UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

Monitorização de condições ambientais em Explorações Agrícolas

Daniela Andrade Barbosa

DISSERTAÇÃO

MESTRADO EM INFORMÁTICA

2012

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

Monitorização de condições ambientais em Explorações Agrícolas

Daniela Andrade Barbosa

DISSERTAÇÃO

Trabalho orientado pelo Prof. Doutor Francisco Cipriano da Cunha Martins

MESTRADO EM INFORMÁTICA

2012

Agradecimentos

Agradeço em primeiro lugar ao meu orientador, o Professor Doutor Francisco Cipriano da Cunha Martins, pelo apoio e disponibilidade na realização deste trabalho, conselhos e sugestões, além das palavras de ânimo que incutia, sempre que achava necessário.

Artemisa, Badjinca, Bruno, Camila e Marta, meus colegas do mestrado, que sempre me incentivaram e apoiaram, mesmo quando estava mais desanimada.

Sou muito grata a todos os meus familiares pelo incentivo recebido ao longo destes anos, em especial ao meu marido Carlos, a paciência e compreensão pelo tempo que não pude estar com ele.

O meu profundo e sentido agradecimento a todas as pessoas que contribuíram para a concretização deste projeto, estimulando-me intelectual e emocionalmente.

Deixo também uma palavra de agradecimento ao LaSIGE por permitir a utilização da sala e pelos materiais disponibilizados.

Ao meu marido, com todo o meu carinho.

Resumo

Ao longo dos últimos anos, as redes de sensores sem fios têm despertado um interesse crescente por parte da comunidade científica, devido à sua grande aplicabilidade nas mais diversas áreas, como por exemplo, monitorização do ambiente, saúde, agricultura, etc. Dentro do contexto da agricultura, as redes de sensores podem vir a desempenhar um papel de destaque, permitindo que diversos dispositivos possam recolher e processar informações de várias fontes e, ao mesmo tempo, contribuir para o controlo de processos físicos que automatizem tarefas até agora desempenhadas exclusivamente por humanos.

Este projeto tem por objetivo programar sensores para recolha de informação em explorações agrícolas, envia-las a um *middleware* que as armazena e publica através de serviços na *Web* a aplicações de alto nível. Foi desenvolvida uma aplicação *Web* que permite aceder a esta informação de acordo com parâmetros de consultas estabelecidas pelo utilizador. A aplicação permite ainda adicionar terrenos, sensores e dispositivos, podendo ser verificado o número de dispositivos num determinado terreno, quais as suas posições no mapa e ainda qual o seu tipo. Os resultados da consulta efetuada pelos utilizadores e apresentado de duas formas: tabelas ou gráficos de acordo com o desejo do utilizador.

Palavras-chave: Agricultura, Monitorização Ambiental, Redes de Sensores, Sistemas Distribuídos.

Abstract

Over the past few years, wireless sensor networks have attracted a growing interest by the scientific community due to its wide applicability in several areas, such as environmental monitoring, health, or agriculture. Within the context of agriculture, sensor networks can play an important role, allowing multiple devices to collect and process information from multiple sources, and, at the same time, contributing to the control of physical processes and automate tasks so far exclusively performed by humans.

This project aims at programming sensors to collect enviromental information on farms, send it to a middleware that stores and publish them through web services, making them available to high-level applications. We developed a web application that allows to access this information according to user queries. The application also allows to add terrains, sensors, and devices, and makes available the number of devices in a particular field, their positions on the map, and its type. Query results are presented either as tables or as graphs according to the user request.

Keywords: Agriculture, Monitoring Environmental conditions, Sensor Networks, Distributed Systems

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviatura

RSSFs	Rede de Sensores Sem Fios
RTC	Relógio de tempo real
USB	*Universal Serial Bus*
SOS	*Sensor Observation Service*
SWE	*Sensor Web Enablement*
ISM	*Industrial, Scientific, and Medical*
FFD	*Full Function Device*
RFD	*Reduced Function Device*
PHP	*Hypertext Processor*
HTML	*HyperText Markup Language*

Capítulo 1

Introdução

As redes de sensores sem fios (RSSFs) são compostas por um conjunto de nós sensores que podem ser colocadas numa área geográfica para recolha de informações sobre as condições ambientais como, por exemplo, temperatura, humidade ou pressão atmosférica [15].

As RSSFs têm-se afirmado com um tópico de grande interesse na investigação, que é visível nos seus avanços tecnológicos e nas áreas de aplicabilidade. Apontadas como uma das principais tecnologias do século XXI, as RSSFs estão a emergir em áreas de investigação tão distintas como a saúde, a agricultura, a prevenção de fogos, a segurança ou as áreas militares. Este tipo de redes permitem monitorizar determinados fenómenos e atuar sobre muitos dispositivos [27]. Por exemplo, podemos detetar o valor da luminosidade e atuar sobre a instalação elétrica para adequar o nível de luminosidade.

As RSSFs são um tipo especial de redes *ad hoc* potencialmente compostas por centenas de elementos de rede chamados nós sensores ou simplesmente nós. Normalmente, um nó sensor é um dispositivo de tamanho reduzido que possui três componentes básicos [14]:

- Um subsistema de deteção para a aquisição de dados a partir de um ambiente monitorizado;
- Um subsistema de processamento e armazenamento local de dados;
- Um subsistema de comunicação sem fios para a transmissão e receção dos dados.

Esta dissertação apresenta o desenvolvimento de um projeto relativo a uma aplicação que utiliza redes de sensores para monitorizar as condições ambientais em explorações agrícolas. O projeto foi realizado no Laboratório de Sistemas Informáticos de Grande Escala (LaSIGE), do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa.

Neste capítulo são apresentadas as motivações, os objetivos, as contribuições, bem como a estrutura do presente documento.

1.1 Motivação

O clima dos Açores é normalmente variável. É ameno todo o ano porque beneficia da corrente quente do Golfo do México, com dias de chuva em qualquer altura do ano. Assim, a humidade relativa é geralmente elevada (> 80%), o que conjugado com uma temperatura acima dos 25ºC pode originar uma situação climatérica propícia ao aparecimento de um fungo das pastagens. Este tipo de fungo pode provocar algumas doenças no gado como, por exemplo, a hipersensibilidade à luz. A utilização de redes de sensores sem fios para recolha dos valores de temperatura, humidade do ar e humidade das folhas e ulterior disponibilização para análise por peritos, constitui a principal motivação do nosso trabalho.

1.2 Objetivos

O projeto tem como objetivos:

Implementar um sistema que monitorize e recolha informação para ajuda na prevenção da referida patologia;

Programar os sensores de modo a capturar informações de humidade do ar, de temperatura, e de humidade das folhas de acordo com um intervalo de tempo especificado por um especialista.

Programar os sensores de modo a minimizar o consumo da bateria dos sensores, porque não é viável trocar a bateria neste tipo de redes de sensores, que podem ser instalados em lugares de difícil acesso.

Disponibilizar as informações recolhidas pelos sensores na Internet de forma a ser acedida por outras aplicações que utilizam um formato *standard*, por exemplo, *Sensor Observation Service* (SOS) [6].

1.3 Contribuições deste trabalho

O contributo do trabalho é a disponibilização da informação recolhida pelos sensores de forma adequada a ser usada por especialistas para determinar as condições reais dos terrenos para tentarem estabelecer as condições do aparecimento da patologia.

Uma contribuição indireta consiste na colocação das redes de sensores em vários pontos para que se possa ter uma maior visão global ao longo do tempo e determinar os locais de foco da doença.

1.4 Estrutura do documento

Este documento está organizado da seguinte forma: além desta introdução segue-se o Capítulo 2 onde são apresentadas as características, arquitetura e as áreas de aplicações das redes de sensores sem fios. No Capítulo 3 descrevemos os sensores agrícolas e como é gerido o consumo de bateria, a comunicação via *XBee* – ZigBee entre os dispositivos e o computador. No Capítulo 4 é apresentado o trabalho efetuado, destacando-se o desenho de uma base de dados para armazenar os dados recolhidos pelos sensores, tendo sido feita também uma aplicação *Web* para apresentação das leituras dos sensores aos utilizadores, e ainda a manutenção dos dados da aplicação (adicionar e remover terrenos, dispositivos e sensores). O Capítulo 5 apresenta os resultados dos testes efetuados ao dispositivo *waspmote*. Por último Capítulo 6 apresenta as conclusões e indica as potenciais orientações futuras deste trabalho.

Capítulo 2 Redes de Sensores Sem Fios

Uma rede de sensores possui características particulares, como a utilização de recursos restritos de energia e a topologia dinâmica de rede que dificultam a reutilização de alguns algoritmos desenvolvidos para outros tipos de sistemas distribuídos. Segundo uma perspetiva lógica as RSSFs, divide-se em sensores, observadores e fenómenos. Geralmente, os sensores são compostos por detetores de *hardware*, memória, bateria, processador e transmissores. Os observadores estabelecem consultas à rede de sensores com o intuito de obter informações recolhidas pelos nós sensores relativamente a um fenómeno. Por fim, um fenómeno é definido como uma entidade de interesse para os observadores. O objetivo da rede de sensores é recolher as informações relativas aos fenómenos, processá-las e disseminar essa informação pela rede, atuando de forma colaborativa, até aos observadores. As RSSFs suportam mais de que um observador, que pode realizar consultas a mais de um fenómeno observado simultaneamente pela rede [29].

Os sensores podem ser usados para monitorizar ambientes que sejam de difícil acesso ou perigosos, tais como o fundo do oceano, proximidade de atividades vulcânicas, áreas de desastres e campos de atividade nuclear. Estes, também, podem ser usados para tarefas interativas, como encontrar sobreviventes de desastres naturais ou conter e isolar óleo derramado, para proteger a costa marítima e consequentemente o ambiente.

O baixo consumo de energia é um fator essencial na sua operação, para que os nós possam funcionar o maior tempo possível sem qualquer manutenção. O protocolo de comunicação é outra peça chave da operação das redes de sensores. Quanto mais eficiente for o protocolo, menos tempo cada sensor terá que ficar em funcionamento para transmitir os seus dados e, portanto, menor será o consumo das suas baterias.

2.1 Características das Redes de Sensores Sem Fios

As RSSFs apresentam características particulares conforme as áreas em que estão relacionadas. De seguida, apresentam-se algumas dessas características:

Restrições dos dados recolhidos

Indica se as informações recolhidas pelos sensores têm algum tipo de restrições, como um intervalo de tempo máximo para disseminação dos seus valores para um observador [23].

Endereçamento dos sensores

De acordo com a aplicação, cada sensor pode ser endereçado unicamente ou não. Por exemplo, sensores colocados no corpo humano devem indicar o local de onde a informação foi recolhida. Por outro lado, sensores que monitorizam o ambiente numa dada região com o objetivo de obterem um certo valor de um dado fenómeno, não precisam ser identificados individualmente.

Quantidade de Sensores

Característica que requer uma atenção especial, já que em muitas aplicações a quantidade de sensores requerida para a observação de um fenómeno é elevada e não deve interferir na eficiência da rede.

Mobilidade dos sensores

Indica se os sensores podem ou não mover-se relativamente ao sistema em que estão a recolher os dados. Por exemplo, sensores colocados em áreas agrícolas para a captação de informações de humidade e temperatura são tipicamente estáticos, enquanto sensores colocados na superfície de um oceano para medir o nível de poluição da água são dinâmicos.

Tarefas colaborativas

Requerem um alto grau de cooperação entre os nós da rede para a realização de uma tarefa comum. Essa característica difere da maioria das redes tradicionais em que os nós executam aplicações diferentes.

Limitação da energia disponível

Em muitas aplicações, os sensores são colocados em áreas remotas, o que não permite o fácil acesso a esses elementos para manutenção. Neste cenário, o tempo de vida de um sensor depende da quantidade de energia disponível. Aplicações, protocolos, e algoritmos para RSSFs não podem ser escolhidos considerando apenas o seu desenho e capacidade, mas principalmente a quantidade de energia consumida.

