId;
    private static Enumeration<?>                   portList;
    private static SerialPort           serialPort;
        private   static ComReader                  leitor;

  public static void StartReading(int sessionId)
  {
      OpenSerialPort();
      leitor = new ComReader(serialPort, sessionId);
  }

  public static void StopReading()
  {
          leitor.Stop();
      serialPort.close();
  }

  public static void OpenSerialPort(){
     try {
        serialPort = (SerialPort) portId.open("Sensores UMA", 2000);
        serialPort.notifyOnDataAvailable(true);
        serialPort.setSerialPortParams(9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);
     } catch (PortInUseException e) {
        System.out.println(e + " - " + e.getCause() + " - " + e.getStackTrace());
     } catch (UnsupportedCommOperationException e) {
        System.out.println(e + " - " + e.getCause() + " - " + e.getStackTrace());
     }
  }

  public static String GetPortName()
  {
          return portId.getName();
  }

        public static boolean Search()
        {
                boolean deviceFound = false;
                String port = GetDevicePort();
                if(port != null)
                {
                        deviceFound = DeviceExists(port);
                }
                return deviceFound;
        }

        private static boolean DeviceExists(String DevicePort)
   {
        boolean DeviceFound = false;
        SpamFilter.Enable();
```

```
        portList = CommPortIdentifier.getPortIdentifiers();
        SpamFilter.Disable();
        while(portList.hasMoreElements())
        {
                portId = (CommPortIdentifier)portList.nextElement();
                if(portId.getPortType() == CommPortIdentifier.PORT_SERIAL)
                {
            if(portId.getName().equals(DevicePort))
                        {
                        DeviceFound = true;
                        break;
                        }
         }
        }
        return DeviceFound;
   }

   private static String GetDevicePort()
   {
        String ID = "";
        for(int n = 0; ID != null; n++)
        {
                ID = RegQuery.getInstanceID(n);

                if(ID != null && (ID.contains("Pid_6001") || ID.contains("PID_6001")))
                {
            break;
                }
        }
        if(ID != null)
        {
                ID = RegQuery.getFriendlyName(ID);

                if(ID != null && ID.length() >= 22)
                {
                        ID = ID.substring(17, ID.length() -1);
                }
                else { ID = null; }
        }
        return ID;
   }
}
```

SyncThread,java

```
package fsparser;

import java.io.IOException;
import java.io.OutputStream;
import java.util.logging.Level;
import java.util.logging.Logger;


public class SyncThread implements Runnable {

   private OutputStream stream;
```

```
private final long MAX_TARDINESS = 300000; //para 5 minutos


  private long next_execution;

  SyncThread(OutputStream s) {
     stream = s;
     next_execution = System.currentTimeMillis() + MAX_TARDINESS;
  }

  public void run() {

     System.out.println("tou correndo");
     next_execution = System.currentTimeMillis() + MAX_TARDINESS;
     boolean aux = true;
     long next = System.currentTimeMillis() + 1000;
     long inicio = System.currentTimeMillis();
     int ciclos = 0;
     while (true)
                    {

        if (System.currentTimeMillis() - next_execution > 0) {
           ciclos++;
           // Perform the task
           if(ciclos == 2){
              ciclos = 0;
              // Pede RSSI
              sendRSSIMsg();
              try{
                 Thread.sleep(500); // Espera 500ms pela resposta

              }catch (InterruptedException ie){
                 System.out.println(ie.getMessage());
              }
              sendMessage();

           }else{
              // Envia broadcast
              sendMessage();
           }
           next_execution = System.currentTimeMillis() + MAX_TARDINESS;
           aux = false;
           inicio = System.currentTimeMillis();
        }

        if ((System.currentTimeMillis() == next)) {

           System.out.print((System.currentTimeMillis() - inicio) / 1000);

           if (((System.currentTimeMillis() - inicio) / 1000) == 60) {
              System.out.println(",");}
           else if
              (((System.currentTimeMillis() - inicio) / 1000) == 119) {
              System.out.println(",");}

            else {
              System.out.print(",");
           }
           next = System.currentTimeMillis() + 1000;
```

```
        }
    }
}


    private void sendMessage() {

        //t2send = 35s = 0x23
        //tbsleep = 5s = 0x05
        // sleept = 260s = 0x1A x 10

  char[] d = {0x7E, 0x00, 0x12, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFE,
0x00, 0x00, 0xBF, 0x1a, 0x26, 0x02, 0xf3}; // para 5 minutos


        byte[] b = new byte[d.length];
        for (int i = 0; i < d.length; i++) {
            byte aux = (byte) d[i];
            b[i] = aux;
        }
        try {
            stream.write(b);
            stream.flush();
            System.out.println("Já mandei Broadcast");
        } catch (IOException e) {
            System.out.println("IOex: " + e + " - " + e.getCause());
        } catch (Exception e) {
            System.out.println("Ex: " + e + " - " + e.getCause());
        }
    }

    private void sendRSSIMsg() {

        // broadcast para RSSI
        char[] d = {0x7E, 0x00, 0x0F, 0x17, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFE,
0x01, 0x44, 0x42, 0x65};

        byte[] b = new byte[d.length];
        for (int i = 0; i < d.length; i++) {
            byte aux = (byte) d[i];
            b[i] = aux;
        }
        try {
            stream.write(b);
            stream.flush();
            System.out.println("Pedido de RSSI");
        } catch (IOException e) {
            System.out.println("IOex: " + e + " - " + e.getCause());
        } catch (Exception e) {
            System.out.println("Ex: " + e + " - " + e.getCause());
        }
    }
}
```

DateUtils.java