Auto-organização da rede

Numa rede de sensores sem fios podem perder-se sensores devido à sua destruição física ou falta de energia. Alguns sensores podem também ficar incomunicáveis devido a problemas no canal de comunicação sem fios ou por decisão de um algoritmo de

gestão da rede. Isso pode acontecer por diversas razões como, por exemplo, para economizar energia ou devido à presença de outro sensor na mesma região que já recolheu a informação desejada. A situação contrária também pode acontecer quando sensores inativos se tornam ativos ou novos sensores passam a fazer parte da rede. Em qualquer um dos casos, se os sensores ficarem inoperacionais ou passarem a participar da sua estrutura, é necessário haver mecanismos de auto-organização para que a rede continue a executar a sua função.

2.2 Arquitetura

A arquitetura de uma rede de sensores sem fios está dividida em infraestrutura, protocolos de rede e aplicação [29].

A infraestrutura é influenciada pelo número de sensores e pelas suas características, como a capacidade de memória, precisão na deteção do fenómeno, alcance de transmissão, ou duração da bateria. O aumento do número de sensores não é uma garantia de gerar maior precisão na recolha de informações, caminhos mais eficientes e maior disponibilidade de energia na rede. O que acontece é que há um maior número de sensores a transmitir os seus resultados num curto espaço de tempo. Se a capacidade do meio compartilhado é excedida, ocorre congestionamento na rede e o seu desempenho pode falhar, portanto a qualidade de serviço não é satisfeita.

Um protocolo de rede é responsável por dar suporte à comunicação entre os nós sensores e os observadores. Otimizar a capacidade da rede significa evitar colisão e congestionamento. Para que isso ocorra, os sensores que estiverem numa posição intermédia não devem transmitir simultaneamente. Outra boa estratégia para otimizar a capacidade da rede é diminuir a taxa de envio de informação por sensor, através da desativação de alguns deles ou juntando informações de diferentes sensores num único sensor e depois enviá-las para a aplicação.

A aplicação é onde o observador faz consultas aos fenómenos. A aplicação transforma os dados enviados pelos sensores em informações para o observador. As consultas podem ser estáticas ou dinâmicas. Segundo a ótica da arquitetura, o que uma rede de sensores sem fios faz é recolher e transmitir informação sobre um fenómeno para um observador, garantindo a qualidade do serviço necessária para a análise do fenómeno, mas a limitação de energia é um obstáculo para garantir a qualidade de serviço, pois quanto maior o número de parâmetros de qualidade de serviço, maior é o consumo de energia.

2.3 Áreas de Aplicações das RSSFs

As tarefas efetuadas em RSSFs são muito diversificadas, como no caso de aplicações ambientais, industriais, militar, entre outras. Todas estas têm um objetivo comum de recolher informação necessária para resolver um problema estabelecido.

A utilização de uma RSSFs baseia-se na sua capacidade de fornecer informações de determinadas áreas em resposta às questões colocadas pelo utilizador. Por exemplo, numa aplicação da monitorização ambiental pode-se pretender saber qual é o valor da temperatura, pressão atmosférica e humidade em diferentes locais. A abordagem de consulta é o modo mais comum em que a rede de sensores é utilizada. Num outro modo de operação, nós sensores podem permanecer em espera aguardando por qualquer ocorrência específica.

As redes de sensores sem fios são vocacionadas para utilização nas seguintes áreas:

Ambiente: Para monitorizar condições ambientais em locais internos, como habitações, locais externos como agricultura, florestas, desertos, oceanos, vulcões, etc. Por exemplo, o projeto *Zebranet* da Universidade de *Princeton* utiliza sensores com o objetivo de rastrear o movimento de zebras em África [26]. A seguir será exibida uma figura com várias redes de sensores aplicadas numa floresta. Esses sensores irão recolher informações específicas e envia-la para uma estação no local que depois envia por satélite para uma central, onde a informação irá ser processada.

Controlo: Para realizar tarefas de controlo, seja num ambiente industrial ou não. Por exemplo, sensores sem fios podem ser embutidos em "peças" numa linha de montagem para realizar testes no processo de fabrico.

Biologia: Detectar a presença de substâncias nos organismos que possam indiciar a presença de algum problema biológico.

Medicina: Os sensores podem ser utilizados em hospitais para monitorizar os movimentos dos pacientes ou controlar determinadas funções do corpo, como, os batimentos cardíacos ou a pressão arterial.

Segurança: Para aumentar a segurança em residências, estacionamentos, centro comerciais, etc.

Tráfego: Para monitorizar tráfego de veículos, malhas viárias urbanas, etc.

Figura 1 – Monitorização ambiental com RSSFs [3]

Militar: Uma das mais importantes e motivadoras aplicações é a militar. Nos Estados Unidos, o programa militar SensIT (*Sensor Information Technology*) [22], do início dos anos 2000, foi responsável por um grande avanço no uso das redes sem fios e motivou o uso dessa tecnologia noutras aplicações não militares. A eficiência no campo de batalha foi significativamente aumentada, uma vez que os sensores podem detetar inimigos ou aliados, descobrir o número de combatentes, a sua posição e rastreá-los [3]. Também é crucial usá-los para identificar armas nucleares e químicas, sem necessitar de arriscar a vida de soldados. Os dados são cifrados e submetidos a processos de assinatura digital.

Capítulo 3 *Hardware* e Programação dos Sensores

Este capítulo descreve o *hardware* utilizado no projeto, em particular os dispositivos sem fios, as placas específicas para ligação dos sensores e os nós sensores utilizados. Os dispositivos utilizados têm fortes restrições quanto à capacidade de processamento e autonomia. Vamos abordar o modo de consumo da bateria e a comunicação sem fios, que será feita utilizando o protocolo *XBee – ZigBee*. Por fim, iremos mostrar como é executada a comunicação entre os dispositivos *waspmote* e *gateway* através de comunicação sem fios.

Um dispositivo *waspmote* [35] é composto por uma placa que serve como base para a conexão dos componentes de uma rede de sensores, como por exemplo: sensores, *XBee*, placa agrícola, etc. A figura que se segue mostra-nos como é constituído um dispositivo *waspmote*.

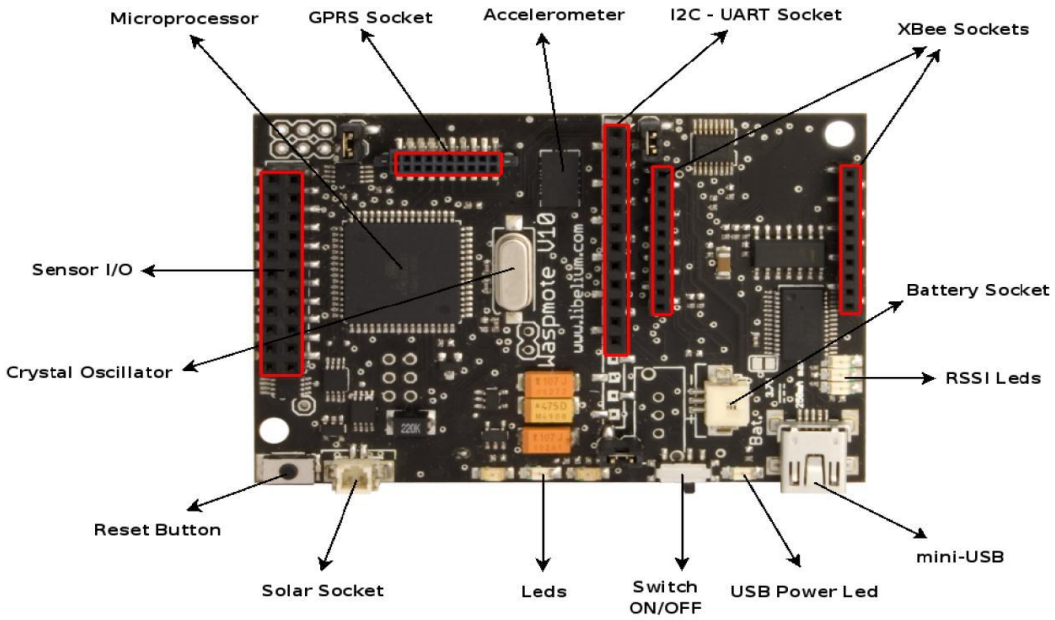


Figura 2 – *Waspmote Board* – frente [38]

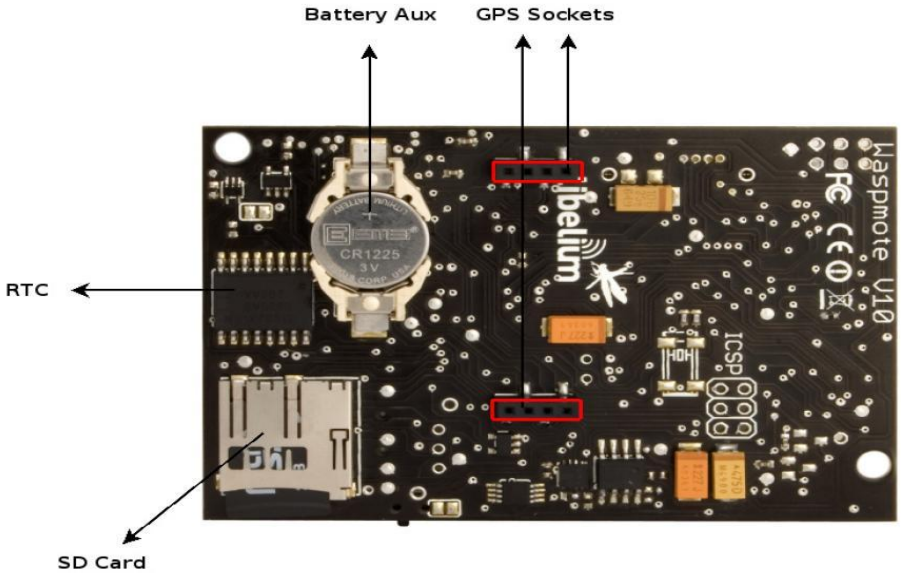


Figura 3 – Waspmote Board – traseira

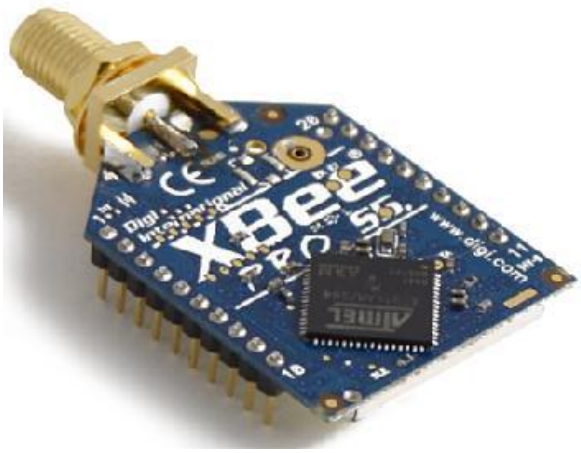


Figura 4 – *XBee*

Um *gateway* é um dispositivo que serve para estabelecer a comunicação entre as redes de sensores *waspmote* e um computador ou outro dispositivo com interface USB. É composto por uma placa, um *XBee* (figura 4) e uma antena. Para a sua utilização é necessário conectá-lo a uma porta USB. A Figura 5 mostra-nos um *gateway* pronto a utilizar.

Figura 5 – *Waspmote gateway*

3.1 Sensores

Sensores são dispositivos que recebem sinais ou estímulos físicos ou químicos, como temperatura e pressão, e convertem os dados obtidos em sinais eletrónicos. Existem vários tipos de sensores, como, por exemplo: sensores de presença, sensores de pressão, sensores agrícolas, etc.

Os sensores agrícolas são equipamentos especializados em recolher informação que pode ser utilizada na área agrícola. As informações recolhidas pelos sensores são transmitidas para uma central de processamento que poderá acionar sistemas que atuam em função dos dados recolhidos. Por exemplo, podem proceder à irrigação ou adubação automatizada em função da humidade ou da concentração de nutrientes que detetam no solo.

Os dispositivos *waspmote* podem ser equipados com uma placa agrícola, que foi desenhada propositadamente para conectar simultaneamente até 14 sensores diferentes, como por exemplo, o sensor de temperatura do solo, o sensor de humidade relativa do ar, o sensor de humidade do solo, o sensor de humidade das folhas, entre outros. Estes sensores permitem analisar parâmetros importantes na agricultura relacionados com as condições meteorológicas e do solo.

A placa de agricultura serve de interface entre os sensores e a placa *waspmote.* A figura que se segue apresenta-nos uma placa agrícola com alguns sensores conectados.

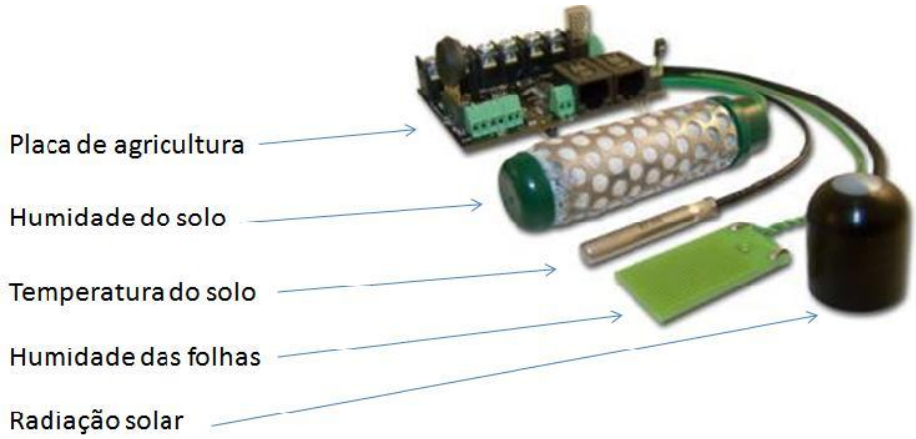


Figura 6 – Placa da agricultura

Os sensores agrícolas que vamos utilizar no projeto são:

- Sensor de temperatura;
- Sensor de humidade do ar; e
- Sensor de humidade das folhas.

Estes sensores têm características próprias que iremos detalhar nas próximas secções.

3.1.1 Sensor de Temperatura

É um sensor analógico que mede a temperatura do ambiente. As suas características são as seguintes:

Tempo de resposta: 1.65 segundos
Faixa de medição: -40ºC ~ +125ºC
Tensão de saída (0ºC): 500mV
Sensibilidade: 10mV/ºC
Consumo típico: 6µA
Consumo máximo: 12µA
Fonte de alimentação: 2.3 ~ 5.5V
Operação de temperatura: -40 ~ +125ºC
Temperatura de armazenamento: -65 ~ 150ºC
Precisão: ±2ºC entre 0ºC e 70ºC, ±4ºC entre -40 e 125ºC

Figura 7 – Sensor de temperatura

```
float value_temperature = 0;

void setup()
{
SensorAgr.setSensorMode(SENS_ON, SENS_AGR_TEMPERATURE);
USB.begin();
delay(1000);
}

void loop()
{
value_temperature = SensorAgr.readValue(SENS_AGR_TEMPERATURE);
USB.print( value_temperature);
delay(1000);
}
```

Figura 8 – Código para utilizar o sensor de temperatura

A figura 8 apresenta o código que será inserido no dispositivo *waspmote* para que possamos interagir com o sensor e obter os valores da temperatura. Este programa é composto por dois módulos: o *USB* e o *SensorAgr*, que permitem a interação com a porta USB e com a placa agrícola. O programa apresenta duas funções: *setup* e *loop.* A função *setup* é executada uma única vez. O *SensorAgr.setSensorMode* recebe dois parâmetros: o primeiro parâmetro indica se pretende ativar (*SENS_ON*) ou desativar (*SENS_OFF*) um sensor agrícola e o segundo indica qual o tipo de sensor sobre o qual se pretende atuar. Depois, fazemos *USB.begin* para ativar as portas USB. O *delay* (1000) serve para que o *hardware* tenha tempo de se iniciar antes do programa continuar.

A função *loop* é executada continuamente. Dentro desta função usamos a variável *value_temperature,* que foi inicializada anteriormente, para guardar o valor recolhido através da função *SensorAgr.readValue*. O *delay* (1000) significa que o sensor espera um segundo antes de voltar a executar a função *loop*. A tabela que se segue mostra-nos os tipos de identificadores para os sensores, que podem ser usados nas funções *setSensorMode* e *readVAlue.* O *USB.print(value_humidity_1)* vai comunicar para a rede via USB, e irá enviar o valor guardado nesta variável (*value_humidity_1*).

Temperature (sensor de temperatura)	SENS_AGR_TEMPERATURE
Humidity (sensor de humidade)	SENS_AGR_HUMIDITY
Leaf wetness (sensor de humidade das folhas)	SENS_AGR_LEAF_WETNESS

Tabela 1 – Tipos de identificadores para os sensores agrícolas

3.1.2 Sensor de Humidade

É um sensor analógico para medição da humidade do ambiente, que fornece uma tensão de saída proporcional à humidade relativa do ar na atmosfera. De seguida apresentamos as características e a imagem do referido sensor.

Faixa de medição: 0 ~ 100%RH
Sinal de saída: 0,8 ~ 3.9V (25ºC)
Consumo Típico: 0.38mA
Consumo máximo: 0.5mA
Fonte de alimentação: 5VDC ±5%
Temperatura de operação: -40 ~ +85ºC
Temperatura de armazenamento: -55 ~ +125ºC
Tempo de resposta: <15 segundos
Precisão: <±4%RH entre 30 e 80%, <±6%RH entre 0 e 100%

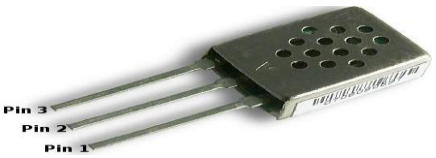


Figura 9 – Sensor de humidade do ar

```
float value_humidity_1 = 0; // Inicializar variáveis para as leituras dos sensores

void setup()
{
SensorAgr.setSensorMode(SENS_ON, SENS_AGR_HUMIDITY);//ligar o Sensor
USB.begin();
delay(1000);
}

void loop()
{
value_humidity_1 = SensorAgr.readValue(SENS_AGR_HUMIDITY);
USB.print(value_humidity_1);
delay(1000);
}
```

Figura 10 – Código para utilizar o sensor de humidade do ar

A Figura 10 ilustra um exemplo de um código que podemos inserir no dispositivo *waspmote* para programar os sensores de humidade do ar. A função *SensorAgr.setSensorMode* serve para ativar o sensor pretendido (nesse caso o *SENS_AGR_HUMIDITY*).

3.1.3 Sensor de Humidade das Folhas

O sensor de humidade das folhas necessita de estar permanentemente junto das folhas (as folhas têm que estar encostadas na placa castanha como mostra a figura 11) para podermos saber o verdadeiro valor da humidade das folhas. Este sensor comporta-se como uma resistência de valor muito alto na ausência de condensação nos favos

condutores que o compõem, e que pode cair para cerca de 5kΩ quando está completamente submerso na água [37]. As características do sensor são as seguintes:

Faixa de Resistência: 5kΩ ~ > 2MΩ
Faixa de tensão de saída: 1V ~ 3.3V
Comprimento: 3.95cm
Largura: 1.95 cm

Figura 11 – Sensor de humidade das folhas

```
float value_lw = 0;

void setup()
{
SensorAgr.setSensorMode(SENS_ON, SENS_AGR_LEAF_WETNESS);
USB.begin();
delay(1000);
}

void loop()
{
value_lw = SensorAgr.readValue(SENS_AGR_LEAF_WETNESS);
USB.print( value_lw);
delay(1000);
}
```

Figura 12 – Código para utilizar o sensor de humidade das folhas

3.2 Relógio de Tempo Real

O Relógio de tempo real (RTC) é um relógio integrado no dispositivo *waspmote*, que mantém o controlo do tempo presente. Os RTCs estão presentes na quase totalidade dos dispositivos eletrónicos que precisam manter um controlo rigoroso do tempo [32]. É através do RTC que especificamos a que instante do tempo o dispositivo *waspmote* deve executar as suas ações programadas. Permite que o *waspmote* funcione em modo de poupança de energia (*deep sleep* e *hibernate*) para "acordar" apenas nos momentos necessários. Por exemplo, pode ficar inativo durante 1 hora 20 minutos e 15 segundos, acordar e executar o seu programa. Quando o dispositivo *waspmote* está no modo poupança de energia (*deep sleep* e *hibernate*), desliga a sua bateria principal.

O RTC tem duas fontes de alimentação: a bateria principal e a bateria auxiliar. Quando o *waspmote* é ligado, o RTC é alimentado através da bateria principal. No entanto, para garantir que o tempo funcional é sempre mantido e a informação não é perdida quando a bateria principal é trocada ou não esta carregada, existe uma bateria auxiliar para alimentar o RTC.

O RTC pode ser usado para definir uma base de tempo comum em todos os *waspmote* numa rede. Permite o máximo de precisão sem desvios ao longo do tempo, o que torna possível à rede permanecer sincronizada a enviar ou receber tarefas a serem realizadas simultaneamente. A maioria dos RTCs no mercado tem uma variação de ± 20 ppm (partes por milhão), que é equivalente a uma perda de precisão 1.7 segundos por dia (10.34 minutos/ano), no entanto o modelo escolhido para o dispositivo *waspmote* tem uma perda de apenas ± 2 ppm, o que equivale a uma variação de 0.16 segundos por dia (1 minuto/ano) [36].

Interrupções são alertas recebidos pelo microcontrolador que indica que o dispositivo *waspmote* deve interromper a tarefa que está a executar, para efetuar a ação pretendida. Por exemplo, quando um sensor estiver no modo *deep sleep* e recebe uma interrupção do RTC deve sair desse modo e executar a ação programada (recolha de dados). Existem dois tipos de interrupção: síncrona (é uma interrupção programada para um determinado acontecimento), e assíncrona (é programada de modo a não se saber quando vai ocorrer).