```
package fsparser;
```

```
import java.text.ParseException;
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateUtils {
        public static final String DATE_FORMAT_NOW = "yyyy-MM-dd HH:mm:ss";

        public synchronized static String GetCurrentDateTime() {
                Calendar cal = Calendar.getInstance();
                SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT_NOW);
                return sdf.format(cal.getTime());
        }

    public synchronized static Date parseDate(String date){
       SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT_NOW);
       try {
          return sdf.parse(date);
       } catch (ParseException ex) {
          return null;
       }
    }

    public synchronized static Date currentToDate(){
       return parseDate(GetCurrentDateTime());
    }
}
```

Anexo L - Código para aplicação da página de Internet

index.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
<title>Project - WSN monitoring - by Mathematics and Engineering Department, University of Madeira</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<META name="description" content="Publications from members of the Department of Mathematics and Engineering of University
of Madeira.">
<META name="keywords" content="Publications,DME,UMa,University,Madeira">

<link type="text/css" href="calendar/css/smoothness/jquery-ui-1.7.2.custom.css" rel="stylesheet" />
                        <script type="text/javascript" src="calendar/js/jquery-1.3.2.min.js"></script>
                        <script type="text/javascript" src="calendar/js/jquery-ui-1.7.2.custom.min.js"></script>
                        <script type="text/javascript">
                                $(function(){
                                        // Datepicker
                                        $('#datepicker').datepicker({
                                                inline: true
                                        });
                                        // Datepicker 2
                                        $('#datepicker2').datepicker({
                                                inline: true
                                        });
                                        // Datepicker 3
                                        $('#datepicker3').datepicker({
                                                inline: true
                                        });
                                        // Datepicker 4
                                        $('#datepicker4').datepicker({
                                                inline: true
                                        });


                                });
                        </script>

<style type="text/css" media="all">

                @import "forms.css";
/*
** Header Styles
*/

.textBox{
                font-size:x-small;
}
#header {
  background-color: #219CD5;
  margin: 0px;
  border: 1px solid #72a2cb;
}

#header #logo {
                text-align: left;
                width:89px;
}

#header #DME {
                color:White;
                font-stretch:condensed;
                font-weight:bolder;
}

#header #UMA a {
                color:White;
                font-size:small;
```

```
		font-stretch:condensed;
		text-decoration: none;
}

#header #UMA a:hover {
		color: #456699;
		text-decoration: underline;
}

#header #menu {
		color: white;
		font-size: small;
		vertical-align: top;
		text-align: right;
}

#header #menu a {
		color: white;
		text-decoration: none;
}

#header #menu a:hover {
		color: #456699;
		text-decoration: underline;
}
/*
** Navigation Bar Styles
*/
#navigation {
		color: white;
		background-color: #456699;
		border-top: 4px solid White;
		border-bottom: 4px solid White;
		padding: 3px 5px 3px 5px;
		text-align: left;
		font-size: small;
}

#navigation a {
		color: white;
		text-decoration:none;
}

#navigation a:hover {
		color: #C6DAEC;
		text-decoration: underline;
}
#wrapper {
		width:1180px;
		background-color: White;
		margin-left: auto;
		margin-right: auto;
}
BODY {
 background-color: #DCDDCB;
 text-align: left;
 font-family: Verdana, Arial, sans-serif;
 font-size: x-small;
 margin: 10px;
 border: 0px solid White;
}
</style>

</head>

<body>
<TABLE cellpading="0" cellspacing="0" border="0" style="margin-left: auto; margin-right: auto;">
<TR>
<TD>
<TABLE  id="wrapper" align=center cellpading="0" cellspacing="0" border="0" style="border:0px;border:10px solid white;">
 <TR>
  <TD id="header">
		<table width="100%" cellpadding="0" cellspacing="0" border="0">
			<tr>
				<td id="logo" rowspan="2">
					<img src="images/UMadeira01.jpg" alt="banner" width="230" height="74"/>
```

```
                                        </td>
                                        <td valign="middle" rowspan="2">
                        <table cellpadding="0" cellspacing="0">
                <tr><td id="DME"><font style="font-size:20px;">WSN Monitoring</font> - <a href="http://dme.uma.pt"
target="_blank"  style="font-size:16px;color:#ffffff;text-decoration:none;">EXACT SCIENCES AND
ENGINEERING</a></td></tr>
<tr><td id="UMA"><a href="http://www.uma.pt">UNIVERSITY OF MADEIRA</a></td></tr>
                        </table>
                        </td>
                        <td id="menu">

                        <font style='font-size:10px;'>
                        <a href="mailto:jara@uma.pt" alt="Contact us">Prof. Amândio Azevedo</a><br />
                        <a href="mailto:fsantos@uma.pt" alt="Contact us">Filipe Santos</a><br />
                        <a href="mailto:tcbraga@gmail.com" alt="Contact us">Tiago Braga</a><br />
                        <a href="mailto:lmfteixeira@hotmail.com" alt="Contact us">Lina Teixeira</a></font>

                                        </td>
                                </tr>
                                <tr>
                         </tr>
                        </table>
  </TD>
 </TR>

 <tr>
  <td id="navigation">
                <?php
                $ipservidor = 'localhost';
                require('mysql4.php');
                $dbObject = @new sql_db($ipservidor, 'root', '', 'flosense', false);
                //var_dump($dbObject);
                $motesIdList = array();
                $aQueryResult = $dbObject->sql_query("SELECT dateTime FROM ambientd0 ORDER BY dateTime DESC limit 1;");
                if($dbObject->sql_numrows()>0 && $row = mysql_fetch_array($aQueryResult, MYSQL_ASSOC)){
                        $lastUpdateDate = $row['dateTime'];

                }
                ?>


                Project FloSense - Stress test on XBee ZigBee Network (last updated: <?php echo $lastUpdateDate; ?>)
  </td>
 </tr>
<tr>
                <td>
                        <table cellpadding="0" cellspacing="0" border="0" style="width:100%;margin-left: auto; margin-right: auto;">
                                <tr>
                                        <td style="width:48%">
                                                <table cellpadding="0" cellspacing="0" border="0" style="margin-left: auto;
margin-right: auto;">
                                                        <tr>
                                                          <td colspan="4">
                                                            <b>Sensor nodes location</b>
                                                          </td>
                                                        </tr>
                                                        <tr>
                                                           <td>
                <!-- GOOGLE MAPS  -->
<iframe align="middle" width=480px height=450px src="phpsqlajax_map.htm" name="mapa" scrolling="no"
frameborder="0"></iframe>

                                                           </td>
                                                        </tr>
                                                </table>
                                        </td>
                                        <td style="width:52%" valign="top">
                                <table cellpadding="0" cellspacing="0" border="0" style="margin-left: auto; margin-right: auto;">
                                                        <tr>
                                                          <td colspan="4" valign="top">
                                <table cellpadding=0 cellspacing=0 border=0>
                                        <tr>
                                                <td>

                                <?php
```

```
                                        // busca lista de nós activos
                                        require('mysql4.php');
                                        $dbObject = @new sql_db($ipservidor , 'root', '', 'flosense', false);

                                        $motesIdList = array();
                                        $motesColor = array();
                $aQueryResult = $dbObject->sql_query('SELECT distinct(a.towerId),t.cor FROM ambientd0 a, tower t WHERE t.towerId =
a.towerId ORDER BY towerId ASC;');
                        while($dbObject->sql_numrows()>0 && $row = mysql_fetch_array($aQueryResult, MYSQL_ASSOC)){
                                if($row['towerId'] != 14 && $row['towerId'] != 102 && $row['towerId'] != 103 && $row['towerId'] != 104){
                                                $motesIdList[] = $row['towerId'];
                                                $motesColor[$row['towerId']] = $row['cor'];
                                        }
                                }
                                if(count($motesIdList) == 0){
                                        $motesColor = array();
                                        $motesIdList = array();
                                }