A figura 13 mostra um programa que coloca o *waspmote* em modo *deep sleep* durante cinco minutos. Isto faz com que as interrupções e módulos fiquem desligados nesse período e que após os cinco minutos, o *waspmote* acorda e executa a ação programada. Este programa é composto por quatro módulos: *USB, RTC, Utils* e *PWR*. O módulo *RTC* permite-nos definir alarmes, que interrompem o processador em determinados instantes do tempo. O módulo *Utils* permite-nos interagir com os utilitários; neste caso está a ser usado para ligar e desligar os *leds* e o módulo *PWR* interagir com os modos de operação do *waspmote* (*deep sleep*, *hibernate* e *sleep*) e ainda permite ligar e desligar o *hardware*. É através da função *RTC.setTime* que definimos a data e hora no RTC (especificando o ano, mês, dia do mês, dia da semana, hora, minuto e segundo). Na função *PWR.deepSleep* começamos por especificar o dia do mês, hora, minuto, e depois qual o dispositivo que iremos ligar e os que vamos desligar para ajuda na poupança da bateria. O *RTC_OFFSET* significa que vamos utilizar o tempo real lido a partir do RTC [34]. Foi utilizado o *RTC_ALM1_MODE1*, porque aqui temos um leque maior de escolha, por exemplo, podemos programar o *waspmote* para recolha de informação no dia 19 de cada mês, às 12 horas, 27 minutos e 4 segundos. Depois de executar essa ação entramos no modo de poupança de energia (*deep sleep*), e por fim podemos especificar as opções do modo *deep sleep*, como, por exemplo:

- ALL_OFF: desliga todas as opções no dispositivo *waspmote*;
- SENS_OFF: desliga os interruptores relacionados com a placa de sensor;
- UART0_OFF: fecha a UART0 e desliga o interruptor relacionado com o *XBee*;
- BAT_OFF: desliga o interruptor relacionado com a bateria;
- RTC_OFF: desliga o interruptor relacionado com o *real time clock.*

O *intFlag* é uma variável global do módulo *waspmote* e é usado para identificar qual o módulo que gerou a interrupção capturada. O *RTC_INT* é uma constante do módulo *RTC* e identifica que a interrupção foi gerada pelo RTC. O *intFlag &= ~(RTC_INT)* vai apagar a interrupções *RTC_INT*. É muito importante escolher quais os dispositivos que queremos desligar quando invocamos o *PWR.deepSleep.* Por exemplo, se colocarmos o *ALL_OFF,* isso significa que estamos a desligar todos os componentes do sensor, inclusivamente o RTC, e assim este não vai gerar a interrupção para acordar o dispositivo.

```
void setup()
{
USB.begin();
RTC.setTime("09:10:21:04:14:25:00");
}

void loop()
{
RTC.ON();
PWR.deepSleep("00:00:00:10",RTC_OFFSET,RTC_ALM1_MODE1,SENS_OFF|UART0_O
FF|BAT_OFF);
if( intFlag & RTC_INT )
{
  Utils.blinkLEDs(1000);
  Utils.blinkLEDs(1000);
  Utils.blinkLEDs(1000);
  intFlag &= ~(RTC_INT);//Clear flag
}
}
```

Figura 13 – Modo *Deep Sleep*

3.3 Consumo de Energia

De uma forma geral, a operação básica das redes de sensores consiste em realizar a recolha e o envio de dados em períodos cíclicos, ficando o menor tempo possível a realizar estas operações. Entre um e outro ciclo de operações, é ideal os nós da rede passarem a um estado de espera com baixo consumo de energia (*deep sleep* e *hibernate*). Tipicamente, este estado de espera corresponde a uma elevada percentagem do tempo de operação, permitindo uma poupança de bateria.

A conservação de energia é um dos aspetos mais importantes a ser considerado quando estamos a trabalhar com redes de sensores sem fios. Os maiores consumidores de energia de uma rede de sensores são os modelos de rádio, processador e elementos que fazem a deteção do ambiente (sensores) **[23]**.

O *waspmote* tem um circuito que controla o nível da bateria. Quando um certo limite é atingido é acionado um alarme, que adverte o processador de que o *waspmote* está a ficar sem bateria. Este nível de bateria crítica, varia dependendo do tipo de módulos do *waspmote*, bem como da aplicação final. Este é um exemplo da interrupção assíncrona [36].

Os elementos do modelo de energia são:

- Bateria: É o armazenador de energia do sensor, que tem uma capacidade finita e uma taxa de consumo.
- Sensores: O consumo depende do modo de operação e do tipo de grandeza medida.
- Processador: O consumo depende da velocidade do relógio (quanto menor a frequência menor o consumo) e do modo de operação. O consumo pode ser medido pelo número de ciclos de relógio para diferentes tarefas, como o processamento de sinais, verificação de código de erro, etc. Este modelo é usado em todas as operações que fazem parte do modelo de sensor.
- Rádio: O consumo de energia depende da operação efetuada. Tipicamente a transmissão de dados consome mais energia que a sua receção. Este modelo é utilizado pela pilha de protocolos da rede.

As redes de sensores *waspmote* têm quatro tipos de modos de operação:

- ON
- *Sleep*
- *Deep Sleep*
- *Hibernate*

Cada um destes modos tem um consumo de energia próprio. A tabela abaixo mostra o consumo de energia e os intervalos de ciclos para cada modo.

	Consumo	Micro	Ciclo	Interrupções Aceites
ON	9mA	ON		Síncronas e Assíncronas
SLEEP	62µA	ON	32ms-8s	Síncronas (*watchdog*) e Assíncronas
DEEP SLEEP	62µA	ON	8s-min/H/dia	Síncronas (RTC) e Assíncronas
HIBERNATE	0.7µA	OFF	8s-min/H/dia	Síncronas (RTC)

Tabela 2 – Consumo de energia dos modos *waspmote*

O *Watchdog* permite acordar (gerando uma interrupção) o microcontrolador de um estado de baixo consumo. A interrupção gerada pelo *Watchdog* é usada para controlar o *waspmote* no modo *sleep* [36].

3.3.1 Modo *Sleep*

O modo *sleep* coloca o dispositivo a utilizar sempre a mesma bateria quando está a dormir [35]. O programa principal e o microcontrolador ficam em pausa, podendo ser acordado por todas as interrupções síncronas e assíncronas, gerada pelo *Watchdog*.

3.3.2 Modo *Deep Sleep*

O programa principal fica em modo pausa, e o microcontrolador pode ser acordado por todas as interrupções assíncronas e síncronas provocadas pelo RTC. Com o RTC podemos programar os sensores de forma a ficarem no modo *deep sleep* e acordarem, por exemplo, somente no dia 10 de cada mês às 12 horas e 20 minutos para recolha de informação.

A diferença do modo *sleep* para o *deep sleep* é que no *sleep* o dispositivo *waspmote* é acordado através da interrupção *watchdog* e o intervalo do ciclo é muito pequeno (de 32 milissegundos a 8 segundos), enquanto que no *deep sleep* o intervalo é mais longo (superior a 8 segundos) ou seja, no modo *sleep* o sensor só pode dormir até 8 segundos, enquanto que no *deep sleep* pode dormir mais de 8 segundos. Neste modo o *waspmote* é acordado através da interrupção RTC, o que faz consumir menos energia da bateria principal porque enquanto o dispositivo *waspmote* estiver a dormir, a alimentação será feita através da bateria auxiliar [35].

3.3.3 Modo *Hibernate*

O programa principal pára, o microcontrolador e todos os módulos *waspmote* são completamente desligados. A única maneira de reativar o dispositivo é através do alarme previamente programado na RTC (interrupção síncrona). O intervalo deste ciclo é para mais de 8 segundos. Quando o dispositivo está totalmente desconectado da

bateria principal, o RTC é alimentado através da bateria auxiliar com um consumo de 0.7μA.

A diferença deste modo para o *deep sleep* é no consumo de energia, e que no modo *hibernate* a memória perde a informação porque também é desligada. Ao sair deste estado o micro é reiniciado para instalação e rotinas do *loop*. Por este motivo é que usamos o modo *deep sleep* no nosso projeto.

3.4 *XBee – ZigBee*

Os sensores *waspmote* integram os módulos *Digi XBee* para a comunicação. Esses módulos comunicam com o microcontrolador usando o UART_0 com uma velocidade de 38400bps. Existem sete módulos *XBee* possíveis, distribuídos pela *Libelium* para integração em *waspmote*. Nós iremos utilizar o *XBee ZigBee* PRO porque para a experiência que estamos a fazer permite comunicar dentro do alcance desejado.

O *waspmote* funciona com diferentes protocolos (*ZigBee*, *Bluetooth*, *GPRS*) e utiliza frequência ISM (*Industrial, Scientific, and Medical*) de respetivamente, 2,4 GHz, 868MHz, e 900MHz, sendo capaz de fazer ligações de até 12 quilómetros [4]. As redes *ZigBee* oferecem uma capacidade de conectar milhares de dispositivos numa rede, com taxas de transferência de dados que varia de 20Kbps a 250Kbps.

Os módulos RF padrão *ZigBee* foram criados para economizar o máximo de energia. Assim, é possível criar dispositivos remotos alimentados com pilhas ou baterias comuns, que durarão meses ou mesmo anos sem necessitarem de substituição.

Em seguida vamos apresentar um quadro onde é mostrado o protocolo e respetiva frequência para cada modelo *XBee*.

Model	**Protocol**	**Frequency**	**TxPower**	**Sensitivity**	**Range**
XBee - 802.15.4	802.15.4	2.4GHz	1mW	-92dB	500m
XBee - 802.15.4 – Pro	802.15.4	2.4GHz	100mW	-100 dBm	7000m
XBee – ZB	ZigBee – Pro	2.4GHz	2mW	-96dBm	500m
XBee - ZB – Pro	**ZigBee - Pro**	**2.4GHz**	**50mW**	**-102dBm**	**7000m**
XBee – 868	RF	868 MHz	315mW	-112dBm	12Km
XBee – 900	RF	900 MHz	50mW	-100dBm	10Km
XBee – XSC	RF	900MHz	100mW	-106dBm	12Km

Tabela 3 – Módulo *Xbee*

Numa rede *ZigBee* são identificados dois tipos de dispositivos: *Full Function Device* e *Reduced Function Device*.

Os *Full Function Device* (FFD) são dispositivos mais complexos e necessitam de um *hardware* mais potente para a implantação da pilha de protocolos, consequentemente, consomem mais energia. Numa topologia de rede *ZigBee* estes podem assumir o papel de coordenador, encaminhador (*router)* ou mesmo de um dispositivo final (*end device*). Podem comunicar com quaisquer membros da rede. Por outro lado, os *Reduced Function Device* (RFD) são dispositivos mais simples, onde a pilha de protocolo pode ser implementada usando os mínimos recursos possíveis de *hardware*. Numa topologia de rede *ZigBee* estes assumem o papel de dispositivo.

No padrão *ZigBee* existem três classes de dispositivos lógicos (*Coordenador*, *Router* e *End Device*) que definem a rede:

ZigBee Coordenador - Só pode ser implementado através de um dispositivo FFD. O coordenador é responsável pela inicialização, distribuição de endereços, manutenção da rede, reconhecimento de todos os nós, entre outras funções, podendo servir como ponte entre várias outras redes *ZigBee*. Só pode existir um coordenador em cada rede.

ZigBee Router - Só pode ser implementado através de um dispositivo FFD. Pode servir como *router* intermédio entre sensores. Através de um *router*, uma rede *ZigBee* pode ser expandida, e assim ter mais alcance.

ZigBee End Device - Tem funcionalidades suficientes para conversar com o coordenador e o *router*, mas não pode repassar os dados de outros dispositivos. Pode ser implementado através de um dos dispositivos *full function device* ou *reduced function device*. Assim, este é o nó que consome menos energia, pois na maioria das vezes ele fica no modo *sleep*.

O *XBee ZigBee* utiliza os 16 canais, no entanto o modelo *XBee ZB PRO* está limitado a 13 canais. As topologias em que estes módulos podem ser utilizados são: estrela e árvore.

Estrela é uma das topologias de rede *ZigBee* mais simples de ser implantada. É composta por um nó *coordenador*, e vários nós finais. Todas as decisões são concentradas num único nó, simplificando bastante, nesse aspeto, a implementação de cada um dos outros nós. É da responsabilidade do *ZigBee coordenador* controlar a rede, assumindo este papel central para comunicar diretamente com todos os dispositivos finais. É portanto o coordenador que inicia e mantém os dispositivos na rede. Toda a informação em circulação na rede passa pelo nó *coordenador*. A figura que segue mostra na raiz o coordenador, e à sua volta os nós finais.

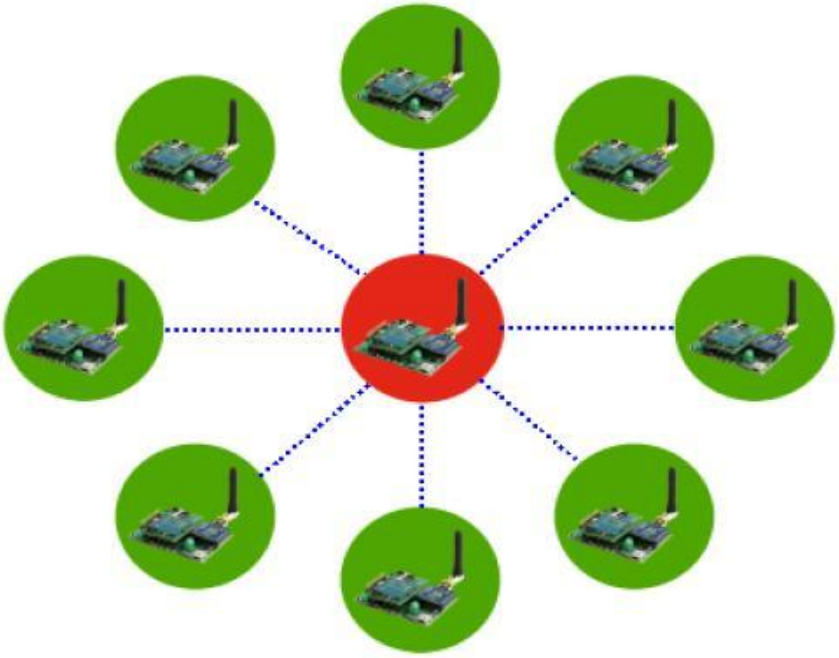


Figura 14 – Topologia de rede em estrela

A organização em árvore permite a distribuição de dados e mensagens de controlo numa estrutura hierárquica, onde o coordenador assume o papel de nó raiz da rede. A figura que se segue mostra na raiz o coordenador, seguidamente os *router* e por fim os nós finais nas folhas.

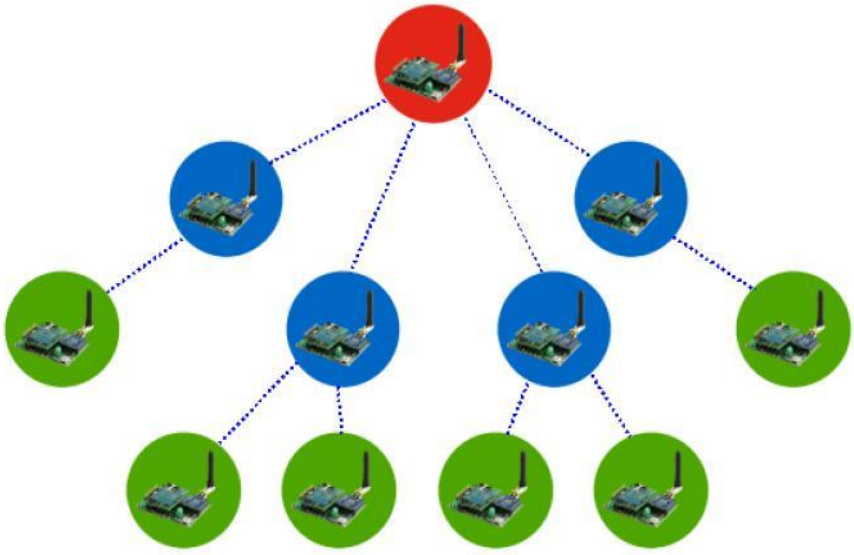


Figura 15 – Topologia de rede em árvore

Capítulo 4 Trabalho Realizado

Neste capítulo é descrito o trabalho realizado durante a execução deste projeto, designadamente o desenvolvimento da aplicação *Web*, modelo de dados, a forma de comunicação entre o *waspmote* e *gateway*, e por último, a utilização do *middleware* MuFFIN.

4.1 Desenvolvimento da aplicação

A aplicação *Web* foi desenvolvida utilizando a linguagem de programação PHP para registar informações sobre os sensores, terrenos, dispositivos, que vão ser armazenadas numa base de dados MySQL.

No ambiente PHP, o código é embebido diretamente no documento HTML, dando origem a um *script* contendo instruções específicas. O servidor *Web*, a que foi acrescentado um módulo PHP, consegue interpretar os comandos neles inseridos, e transformar o resultado em código HTML interpretável pelo navegador.

A Figura que se segue apresenta-nos a arquitetura do nosso projeto.

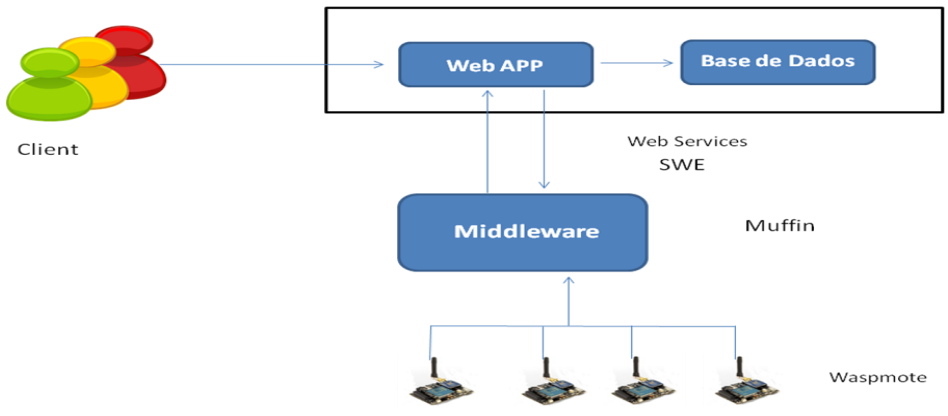


Figura 16 – Arquitetura do projeto

As redes de sensores foram programadas para recolher informações e envia-las para o *middleware* MUFFIN de cinco em cinco minutos, que por sua vez guarda esta

informação. Quando o utilizador pedir uma consulta, será enviado um pedido ao *middleware*, com as datas pretendidas das leituras dos sensores, que responde com as leituras para o utilizador. Estas são guardadas numa base de dados da aplicação *Web*. Este acesso será efetuado utilizando serviços na *Web* de acordo com a norma SWE (*Sensor Web Enablement*).

A aplicação *Web* permite adicionar terrenos, indicando o nome e as coordenadas dos mesmos, em cada um destes terrenos, podemos adicionar vários dispositivos indicando o nome e o seu tipo. Estes dispositivos podem ter vários sensores (sensores de humidade do ar, sensores de temperatura, sensores de humidade das folhas e etc.). E ainda podem ser visualizados as listas dos dispositivos instalados num determinado terreno, bem como podemos inquirir as leituras dos sensores, de acordo com parâmetros solicitados.

A figura que se segue mostra-nos as listas dos sensores adicionados. Ainda nessa página poderemos apagar e adicionar mais sensores.

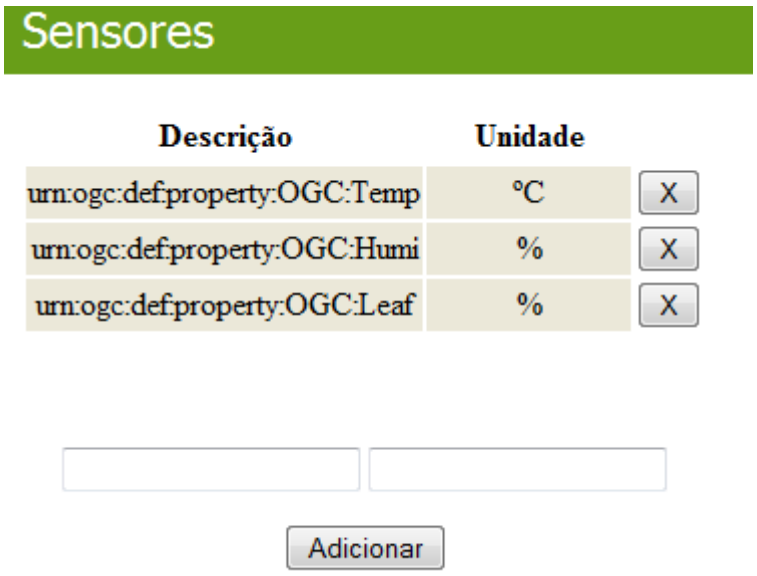


Figura 17 – Lista dos sensores

Ao adicionar sensores é necessário fornecer uma descrição que respeitar o formato "*urn:ogc:def:property:OGC:nome do sensor* ", porque o *middleware* MUFFIN utiliza os formatos SOS [6] para definir os seus serviços.

A informação de cada dispositivo pode ser consultada de duas formas: diretamente a partir do menu dispositivo ou a partir da lista dos terrenos. Por último, podemos inserir ou remover os dispositivos num determinado terreno. A figura 18 ilustra-nos um exemplo de um formulário preenchido para adicionar os dispositivos indicando em que terreno irá ser colocado e quais os sensores a utilizar.

Figura 18 – Formulário para adicionar dispositivo

Para visualização das leituras feitas pelos sensores necessitamos preencher o formulário onde especificamos o intervalo de tempo para as consultas pretendidas (data/hora de início e data/hora fim) e se pretendemos agregar ou não os sensores. Se sim, necessitamos de indicar se a agregação é por minuto, por hora, por dia, por mês, ou ano e indicar o valor pretendido, ou seja de quantas em quantas unidades de tempo se pretende agregar a informação. Além disso, devemos indicar quais as funções pretendidas (máximo, mínimo ou média); por último, qual o modo de visualização dos dados (tabela ou gráficos). A figura que se segue apresenta um formulário preenchido, onde o utilizador indica que pretende visualizar um gráfico com os valores médios das leituras de 20 em 20 minutos.

Figura 19 – Formulário para as leituras dos sensores

4.2 Modelo de Dados

Esta secção aborda a base de dados, especificando a ferramenta utilizada para a sua construção, e explicando o modelo de dados.

A base de dados foi construída utilizando MySQL para armazenamento dos dados dos sensores, dos terrenos e dos dispositivos que foram introduzidos na aplicação *Web*, servindo ainda para guardar as informações recolhidas pelos sensores vindas do *middleware* MuFFIN.

Utilizamos o servidor XAMPP que consiste num pacote que contém uma base de dados MySQL, servidor Apache, PHP, entre outras linguagens e é adaptável a qualquer sistema operativo.

A base de dados contém seis tabelas, como ilustra a figura que segue.