                                $dbObject->sql_close();
                                        $h = 3;
                                        if(isset($_GET['h']))
                                        {
                                                        $h = $_GET['h'];
                                        }

                                        echo '<div align="right"><!-- Select the data for graph 1: -->
                                        <form id="form" action="graph2.php" method="post" target="graph" -->
                                        Start <input id="datepicker" type="text" name="data" value="'.date("d-m-Y").'" class="textBox"
style="margin-bottom:3px;"size="10" style="text-align:center;border:1px solid #606060;" /><br />
                                        End <input id="datepicker2" type="text" name="enddata" value="'.date("d-m-Y").'"
class="textBox" style="margin-bottom:3px;"size="10" style="text-align:center;border:1px solid #606060;" />';

                                                echo '<br />
                                                <table cellpadding=0 cellspacing=0 border=0><tr><td style="text-
align:right;padding-right:4px;">
                                        Hold CTRL to select multiple motes
                                        </td><td><select id="motelist" multiple size="9" name="motelist[]" onchange="submit();"
class="textBox" style="margin-bottom:3px;width:70px;">';

                                        $motesid = array();

                                        foreach ($motesIdList as $mid){
                                                $midaux = $mid;

                                                $motesid[$mid] = "Mote ".$midaux;
                                        }
                                        $moteIdc =15;
                                        if(isset($_POST['moteidc'])){
                                                $moteIdc = $_POST['moteidc'];
                                        }
                                        foreach ($motesid as $j => $ch){
                                                if($j == $moteIdc){
                                                        echo '<option value="'.$j.'" selected
style="color:'.$motesColor[$j].';">'.$ch.'</option>';
                                                }else{
                                                        echo '<option value="'.$j.'"
style="color:'.$motesColor[$j].';">'.$ch.'</option>';
                                                }
                                        }

                                        echo '</select></td></tr></table>';

                                        echo '<br /><select id="channel" name="channel" onchange="submit();" class="textBox"
style="margin-bottom:3px;">';
                                        $channels = array(0=>'Humidity',
                                                          1=>'Temperature',
                                                          2=>'Light',
                                                          3=>'Volt',
                                                          4=>'Corrente PainelSolar',
                                                          5=>'Corrente Aerogerador',
                                                          6=>'Potencia',
                                                          7=>'RSSI');

                                        $channel = 0;
                                        if(isset($_POST['channel'])){
```

```
                                        $channel = $_POST['channel'];
                                }
                                foreach ($channels as $j => $ch){
                                        if($j == $channel){
                                                echo '<option value="'.$j.'" selected>'.$ch.'</option>';
                                        }else{
                                                echo '<option value="'.$j.'" >'.$ch.'</option>';
                                        }
                                }

                                echo '</select>';




                                echo '<br />
                                <input type="submit" value="Search >>" style="border:1px solid gray;" class="textBox" />
                                </form>
                                </div>';

                                ?>
                                                </td>
                                                <td>

                                                        <?php
                                                        //include('graph2.php'); ?>
<iframe height="250" width="410" src="graph2.php?channel=0" name="graph" scrolling="no" frameborder="0" ></iframe>
                                                </td>
                                        </tr>
                                </table>
                                </td>
                                </tr>
                                <tr>
                                        <td valign="top">

                                <?php
                                echo '<div align="right"><br /><!-- Select the data for graph 1: -->
                                <form id="form" action="graph2.php?leg=1" method="post" target="graph2" -->
                                Start <input id="datepicker3" type="text" name="data" value="'.date("d-m-Y").'"
class="textBox" style="margin-bottom:3px;"size="10" style="text-align:center;border:1px solid #606060;" /><br />
                                End <input id="datepicker4" type="text" name="enddata" value="'.date("d-m-Y").'"
class="textBox" style="margin-bottom:3px;"size="10" style="text-align:center;border:1px solid #606060;" />';

                                        echo '<br />
                                        <table cellpadding=0 cellspacing=0 border=0><tr><td style="text-
align:right;padding-right:4px;">
                                Hold CTRL to select multiple motes
                                </td><td><select id="motelist" multiple size="9" name="motelist[]" onchange="submit();"
class="textBox" style="margin-bottom:3px;width:70px;">';

        $motesid = array();

        foreach ($motesIdList as $mid){
                $midaux = $mid;
        $motesid[$mid] = "Mote ".$midaux;
                        }
                $moteIdc =15;
                if(isset($_POST['moteidc'])){
                $moteIdc = $_POST['moteidc'];
                        }
                        foreach ($motesid as $j => $ch){
                        if($j == $moteIdc){
                                echo '<option value="'.$j.'" selected style="color:'.$motesColor[$j].';">'.$ch.'</option>';
                                }else{
                                echo '<option value="'.$j.'" style="color:'.$motesColor[$j].';">'.$ch.'</option>';
                                }
                        }

        echo '</select></td></tr></table>';

        echo '<br /><select id="channel" name="channel" onchange="submit();" class="textBox" style="margin-bottom:3px;">';
                                $channels = array(0=>'Humidity',
                                                1=>'Temperature',
                                                2=>'Light',
                                                3=>'Volt',
                                                4=>'Corrente PainelSolar',
```

```
                                                5=>'Corrente Aerogerador',
                                                6=>'Potencia',
                                                7=>'RSSI');

                                        $channel = 0;
                                        if(isset($_POST['channel'])){
                                                $channel = $_POST['channel'];
                                        }
                                        foreach ($channels as $j => $ch){
                                                if($j == $channel){
                                                        echo '<option value="'.$j.'" selected>'.$ch.'</option>';
                                                }else{
                                                        echo '<option value="'.$j.'" >'.$ch.'</option>';
                                                }
                                        }

                                        echo '</select>';

                                        echo '<br />
                                        <input type="submit" value="Search >>" style="border:1px solid gray;" class="textBox" />
                                        </form>
                                        </div>';

                                        ?>
                                                        </td>
                                                        <td>
                                                                <!-- GRAPH 2 -->

 <iframe height="300" width="410" src="graph2.php?channel=0&leg=1" name="graph2" scrolling="no" frameborder="0"
></iframe>
                                                          </td>
                                                        </tr>
                                                </table>
                                        </td>
                                        </tr>
                                        </table>

                                        </td>
                                </tr>
                        </table>

                </td>
</tr>
</table>
</body>
</html>
```

phpsqlajax_map.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps AJAX + MySQL/PHP Example</title>