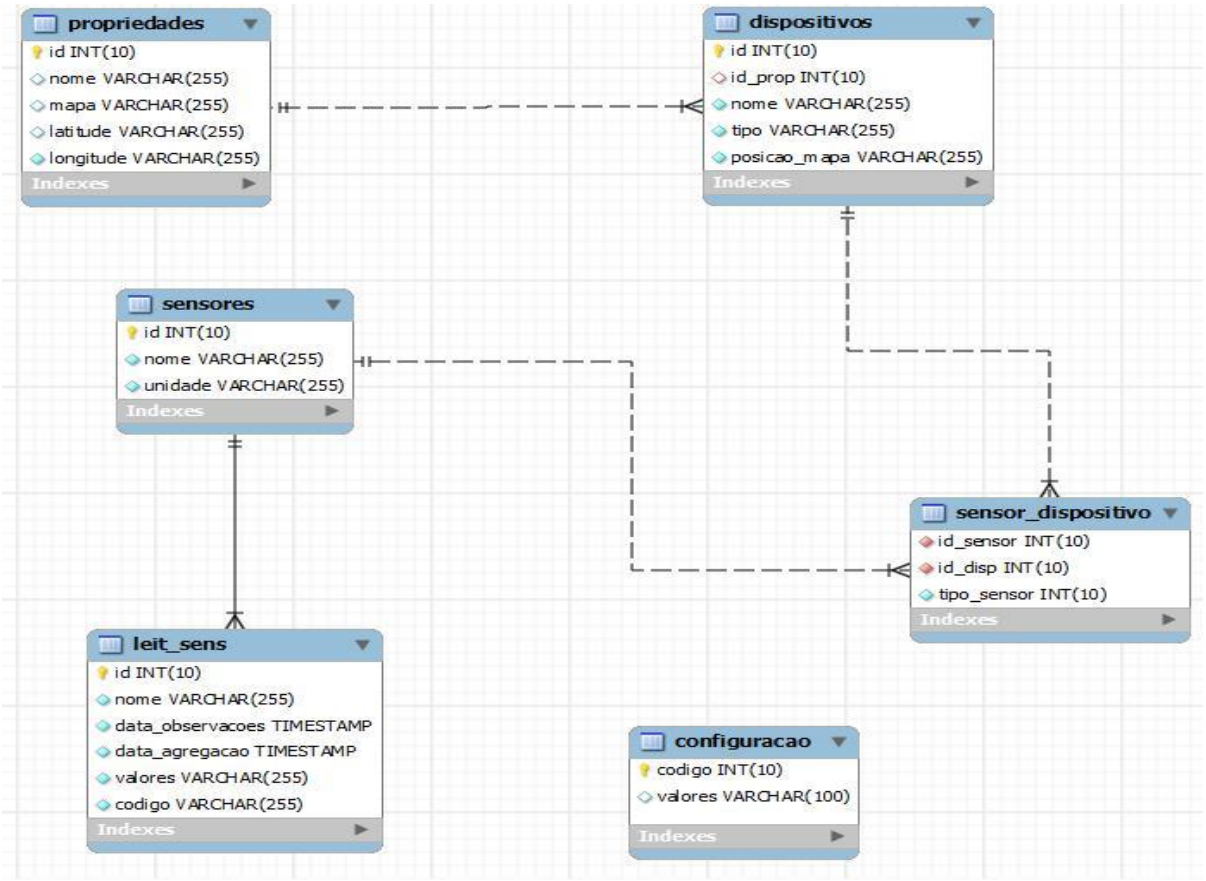


Figura 20 – Tabelas da base de dados

A tabela *propriedades* contém informação (nome do terreno, as coordenadas e um mapa) associada aos terrenos. Esta tabela relaciona-se com a tabela *dispositivos*, que armazena os vários dispositivos do terreno. A tabela *dispositivos* contém as informações sobre qual o tipo de dispositivo e a sua localização no mapa e relaciona-se com a tabela *sensores* que contém os nomes dos sensores e as suas unidades, esta tabela tem uma relação de multiplicidade de muitos para muitos. Por fim, a tabela *sensores* faz a relação com a tabela *leit_sens* que regista as leituras de um sensor.

A tabela *leit_sens* contém vários atributos, como a data da observação, data da agregação, os valores recolhidos e o seu código (unidades, por exemplo: ºC ou %). Como se pode verificar nessa tabela temos dois atributos data: *data_observações* e *data_agregação*. A *data_observações* é a data real da captura das informações dos sensores; o atributo *data_agregação* é um campo que é preenchido de acordo com a pesquisa feita pelo utilizador. Por exemplo, os sensores foram programados para recolher informações de cinco em cinco minutos, mas o utilizador deseja consultar os

dados de dez em dez minutos. A função do campo *data_agregação* é verificar todos os dados do campo *data_observações* que se encontra nesse intervalo e agregar de acordo com a consulta feita. A figura que se segue mostra-nos o nome do sensor que está a enviar informação, seguido da data da recolha das informações, data de agregação e os valores recolhidos pelos sensores.

id	nome	data_observacoes	data_agregacao	valores
32486	urn:ogc:def:property:OGC:Leaf	2012-06-01 04:13:27	2012-06-01 04:10:00	3.3100000198
32487	urn:ogc:def:property:OGC:Humi	2012-06-01 04:18:35	2012-06-01 04:10:00	46.8609046936
32488	urn:ogc:def:property:OGC:Temp	2012-06-01 04:18:35	2012-06-01 04:10:00	24.8387088775
32489	urn:ogc:def:property:OGC:Leaf	2012-06-01 04:18:35	2012-06-01 04:10:00	3.3100000197
32490	urn:ogc:def:property:OGC:Humi	2012-06-01 04:23:43	2012-06-01 04:20:00	46.3406066894
32491	urn:ogc:def:property:OGC:Temp	2012-06-01 04:23:43	2012-06-01 04:20:00	24.8387088775
32492	urn:ogc:def:property:OGC:Leaf	2012-06-01 04:23:43	2012-06-01 04:20:00	3.3100000198
32493	urn:ogc:def:property:OGC:Humi	2012-06-01 04:28:51	2012-06-01 04:20:00	46.8609046936
32494	urn:ogc:def:property:OGC:Temp	2012-06-01 04:28:51	2012-06-01 04:20:00	24.8387088775
32495	urn:ogc:def:property:OGC:Leaf	2012-06-01 04:28:51	2012-06-01 04:20:00	3.3100000196
32496	urn:ogc:def:property:OGC:Humi	2012-06-01 04:33:59	2012-06-01 04:30:00	46.8609046936
32497	urn:ogc:def:property:OGC:Temp	2012-06-01 04:33:59	2012-06-01 04:30:00	24.8387088775
32498	urn:ogc:def:property:OGC:Leaf	2012-06-01 04:33:59	2012-06-01 04:30:00	3.3100000196
32499	urn:ogc:def:property:OGC:Humi	2012-06-01 04:39:07	2012-06-01 04:30:00	46.3406066894
32500	urn:ogc:def:property:OGC:Temp	2012-06-01 04:39:07	2012-06-01 04:30:00	24.8387088775

Figura 21 – Exemplo de conteúdo da tabela *Leit_Sens*

4.3 Comunicações entre *Waspmote* e *Gateway*

Para a rede de sensores *waspmote* comunicar com o dispositivo *gateway*, é necessário configurar os dispositivos *XBee* das redes *waspmote*, de modo a poderem enviar as informações recolhidas pelos sensores. Mas para que isso seja viável deveremos defini-los com o mesmo *baud rate*, paridade, número de bits de paragem (para a comunicação série).

É de salientar que existem três possíveis topologias em que podemos definir os equipamentos *XBees* (*ZigBee coordenador*, *ZigBee router* e *ZigBee end device*). Para o *XBee* que irá permanecer ligado ao *gateway*, definimos como *ZigBee Coordenador* e o que fica conectado ao dispositivo *waspmote* colocamos como *ZigBee Router*.

A figura que se segue mostra-nos várias redes de sensores a enviar informações (como o endereço MAC, temperatura e a percentagem da bateria principal) para o dispositivo *gateway*.

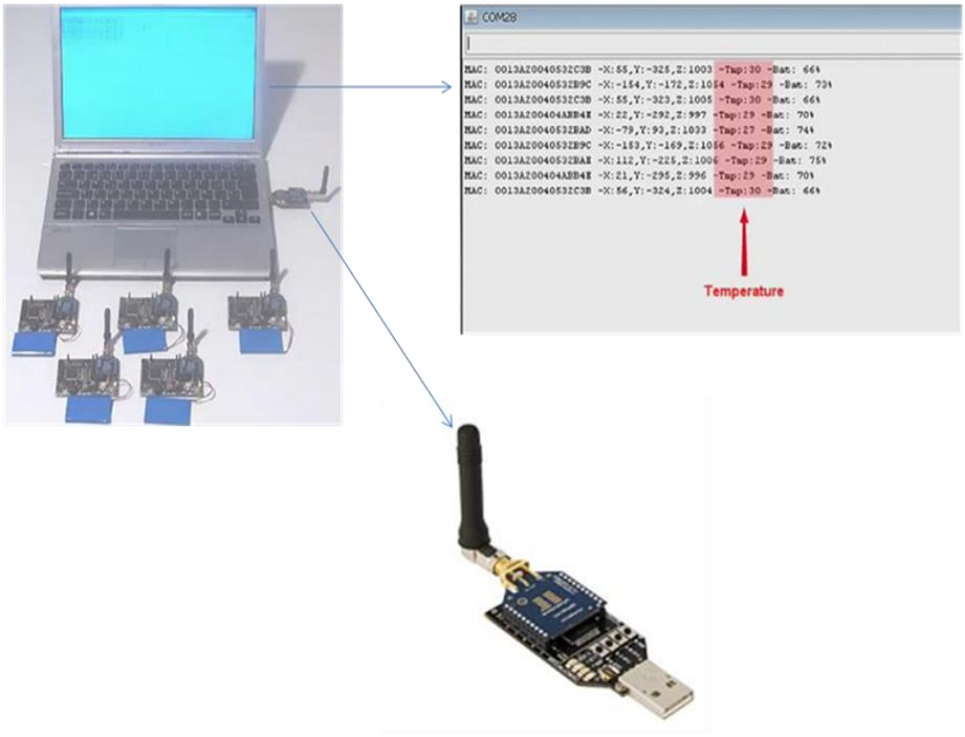


Figura 22 – Comunicação entre *waspmote* e *gateway*

No caso de não recebermos as informações do dispositivo *waspmote*, deveremos configurar os equipamentos *XBees*, usando um *software* X-CTU.

O X-CTU permite definir as configurações que pretendemos colocar nos dispositivos *ZigBee.* Podendo ainda verificar se o *ZigBee* do *waspmote* está a transmitir para o endereço (MAC) do *ZigBee gateway.* No nosso caso utilizamos o X-CTU para configurar os nossos *ZigBee*. A figura que se segue mostra-nos as configurações do *ZigBee* utilizando o *software* X-CTU.

Depois desta configuração, a rede está apta a realizar a comunicações entre os sensores *waspmote* e o dispositivo *gateway*. No nosso caso, as informações enviadas pelo dispositivo *waspmote* irão ser guardadas no *middleware* Muffin. No anexo B irá ser mostrada a comunicação entre as redes de sensores e o dispositivo *gateway*, detalhadamente.

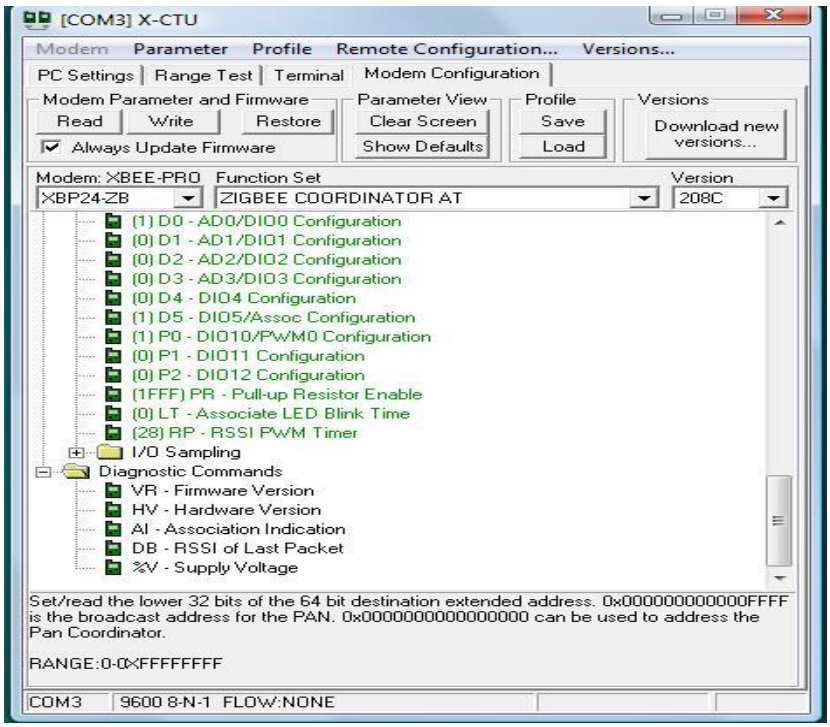


Figura 23 – X-CTU

4.4 MuFFIN

O MuFFIN é um *Middleware* situado entre as redes de sensores e os componentes de aplicações e tem como finalidade facilitar a comunicação e a coordenação desses componentes que estão distribuídos nos diversos computadores de uma rede [7].

O objetivo principal de um *middleware* é facilitar o desenvolvimento de aplicações, no presente caso facilitando a interação com as redes de sensores, em particular no que concerne a obtenção da informação.

Atualmente, as redes de sensores não apresentam uma forma padrão de interação com as aplicações. É necessário desenvolver *software* específico que implementa o modo particular de intersecção com cada rede.

O MuFFIN é constituído por vários módulos, mas só iremos utilizar o *ThingsGateway* para fazer a inserção das informações recolhidas pelos sensores, mas ainda poderíamos fazê-lo de outra forma utilizando o módulo *WS-Gateway* (que implementa a comunicação com aplicações de alto nível através de serviços na *Web*).