    <script src="http://maps.google.com/maps?file=api&v=2&key=ABQIAAAAQm_MzkZh5PK_z_tFcQ6LDxSufY50HmhrM-
oY3lybDTAeZtOcexT-tudezTX4qVD3WpvtMs0UP1Ncxw"
            type="text/javascript"></script>

    <script type="text/javascript">
    //<![CDATA[


    var iconCoordenador = new GIcon();
    iconCoordenador.image = 'imagens/red_MarkerC.png';
    iconCoordenador.shadow = 'http://labs.google.com/ridefinder/images/mm_20_shadow.png';
    iconCoordenador.iconSize = new GSize(12, 20);
    iconCoordenador.shadowSize = new GSize(22, 20);
```

```
iconCoordenador.iconAnchor = new GPoint(6, 20);
iconCoordenador.infoWindowAnchor = new GPoint(5, 1);

var iconRouter = new GIcon();
iconRouter.image = 'imagens/darkgreen_MarkerR.png';
iconRouter.shadow = 'http://labs.google.com/ridefinder/images/mm_20_shadow.png';
iconRouter.iconSize = new GSize(12, 20);
iconRouter.shadowSize = new GSize(22, 20);
iconRouter.iconAnchor = new GPoint(6, 20);
iconRouter.infoWindowAnchor = new GPoint(5, 1);


var iconRouter15 = new GIcon();
iconRouter15.image = 'images/15.gif';
iconRouter15.shadow = 'http://labs.google.com/ridefinder/images/mm_20_shadow.png';
iconRouter15.iconSize = new GSize(12, 20);
iconRouter15.shadowSize = new GSize(22, 20);
iconRouter15.iconAnchor = new GPoint(6, 20);
iconRouter15.infoWindowAnchor = new GPoint(5, 1);

var iconRouter16 = new GIcon();
iconRouter16.image = 'images/16.gif';
iconRouter16.shadow = 'http://labs.google.com/ridefinder/images/mm_20_shadow.png';
iconRouter16.iconSize = new GSize(12, 20);
iconRouter16.shadowSize = new GSize(22, 20);
iconRouter16.iconAnchor = new GPoint(6, 20);
iconRouter16.infoWindowAnchor = new GPoint(5, 1);

var iconRouter17 = new GIcon();
iconRouter17.image = 'images/17.gif';
iconRouter17.shadow = 'http://labs.google.com/ridefinder/images/mm_20_shadow.png';
iconRouter17.iconSize = new GSize(12, 20);
iconRouter17.shadowSize = new GSize(22, 20);
iconRouter17.iconAnchor = new GPoint(6, 20);
iconRouter17.infoWindowAnchor = new GPoint(5, 1);

var iconRouter18 = new GIcon();
iconRouter18.image = 'images/18.gif';
iconRouter18.shadow = 'http://labs.google.com/ridefinder/images/mm_20_shadow.png';
iconRouter18.iconSize = new GSize(12, 20);
iconRouter18.shadowSize = new GSize(22, 20);
iconRouter18.iconAnchor = new GPoint(6, 20);
iconRouter18.infoWindowAnchor = new GPoint(5, 1);

var iconRouter21 = new GIcon();
iconRouter21.image = 'images/21.gif';
iconRouter21.shadow = 'http://labs.google.com/ridefinder/images/mm_20_shadow.png';
iconRouter21.iconSize = new GSize(12, 20);
iconRouter21.shadowSize = new GSize(22, 20);
iconRouter21.iconAnchor = new GPoint(6, 20);
iconRouter21.infoWindowAnchor = new GPoint(5, 1);

var iconRouter22 = new GIcon();
iconRouter22.image = 'images/22.gif';
iconRouter22.shadow = 'http://labs.google.com/ridefinder/images/mm_20_shadow.png';
iconRouter22.iconSize = new GSize(12, 20);
iconRouter22.shadowSize = new GSize(22, 20);
iconRouter22.iconAnchor = new GPoint(6, 20);
iconRouter22.infoWindowAnchor = new GPoint(5, 1);

var iconRouter23 = new GIcon();
iconRouter23.image = 'images/23.gif';
iconRouter23.shadow = 'http://labs.google.com/ridefinder/images/mm_20_shadow.png';
iconRouter23.iconSize = new GSize(12, 20);
iconRouter23.shadowSize = new GSize(22, 20);
iconRouter23.iconAnchor = new GPoint(6, 20);
iconRouter23.infoWindowAnchor = new GPoint(5, 1);

var iconRouter24 = new GIcon
iconRouter24.image = 'images/24.gif';
iconRouter24.shadow = 'http://labs.google.com/ridefinder/images/mm_20_shadow.png';
iconRouter24.iconSize = new GSize(12, 20);
iconRouter24.shadowSize = new GSize(22, 20);
iconRouter24.iconAnchor = new GPoint(6, 20);
iconRouter24.infoWindowAnchor = new GPoint(5, 1);
```

```
    var customIcons = [];
    customIcons["Coordenador"] = iconCoordenador;
    customIcons["Router"] = iconRouter;
    customIcons["Router15"] = iconRouter15;
    customIcons["Router16"] = iconRouter16;
    customIcons["Router17"] = iconRouter17;
    customIcons["Router18"] = iconRouter18;
    customIcons["Router21"] = iconRouter21;
    customIcons["Router22"] = iconRouter22;
    customIcons["Router23"] = iconRouter23;
    customIcons["Router24"] = iconRouter24;

    function load() {
      if (GBrowserIsCompatible()) {

        var map = new GMap2(document.getElementById("map"));
              // map.setUIToDefault();
       // map.addControl(new GSmallMapControl());
                            map.addControl(new GScaleControl());
         //map.addControl(new GMapTypeControl());

                  map.setCenter(new GLatLng(32.661308,-16.924850), 16);
                  //map.disableDragging();
                  map.disableDoubleClickZoom();
                  map.disableScrollWheelZoom();


                            map.setMapType(G_SATELLITE_MAP);


        GDownloadUrl("phpsqlajax_genxml.php", function(data) {
          var xml = GXml.parse(data);
          var markers = xml.documentElement.getElementsByTagName("marker");
          for (var i = 0; i < markers.length; i++) {
            var id = markers[i].getAttribute("id");
            var address = markers[i].getAttribute("address");
            var tipo = markers[i].getAttribute("tipo");
            var point = new GLatLng(parseFloat(markers[i].getAttribute("lat")),
                                parseFloat(markers[i].getAttribute("lng")));
            var marker = createMarker(point, id, address, tipo);

            map.addOverlay(marker);
          }
        });
      }
    }

    function createMarker(point, id, address, tipo) {
      var marker = new GMarker(point, customIcons[tipo]);
      var html =  "<iframe src='teste/tabela.php?idtower="+id+"' width='180' height='120' frameborder='0'></iframe>";
      GEvent.addListener(marker, 'click', function() {
        marker.openInfoWindowHtml(html);
      });
      return marker;
    }
    function createMarkerOld(point, id, address, tipo) {
      var marker = new GMarker(point, customIcons[tipo]);
      var html =  "<b>"+tipo+"<br/>"+"Mote ID: "+id+"</b> <br/>" + "ID: "+ ""+address+"<iframe src='teste/tabela.php?idtower="+id+"'
width='110' height='105'></iframe>";
      GEvent.addListener(marker,'click', function() {
        marker.openInfoWindowHtml(html);
      });
      return marker;
    }

  </script>

  </head>

  <body onload="load()" onunload="GUnload()">

    <div id="map" style="width:640px; height:500px;"></div>
</body>
</html>
```

phpsqlajax_genxml.php

```
<?php


//$username="root";
//$password="";
//$database="flosense";
require('dbinfo.php');

//// Start XML file, create parent node
//$doc = domxml_new_doc("1.0");
//$node = $doc->create_element("markers");
//$parnode = $doc->append_child($node);

$doc = new DOMDocument('1.0');
$node = $doc->createElement('markers');
$parnode = $doc->appendChild($node);


// Opens a connection to a mySQL server
$connection=mysql_connect (localhost, $username, $password);
if (!$connection)
		{
		die('Not connected : ' . mysql_error());
		}

// Set the active mySQL database
$db_selected = mysql_select_db($database, $connection);
if (!$db_selected)
		{
		die ('Can\'t use db : ' . mysql_error());
		}


// Select all the rows in the markers table

$query = "SELECT * FROM tower WHERE 1";
$result = mysql_query($query);
if (!$result) {
  die('Invalid query: ' . mysql_error());
}

header("Content-type: text/xml");

// Iterate through the rows, adding XML nodes for each

while ($row = @mysql_fetch_assoc($result)){
  // ADD TO XML DOCUMENT NODE
  $node = $doc->createElement("marker");

  $newnode = $parnode->appendChild($node);

  $newnode->setAttribute ("id", $row['towerId']);
  $newnode->setAttribute ("address", $row['64address']);
  $newnode->setAttribute ("lat", $row['lat']);
  $newnode->setAttribute ("lng", $row['lng']);
  $newnode->setAttribute ("tipo", $row['tipo']);
}

echo $doc->saveXML();