O *middleware* MuFFIN foi realizado de forma a permitir a interação entre várias instâncias, de forma a disponibilizar aos utilizadores informações vindas de diferentes plataformas instaladas em vários pontos do globo, reduzindo a troca de mensagens entre o utilizador e os centros de observação. A integração com os outros tipos de plataforma (aplicação) só é possível desde que estas suportem o padrão *Sensor Observation Service*

(SOS) [6], criando um sistema distribuído heterogéneo onde a comunicação com o *middleware* é feita através de um *gateway* específico.

Os serviços disponibilizados às aplicações de alto nível, para que os utilizadores possam interagir com o MuFFIN são: *deployModule, deployCode, instantiateService, installGateway, insertObservation, getCapabilities, readObservation, getModulesInfo, subscribeService, unsubscribeService*. Destes serviços vamos utilizar alguns para integrar com o nosso projeto, nomeadamente:

- *InstallGateway:* permite configurar um novo ponto de acesso para uma rede de objetos inteligentes. Esta operação recebe o novo *gateway* e instala-o.
- *ReadObservation*: serviço síncrono especificado na operação *GetObservations* do padrão SOS, que devolve um conjunto de observações que respeitam as restrições enviadas pelo utilizador.

O serviço *installThingGateway* é o responsável por receber os novos pontos de acesso e validar se estes têm as classes necessárias para serem executadas e registadas na camada de persistência.

O MUFFIN é extensível no que se refere à interação com as redes de sensores. O módulo *ThingsGateway* permite a instalação de classes JAVA (fornecidas através de um JAR) que especificam a interligação com redes de sensores específicas. A interligação com rede de sensores *waspmote* é feita via USB.

Podemos inserir as informações no *middleware* MuFFIN de várias formas, uma delas é através do módulo *WS-Gateway,* é preciso utilizar o serviço *insertObservation* que por sua vez fornece às outras aplicações a operação para inserirem as suas informações/observações, no entanto para tornar isso possível temos que respeitar as especificações do SOS. A outra forma de inserir as informações recolhidas pelos sensores é através do módulo *ThingsGateway,* criando uma classe que interaja com a porta USB, para receber os valores que as redes de sensores estão a enviar para o *gateway* e guarda-las no *middleware*; há que gerar um ficheiro JAR tendo em conta as exigências do MuFFIN, em seguida vamos fazer um programa para instalação do JAR utilizando serviços na *Web* no MuFFIN; posteriormente fazemos um outro programa que faz a chamada do *Web Service*, e depois nesse mesmo código iremos ler o ficheiro JAR.

No nosso projeto utilizamos o módulo *ThingsGateway* porque é mais rápido no envio das informações recolhidas pelos sensores para o MuFFIN.

Para consultar as informações recolhidas pelos sensores que se encontram armazenadas no MuFFIN, é necessário preencher um formulário que encontra-se na aplicação *Web.* Após submeter o formulário, será enviado um pedido através do serviço *Getobservation*. Nesse pedido teremos que especificar o *offering* (sensor1), o período de

início e fim da consulta (*beginPosition*) e quais os serviços específicos (*observedProperty*), que são os serviços de temperatura, de humidade do ar e de humidade das folhas. A tabela que se segue mostra como é feita esse pedido de acordo com o SOS. A seguir vão ser armazenados os dados na nossa base de dados; e por fim, vamos calcular a função pretendida (máximo, mínimo ou média) e apresentar o resultado (gráfico ou tabela) de acordo com a consulta feita pelo utilizador.

```
<?xml version="1.0" encoding="UTF-8"?>
<GetObservation xmlns="http://www.opengis.net/sos/1.0"
        xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
        xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.opengis.net/sos/1.0 http://schemas.opengis.net/sos/1.0.0/sosAll.xsd"
        service="SOS" version="1.0.0">
        <offering>urn:MyOrg:Sensor1</offering>
        <eventTime>
                <ogc:TM_During>
                        <ogc:PropertyName>om:samplingTime</ogc:PropertyName>
                        <gml:TimePeriod>
                                <gml:beginPosition>2012-05-31T22:40:12.000-06:00</gml:beginPosition>
                                <gml:endPosition>2012-06-01T09:40:22.000-06:00</gml:endPosition>
                        </gml:TimePeriod>
                </ogc:TM_During>
        </eventTime>
        <observedProperty>urn:ogc:def:property:OGC:SeeWeatherInAirSTS</observedProperty>
        <observedProperty>urn:ogc:def:property:OGC:SeeWeatherInAirSHS</observedProperty>
        <observedProperty>urn:ogc:def:property:OGC:SeeWeatherInAirSHF</observedProperty>
        <responseFormat>text/xml; subtype=&amp;quot;om/1.0.0&amp;quot;</responseFormat>
</GetObservation>
```

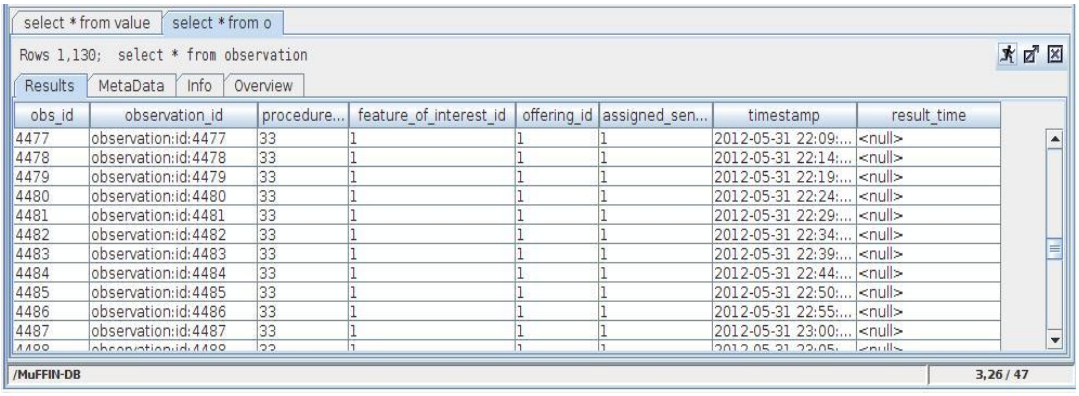
Tabela 4 – Pedido SOS

Capítulo 5 Resultados

Neste capítulo apresentamos os resultados obtidos com a realização dos testes efetuados, dando enfase ao consumo da bateria principal e aos resultados das leituras dos sensores. Para a realização destes testes utilizamos os sensores, *waspmote board*, *agriculture sensor board*, *gateway* e o computador. Esses testes foram realizados no laboratório LaSIGE durante um mês para verificação da precisão da informação recebida dos sensores.

5.1 Resultados das leituras dos sensores

Este teste tem como objetivo avaliar as leituras dos sensores. Podemos verificar com esses resultados se o *waspmote* executa o modo *deep sleep* corretamente como foi programado. A figura que se segue mostra-nos as datas e horas das observações feitas no terreno no dia 31 de Maio de 2012 no período entre as 22 e 23 horas.



obs_id	observation_id	procedure...	feature_of_interest_id	offering_id	assigned_sen...	timestamp	result_time
4477	observation:id:4477	33	1	1	1	2012-05-31 22:09:...	<null>
4478	observation:id:4478	33	1	1	1	2012-05-31 22:14:...	<null>
4479	observation:id:4479	33	1	1	1	2012-05-31 22:19:...	<null>
4480	observation:id:4480	33	1	1	1	2012-05-31 22:24:...	<null>
4481	observation:id:4481	33	1	1	1	2012-05-31 22:29:...	<null>
4482	observation:id:4482	33	1	1	1	2012-05-31 22:34:...	<null>
4483	observation:id:4483	33	1	1	1	2012-05-31 22:39:...	<null>
4484	observation:id:4484	33	1	1	1	2012-05-31 22:44:...	<null>
4485	observation:id:4485	33	1	1	1	2012-05-31 22:50:...	<null>
4486	observation:id:4486	33	1	1	1	2012-05-31 22:55:...	<null>
4487	observation:id:4487	33	1	1	1	2012-05-31 23:00:...	<null>

Figura 24 – Tabela de observação do MuFFIN

As redes de sensores *waspmote* foram programadas para recolher informações de cinco em cinco minutos. Como podemos verificar na Figura 24, as observações decorrem de acordo com a programação, confirmando que depois da recolha de cada informação, o *waspmote* executa o modo *deep sleep*; caso contrário, teríamos recolha de observações quase de forma constante.

A Figura 25 ilustra-nos os valores recolhidos referentes às observações apresentadas na figura anterior.

value_id	observation_id	phenomen...	value
3061	4477	10	47.5546226501
3062	4477	12	24.8387088775
3063	4477	11	3.2999999523
3064	4478	10	47.3811988830
3065	4478	12	24.8387088775
3066	4478	11	3.2999999523
3067	4479	10	47.2077713012
3068	4479	12	24.8387088775
3069	4479	11	3.2999999523
3070	4480	10	47.3811988830
3071	4480	12	24.8387088775
3072	4480	11	3.2999999523
3073	4481	10	47.3811988830
3074	4481	12	24.8387088775
3075	4481	11	3.2999999523
3076	4482	10	47.0343360900
3077	4482	12	24.8387088775
3078	4482	11	3.2999999523
3079	4483	10	47.3811988830
3080	4483	12	24.8387088775
3081	4483	11	3.2999999523
3082	4484	10	47.3811988830
3083	4484	12	24.8387088775
3084	4484	11	3.3100000584
3085	4485	10	47.2077713012
3086	4485	12	25.4838676452
3087	4485	11	3.3100000584
3088	4486	10	46.8609046936

Figura 25 – Valores das observações recolhidos

Estas informações vão ser mostradas aos utilizadores de duas formas: tabela e gráfico. Na tabela os dados apresentados têm uma limitação sobre os números de leituras, sendo exibidas vinte leituras por página.

Na Figura 26 estão indicados os valores máximos recolhidos pelos sensores durante uma hora. Podemos verificar a variação da temperatura e da humidade do ar. O valor máximo da temperatura foi de 27 °C e da humidade do ar de 47 %. Nota-se que estes valores ocorrem no período mais quente do dia. Para a produção deste gráfico é necessário o preenchimento de um formulário como mostra a Figura 19. De notar também que estes valores estão a ser recolhidos dentro de uma sala com climatização.