?>
```

tabela.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
<title>Project - WSN monitoring - by Mathematics and Engineering Department, UMa</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<META name="description" content="Publications from members of the Department of Mathematics and Engineering of University of Madeira.">
<META name="keywords" content="Publications,DME,UMa,University,Madeira">
<meta http-equiv="refresh" content="30">

<style>
body{
 font-family:verdana;
 color:#555555;
 margin:0px;
 padding:0px;
}

</style>

</head>
<body>

<?php
$idTower = $_GET['idtower'];
$ipservidor  = "localhost";

$mes = date("m");
if(strlen($mes) == 1){
        $mes = "0".$mes;
}

require('mysql4.php');
$dbObject = @new sql_db($ipservidor, 'root', '', 'flosense', false);

$motesIdList = array($idTower);

$content .= "<table width='100%' cellpadding='0' cellspacing='1' style='font-size:9px;font-family:Verdana;margin-left: auto; margin-right: auto;'><tr>";
$displayMoteChannelInfo = "";
foreach($motesIdList as $moteId)
    {

        $moteIdaux = $moteId;

$rssi = "?";
$aQueryResult = $dbObject->sql_query("SELECT * FROM rssi WHERE towerId = '".$moteId."' ORDER BY dateTime DESC LIMIT 1;");
if($row = mysql_fetch_array($aQueryResult, MYSQL_ASSOC)){
 $rssi = $row['rssi'];
}



 $aQueryResult = $dbObject->sql_query("SELECT * FROM ambientd0 WHERE towerId = '".$moteId."' ORDER BY dateTime DESC LIMIT 1;");
                        if($row = mysql_fetch_array($aQueryResult, MYSQL_ASSOC))
                          {
                          if($row['towerId'] == 2)
                            {
        $valor = round((-39.6+0.01*$row['temperature']-25.0)*(0.01+0.00008*$row['humidity'])-4.0+0.0405*$row['humidity']-0.0000028*$row['humidity']*$row['humidity'],2);
                                $legend = "Humidity";
                                $valor = "<b>".$valor."</b><td align='center'><b>%</b></td>";
                                $displayMoteChannelInfo = "<tr><td>".$legend."</td><td style='text-align:right;background-color:#ffffff;'nowrap>".$valor."</td></tr>";

                                $valor = round((-39.6+0.01*$row['temperature']),2);
                                $legend = "Temperature";
                                $valor = "<b>".$valor."</b><td align='center'><b>&#176;C</b></td>";
```

```
                                $displayMoteChannelInfo .= "<tr><td>".$legend."</td><td style='text-align:right;background-
color:#ffffff;'nowrap>".$valor."</td></tr>";

                                $valor = (1.081*$row['light']/1024);
                                $valor = round((($valor/5560)*100/0.16e-6),0);
        $valor = "<b>".$valor."</b><td align='center'><b>Lux</b></td>";
                                $legend = "Light";
                                $displayMoteChannelInfo .= "<tr><td>".$legend."</td><td style='text-align:right;background-color:#ffffff;'
nowrap>".$valor."</td></tr>";

        $valor = round(((1.1*13.45*$row['battery']/1024)),2);
        $valor = "<b>".$valor."</b><td align='center'><b>V</b></td>";
        $legend = "T.bateria";
        $displayMoteChannelInfo .= "<tr><td>".$legend."</td><td style='text-align:right;background-color:#ffffff;'
nowrap>".$valor."</td></tr>";


                                $valor = round(((5*1.1*$row['tens_psolar']/1024)),3);
                                $valor = round((($valor - 2.5)*10-.625),2);

                                if ($valor < 0)
                                  {
                                    $valor = 0;
                                    $valor = "<b>".$valor."</b><td align='center'><b>A</b></td>";
                                    $legend = "Corr Psolar";
                                    $displayMoteChannelInfo .= "<tr><td>".$legend."</td><td style='text-align:right;background-
color:#ffffff;' nowrap>".$valor."</td></tr>";

                                  }
                                else
                                  {
                                    $valor = "<b>".$valor."</b><td align='center'><b>A</b></td>";
                                    $legend = "Corr Psolar";
                                    $displayMoteChannelInfo .= "<tr><td>".$legend."</td><td style='text-align:right;background-
color:#ffffff;' nowrap>".$valor."</td></tr>";
                                  }

                                $valor = $corrAerogerador = round(((5*1.1*$row['tens_aeorogerador']/1024)),3);
                                $valor = round((($valor - 2.5)*50-.625),2);
        $valor = "<b>".$valor."</b><td align='center'><b>A</b></td>";
                                $legend = "Corr Aero";
                                $displayMoteChannelInfo .= "<tr><td>".$legend."</td><td style='text-align:right;background-color:#ffffff;'
nowrap>".$valor. "</td></tr>";

                                $displayMoteChannelInfo .= "<tr><td colspan='3' align='center' style='font-
size:8px;'>".$row['dateTime']."</td></tr>";
                                }
         else
         {
                                    $valor = round((-39.6+0.01*$row['temperature']-25.0)*(0.01+0.00008*$row['humidity'])-
4.0+0.0405*$row['humidity']-0.0000028*$row['humidity']*$row['humidity'],2);
                                $legend = "<img src='icons/humidity.jpg' width='16' height='16' border='0'  title='Humidity'  alt='Humidity'
/>"; //"Humidity";
                                $valor = "<b>".$valor."</b><td align='center'><b>%</b></td>";
                                $displayMoteChannelInfo = "<tr><td><table><tr><td>".$legend."</td><td style='text-
align:right;background-color:#ffffff;'nowrap>".$valor."</td></tr>";

                                $valor = round((-39.6+0.01*$row['temperature']),2);
                                $legend = "<img src='icons/temperature.png' width='16' height='16' border='0' title='Temperature'
alt='Temperature' />";//"Temperature";
                                $valor = "<b>".$valor."</b><td align='center'><b>&#176;C</b></td>";
                                $displayMoteChannelInfo .= "<tr><td>".$legend."</td><td style='text-align:right;background-
color:#ffffff;'nowrap>".$valor."</td></tr>";

                                $valor = (1.081*$row['light']/1024);
                                $valor = round((($valor/5560)*100/0.16e-6),0);
                                $valor = "<b>".$valor."</b><td align='center'><b>Lux</b></td>";
                                $legend = "<img src='icons/sun.jpg' width='16' height='16' border='0' title='Light intensity' alt='Light
intensity' />";//"Light";
                                $displayMoteChannelInfo .= "<tr><td>".$legend."</td><td style='text-align:right;background-color:#ffffff;'
nowrap>".$valor."</td></tr>";


                                $valor = round(((1.081*3*$row['battery']/1024)),2);
                                if($valor >=2.65)
                                {
```

```
                                        $legend = "<img src='icons/full_charge.jpg' width='16' height='16' border='0' title='Voltage' alt='Voltage'
/>";
                                    }
                                        elseif($valor <2.65 && $valor >= 2.35)
                                        {
                                        $legend = "<img src='icons/half_charge.jpg' width='16' height='16' border='0' title='Voltage' alt='Voltage'
/>";
                                        }
                                    else
                                    {
                                        $legend = "<img src='icons/no_charge.jpg' width='16' height='16' border='0' title='Voltage' alt='Voltage'
/>";
                                    }
                $valor = "<b>".$valor."</b><td align='center'><b>V</b></td>";



        $displayMoteChannelInfo .= "</table></td><td valign='top'><table><tr><td>".$legend."</td><td style='text-align:right;background-
color:#ffffff;' nowrap>".$valor."</td></tr>";
        $displayMoteChannelInfo .= "<tr><td><img src='icons/rssi.png' width='16' height='16' border='0' title='RSSI' alt='RSSI' /></td><td
style='text-align:right;background-color:#ffffff;' nowrap><b>".$rssi."</b></td><td align='center'><b>dBm</b></td></tr>";
        $displayMoteChannelInfo .= "</table></td></tr><tr><td colspan='3' align='center' style='font-
size:8px;'>".$row['dateTime']."</td></tr>";
        }


                                    $content .= "<td valign='top' style='text-align:center;'><table width='115' style='margin-left: auto; margin-
right: auto;border-bottom:1px solid #333333;
                                    border-right:1px solid #333333;background-color:#f0f0f0;-moz-border-radius: 4px;-webkit-border-radius:
4px;'>
                                    <tr><TD height=20 style='background-color:#e0e0e0;-moz-border-radius: 4px;-webkit-border-radius: 4px;'
colspan='3' align='center'>
                                    Sensor node ID: <b>".$moteId."</b></TD></TR>".$displayMoteChannelInfo."</table></td>";

    }

                                    //$displayMoteChannelInfo .= "<tr><td>".$legend."</td><td style='text-align:right;background-
color:#ffffff;' nowrap>".$valor."</td></tr>";
    }
    if(!isset($row['towerId'])){
            if($moteId == 0){
              $content .= "<td style='text-align:center'>This is the coordinator!</td>";
            }else{
              $content .= "<td style='text-align:center'>Data not available!</td>";
            }

    }
echo $content."</tr></table>";


$dbObject->sql_close();


?>
</body>
</html>
```

graph2.php

```
<?php
$ipServidor = "localhost";
$bdDestino = "flosense";

require('mysql4.php');
$dbObject = @new sql_db($ipServidor, 'root', '', $bdDestino, false);

$motesIdList = array();
```

```
$data = date("d-m-Y");
if(isset($_POST['data']))
{
		$data = $_POST['data'];
}

$enddata = date("d-m-Y");
if(isset($_POST['enddata']))
{
		$enddata = $_POST['enddata'];
}

if(isset($_POST['motelist'])){
		$motesIdList = $_POST['motelist'];
}
$units = array(0=>array('%','Humidity'),
			1=>array('ºC','Temperature'),
			2=>array('Lux','Light'),
			3=>array('V','Tens&atilde;o da bateria'),
			4=>array('A','Corrente PainelSolar'),
			5=>array('A','Corrente Aerogerador'),
			6=>array('W','Pot&ecirc;ncia'),
			7=>array('dBm','RSSI'));
$chf = 0;
$unit = "";
if(isset($_POST['channel']))
{
		$chf = $_POST['channel'];

}
$unit = $units[$chf];
list($d,$m,$Y) = explode("-",$data);

list($de,$me,$Ye) = explode("-",$enddata);

$monthList = array(array('m'=>$m,'Y'=>$Y));
if($me > $m || $Ye>$Y)
		{
		$maux = $m;
		$Yaux = $Y;
		while($Yaux.$maux!=$Ye.$me)
			{
				$maux++;
				if($maux == 13)
				{
				$maux = 1;
				$Yaux++;
				}
				if(strlen($maux)<2)$maux="0".$maux;
				$monthList[] = array('m'=>$maux,'Y'=>$Yaux);
			}
		}
elseif(($m > $me) && $Y >=$Ye)
		{
		echo "End date is smaller than start date!";
		exit;
		}


$diai = $d;
$diaf = $de;

//echo $nomeTabelaDB;exit;
$anoa = $Y;
$mesa = $m;
$horaa = date('H');
$mina = date('i');
$nomeTabelaDB = "ambientd0";

if(count($motesIdList)==0){
		/*$aQueryResult = $dbObject->sql_query('SELECT distinct(towerId) FROM '.$nomeTabelaDB.' ORDER BY towerId ASC
LIMIT 2;');
		while ($row = @mysql_fetch_array($aQueryResult, MYSQL_ASSOC)){
			$motesIdList[] = $row['towerId'];
		}*/
```

```
		$motesIdList[] = 15; // mote por defeito
}

$motesColor = array();
$aQueryResult = $dbObject->sql_query('SELECT towerId,cor FROM  tower ORDER BY towerId ASC;');
while($dbObject->sql_numrows()>0 && $row = mysql_fetch_array($aQueryResult, MYSQL_ASSOC)){
		$motesColor[$row['towerId']] = $row['cor'];
}


// SQL busca as últimas horas de resumo
$totalquinze = $totalHoras*1;
$tquinzemin = $totalquinze*15;

// Calculo da hora inicial
$horai = date("Y-m-d H:i:s.001",mktime(0,0,1,$mesa,$diai,$anoa));
// $horai = date("Y-m-d H:i:s.001",mktime(date("H")-$totalHoras,date("i"),date("s"),date("m"),date("d"),date("Y")));
$horaiquinze = $horai;
$iH = 0;

$horafquinze = date("Y-m-d H:i:s.001",mktime(23,59,59,$me,$de,$Ye));
// PARA CA NO BUSCA os seus dados
$dataMatrix = array();
$datax = array();
foreach($motesIdList as $moteId){

$nomeTabelaDB = "ambientd0";
switch($chf ){
		case '7':
			$nomeTabelaDB = "rssi";
			break;
		default:
			$nomeTabelaDB = "ambientd0";
			break;
}
//echo $nomeTabelaDB;exit;
		//echo "S: ".microtime()."<br />";
		$sql = "SELECT UNIX_TIMESTAMP(t.dateTime) as ts, t.*
							FROM ".$nomeTabelaDB." t
							WHERE t.towerId = '".$moteId."'
							AND t.dateTime BETWEEN  '".$horaiquinze."'
							AND '".$horafquinze."'
							ORDER BY t.dateTime ASC";

		//echo $sql;
		$aQueryResult = $dbObject->sql_query($sql);
		//echo "Q: ".microtime()."<br />";

$valor = 0;
$row = array();

		while ($row = @mysql_fetch_array($aQueryResult, MYSQL_ASSOC))
		{

				$datax[$moteId][] = $row['ts']; //mktime($H,$min,$s,$m,$d,$y); // timeStamp


			switch ($chf)
			{
			case 0:
				if($row['humidity'] > 0)
				{
					$humidity = round((-39.6+0.01*$row['temperature']-25.0)*(0.01+0.00008*$row['humidity'])-
4.0+0.0405*$row['humidity']-0.0000028*$row['humidity']*$row['humidity'],2);
					$valor = $humidity;
				}
				else
				{
					$valor = NULL;
				}
				break;

			case 1:
				if($row['temperature'] > 0)
				{
					$temperature = round((-39.6+0.01*$row['temperature']),2);
```

```
                $valor = $temperature;
        }
        else
        {
                $valor = NULL;
        }
        break;
case 2:
        if($row['light'] > 0)
        {
                $valor = (1.1*$row['light']/1024);
                $valor = round((($valor/5560)*100/0.16e-6),0);
        }
        else
        {
                $valor = NULL;
        }
        break;

case 3:
        if($row['battery'] > 0)
        {
                if($moteId ==2)
                {
                        $voltage = round($row['battery']*(1.1*13.5)/1024,3);
                        $valor = $voltage;
                }
                else
                {
                        $voltage = round($row['battery']*(1.081*3.0)/1024,3);
                }
                        $valor = $voltage;
        }
        else
        {
                $valor = NULL;
        }
        break;

case 4:

        $tens_psolar = round((((5*1.1*$row['tens_psolar']/1024)),3);
        $tens_psolar = round((($tens_psolar - 2.5)*10-.625),2);
        $valor=$tens_psolar;

        if($valor<0)
        {
                $valor = NULL;
        }
        else
        {
        $valor=$tens_psolar;
        }

        break;
case 5:
        if($row['tens_aeorogerador'] > 0)
        {

                $tens_aeorogerador = round((((5*1.1*$row['tens_aeorogerador']/1024)),3);
                $tens_aeorogerador = round((($tens_aeorogerador - 2.5)*50-.625),3);
                $valor=$tens_aeorogerador;
        }
        else
        {
                $valor = NULL;
        }
        break;
case 6:


        $tens_psolar = round((((5*1.1*$row['tens_psolar']/1024)),3);
        $tens_psolar = round((($tens_psolar - 2.5)*10-.625),2);
        $valor1=$tens_psolar;

        $voltage = round($row['battery']*(1.1*13.5)/1024,2);
```

```
                                        $valor2 = $voltage;

                                        $valor=$valor1*$valor2;

                                        break;

                                case 7:

                                        $valor = $row['rssi'];
                                        break;
                                }


                                $valoresSensor[] = $valor;
                        }


                        $dataMatrix[$moteId] = $valoresSensor;
                        $valoresSensor = array();
}

// Criação da várias linhas de dados

$linhas = array();
foreach($motesIdList as $moteId){
$linhas[$moteId] = "[";
$i = 0;
if($dataMatrix[$moteId] != null){
foreach ($dataMatrix[$moteId] as $val){

                        if(strlen($linhas[$moteId]) > 2){
                                $linhas[$moteId].=",";
                        }
                        $linhas[$moteId].="[".$datax[$moteId][$i]."000,".$val."]";
                        $i++;
}
}
$linhas[$moteId].="]";
}

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>
 <head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <title>WSN graph</title>
   <!--[if IE]><script language="javascript" type="text/javascript" src="flot/excanvas.js"></script><![endif]-->
<script language="javascript" type="text/javascript" src="flot/jquery.js"></script>
<script language="javascript" type="text/javascript" src="flot/jquery.flot.js"></script>
<script language="javascript" type="text/javascript" src="flot/jquery.flot.selection.js"></script>
 </head>
   <body style="font-family:verdana;font-size:small;">
   <table cellpadding=0 cellspacing=5 border=0 style="background-color:#f0f0f0;;width:340px;font-
family:arial;color:#555555;border:1px solid #444444;-moz-border-radius: 2px;-webkit-border-radius: 2px;">
                <tr>
                        <td style="width:20px;text-align:left;padding-left:3px;">
                                <?php echo $unit[0]; ?>
                        </td>
                        <td style="text-align:center;">
                                <?php echo $unit[1]; ?>
                        </td>
                        <td style="width:20px;">
                                 
                        </td>
                </tr>
                <tr>
                        <td colspan="3">
                                <div id="placeholder" style="width:320px;height:180px;"></div>
                        </td>
                </tr>
                <tr>
                        <td>
                                 
                        </td>
```

```
                        <td style="text-align:center;">
                                Time
                        </td>
                        <td>
                                 
                        </td>
                </tr>
                </table>
                <?php
                if(isset($_GET['leg'])){
                        ?>
                <br />
                    <font style="font-size:10px;">LEGENDA:</font><br />
                <table cellpadding=0 cellspacing=5 border=0 style="background-color:#e0e0e0;width:340px;font-family:arial;font-
size:11px;color:#555555;border:1px solid #cccccc;-moz-border-radius: 2px;-webkit-border-radius: 2px;">
                        <tr>
                                <td style="text-align:center;">

                                        <?php
                                        $motesIdList2 = array();
                                                $aQueryResult = $dbObject->sql_query('SELECT distinct(a.towerId),t.cor FROM
ambientd0 a, tower t WHERE t.towerId = a.towerId ORDER BY towerId ASC;');
                                                while($dbObject->sql_numrows()>0 && $row = mysql_fetch_array($aQueryResult,
MYSQL_ASSOC)){

                                                        if($row['towerId'] != 14 && $row['towerId'] != 102 && $row['towerId'] !=
103 && $row['towerId'] != 104){
                                                                $motesIdList2[] = $row['towerId'];

                                                                $motesColor[$row['towerId']] = $row['cor'];
                                                        }
                                                }

                                                foreach($motesIdList2 as $towerId){
                                                    echo "<b><font color='".$motesColor[$towerId]."'> ---
".$towerId."</font></b> ";
                                                }


                                        ?>


                                </td>
                        </tr>
                </table>
                <?php
                }
                ?>

   <script id="source" language="javascript" type="text/javascript">
$(function () {


<?php
foreach($motesIdList as $moteId){
        echo "var line".$moteId." = ".$linhas[$moteId].";\n";
}
?>



   $.plot($("#placeholder"),
        [
                <?php
                $i = 0;
                        foreach($motesIdList as $moteId){
                                if($i > 0){
                                        echo ",\n";
                                }
                                echo '{ data: line'.$moteId.', color: "'.$motesColor[$moteId].'"    }';
                                $i ++;
                        }
                ?>

        ],
```

```
        {
          xaxis: { mode: 'time' },
          series: {
                    // points: { show: true,radius: .1,fill: false },
              lines: { show: true,fill: false },
                     shadowSize: 0
                   },

                   grid: { hoverable: true, clickable: true, backgroundColor: { colors: ["#fff", "#eee"]} },

         // yaxis: { min: 1 },
          legend: { position: 'sw' } });

    function showTooltip(x, y, contents) {
       $('<div id="tooltip">' + contents + '</div>').css( {
          position: 'absolute',
          display: 'none',
          top: y + 5,
          left: x + 5,
          border: '1px solid #fdd',
          padding: '2px',
          'background-color': '#fee',
          opacity: 0.80
       }).appendTo("body").fadeIn(200);
    }
    var previousPoint = null;
    $("#placeholder").bind("plothover", function (event, pos, item) {
       $("#x").text(pos.x.toFixed(2));
       $("#y").text(pos.y.toFixed(2));
    if (item) {
            if (previousPoint != item.datapoint) {
               previousPoint = item.datapoint;

               $("#tooltip").remove();
               var x = item.datapoint[0].toFixed(2),
                   y = item.datapoint[1].toFixed(2);

               showTooltip(item.pageX, item.pageY, "" + formDate(Math.round(x)) + " => " + y);
            }
         }
         else {
            $("#tooltip").remove();
            previousPoint = null;
         }
     });
});
function formDate(ts){
              var currentTime = new Date(ts)
              var hours = currentTime.getHours()
              var minutes = currentTime.getMinutes()
              var day = currentTime.getDate()
              var month = currentTime.getMonth()+1
              var year = currentTime.getFullYear()
              if (minutes < 10){
              minutes = "0" + minutes
              }
              return year+"/"+month+"/"+day+" "+hours + ":" + minutes + " "

}

</script>
 </body>
          </html>
```