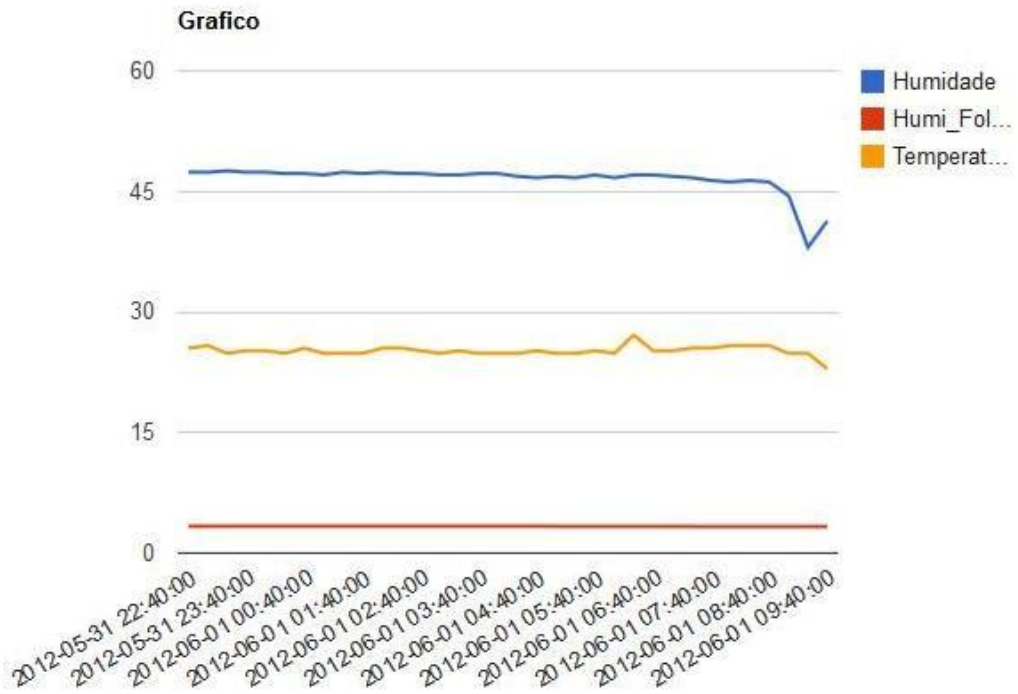


Figura 26 – Gráfico com os valores máximos para um período de 24 horas

Também podemos observar uma tabela com os valores mínimos registados durante um período de duas horas, e com as leituras agregadas de 20 em 20 minutos. Nota-se que a humidade das folhas não varia muito neste período porque tendo em conta que os testes foram feitos dentro do laboratório em ambiente controlado, as variações foram introduzidas artificialmente por nós com um borrifador. A figura que se segue mostra-nos algumas variações nas leituras do sensor de humidade das folhas nas duas últimas linhas em relação às outras.

|<< << >> >>|

nome	data_agregacao	minimo
urn:ogc:def:property:OGC:Humi	Fri 01, Jun 2012, 05:20:00	46.3406066894
urn:ogc:def:property:OGC:Leaf	Fri 01, Jun 2012, 05:20:00	3.2967741489
urn:ogc:def:property:OGC:Temp	Fri 01, Jun 2012, 05:20:00	24.8387088775
urn:ogc:def:property:OGC:Humi	Fri 01, Jun 2012, 05:40:00	46.5140419006
urn:ogc:def:property:OGC:Leaf	Fri 01, Jun 2012, 05:40:00	3.2967741489
urn:ogc:def:property:OGC:Temp	Fri 01, Jun 2012, 05:40:00	24.8387088775
urn:ogc:def:property:OGC:Humi	Fri 01, Jun 2012, 06:00:00	46.5140419006
urn:ogc:def:property:OGC:Leaf	Fri 01, Jun 2012, 06:00:00	3.2967741489
urn:ogc:def:property:OGC:Temp	Fri 01, Jun 2012, 06:00:00	24.8387088775
urn:ogc:def:property:OGC:Humi	Fri 01, Jun 2012, 06:20:00	46.6874771118
urn:ogc:def:property:OGC:Leaf	Fri 01, Jun 2012, 06:20:00	3.2967741489
urn:ogc:def:property:OGC:Temp	Fri 01, Jun 2012, 06:20:00	24.8387088775
urn:ogc:def:property:OGC:Humi	Fri 01, Jun 2012, 06:40:00	46.3406066894
urn:ogc:def:property:OGC:Leaf	Fri 01, Jun 2012, 06:40:00	3.2967737654
urn:ogc:def:property:OGC:Temp	Fri 01, Jun 2012, 06:40:00	24.8387088775
urn:ogc:def:property:OGC:Humi	Fri 01, Jun 2012, 07:00:00	46.1671905517
urn:ogc:def:property:OGC:Leaf	Fri 01, Jun 2012, 07:00:00	3.2967737654
urn:ogc:def:property:OGC:Temp	Fri 01, Jun 2012, 07:00:00	24.8387088775

Figura 27 – Tabela com os valores mínimos recolhidos

5.2 Nível do consumo da bateria

Este teste tem como objetivo avaliar o nível do consumo da bateria principal do dispositivo *waspmote* em diferentes modos de operação (*deep sleep*, *hibernate* e ON). Pretendemos analisar, depois uma semana de recolha de informações, qual irá ser o nível do consumo da bateria.

Para a realização destes testes partimos com uma bateria a 100% de carga. O teste consiste em recolher informação de cinco em cinco minutos e enviar a informação para o *middleware* MuFFIN. É de relembrar que a bateria fica inativa quando o dispositivo *waspmote* está a executar o modo *deep sleep* e *hibernate*, isto quer dizer que quando as redes de sensores estiverem a dormir é utilizada a bateria auxiliar [35].

O resultado do consumo da bateria durante esse período no modo de operação *deep sleep* foi de cerca de 2%.

A tabela que se segue apresenta os valores dos consumos do módulo *XBee* quando está a enviar e receber informações dos sensores no modo *sleep*.

	ON	**SLEEP**	**SENDING**	**RECEIVING**
XBee 802.15.4 PRO	56,68mA	0,12mA	187,58mA	57,08mA
XBee ZigBee	37,38mA	0,23mA	37,98mA	37,68mA
XBee ZigBee PRO	**45,56mA**	**0,71mA**	**105mA**	**50,46mA**
XBee 868	60,82mA	---	135mA	73mA
XBee 900	64,93mA	0,93mA	77mA	66mA

Tabela 5 – Consumo de energia dos módulos *XBee*

Como podemos analisar o *XBee ZigBee* consome muito menos bateria ao enviar e receber dados do que o *XBee ZigBee PRO*, mas apesar de consumir menos bateria também tem um alcance inferior (500 metros) no que pretendíamos.

Com os valores do consumo do *XBee ZigBee PRO* podemos deduzir que quanto menor tempo tiver o dispositivo *waspmote* no modo *deep sleep* maior será o consumo da bateria principal.

5.3 Discussão

Atualmente as nossas redes de sensores recolhem informações e enviam-nas de seguida para o *middleware* Muffin.

Uma ideia para pouparmos bateria nas redes de sensores é programar os sensores de forma a transmitir informações uma vez por dia, isto quer dizer que se recolhe informações várias vezes ao dia, mas só se envia uma única vez. Tendo em consideração que ao enviar informações gasta se 105mA e ao receber 50,46mA como podemos verificar na tabela 5.

Uma outra ideia era testar os sensores em ambientes não controlados e experimentar outros variantes como, por exemplo:

- Recolher informações de cinco em cinco minutos no modo *deep sleep* e enviar uma vez ao dia;
- No modo *hibernate,* recolher e enviar informações de cinco em cinco minutos;

Capítulo 6 Conclusão

As redes de sensores formam uma área de investigação bastante ativa atualmente. Utilizando redes de sensores é possível monitorizar ambientes de difícil acesso, como zonas de conflitos militar, regiões oceânicas, florestas. Além disso, podem ser aplicadas em várias áreas desde saúde à agricultura, e apresentam várias oportunidades. Estas redes são formadas por nós sensores, com capacidade de processamento, memória e interface de comunicação sem fios. Estes sensores permitem medir dados físicos do mundo real como temperatura, pressão, humidade, etc.

O clima dos Açores é ameno todo o ano. Assim a humidade relativa é geralmente elevada, o que conjugado com uma temperatura acima dos 25ºC pode originar uma situação climatérica propícia ao aparecimento de um fungo das pastagens. Este tipo de fungo pode provocar algumas doenças no gado como, por exemplo, a hipersensibilidade à luz.

Neste projeto implementamos um sistema que monitoriza e recolhe informação para ajuda na prevenção das razões que levam ao aparecimento da referida patologia.

No decorrer deste projeto, programámos os sensores para entrar no modo *deep sleep* de cinco em cinco minutos e depois enviar as informações recolhidas para o *middleware* MuFFIN. Os utilizadores podem consultar esses dados, através de uma aplicação *Web* que foi desenvolvida para o efeito.

A aplicação *Web* interage com o *middleware* MuFFIN utilizando o serviço na *Web*, para obtenção das informações solicitadas pelo utilizador. Estas informações irão ser guardadas numa base de dados externa.

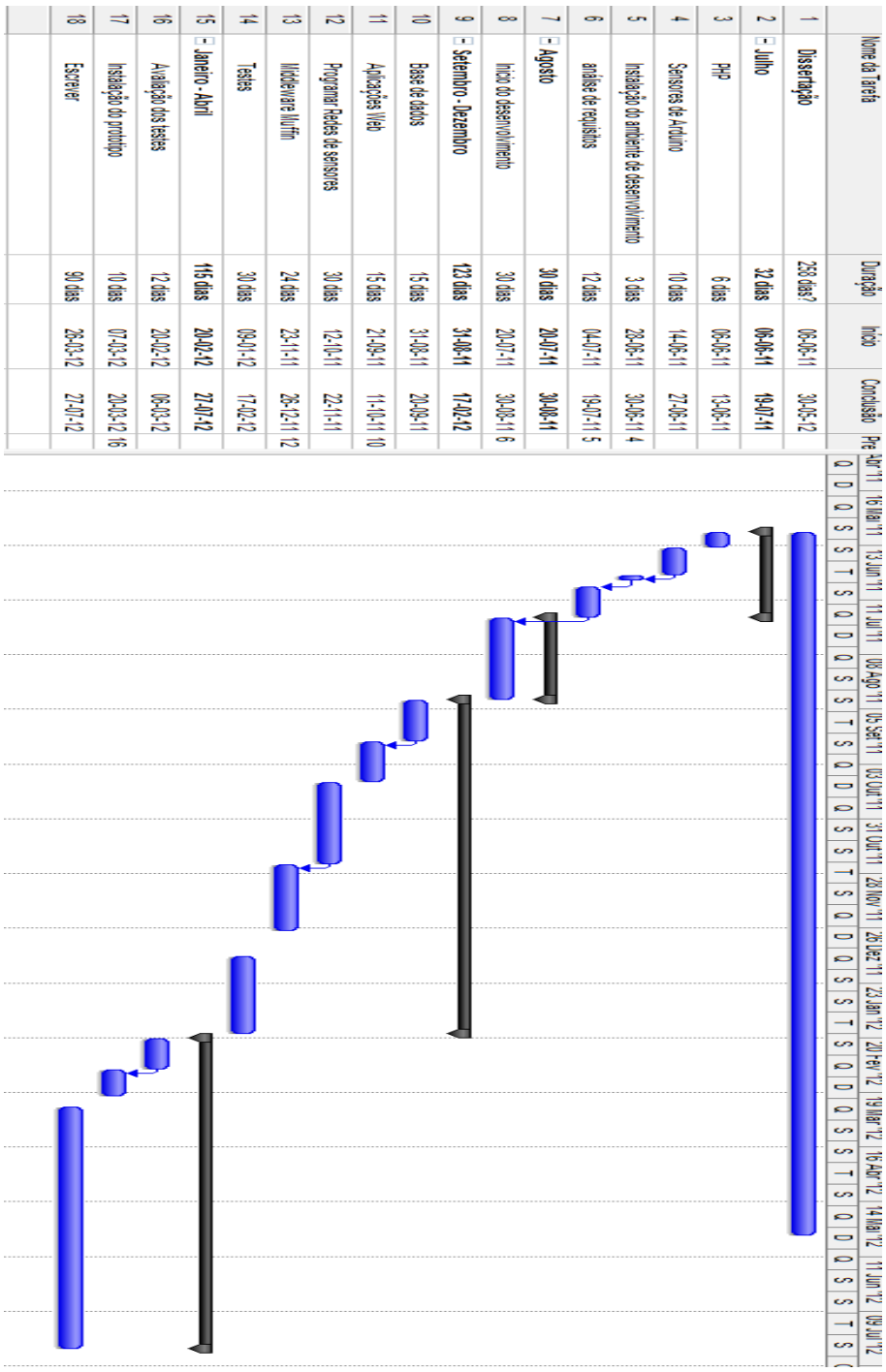
Depois da realização de vários testes entre a aplicação *Web* e o *middleware* MuFFIN concluímos que o tempo de resposta das consultas é satisfatório, mesmo quando o utilizador deseja consultar todas as leituras realizadas durante vários meses.

De uma forma geral consegue responder a todas as nossas espectativas.

Como trabalho futuro, será importante implementar na aplicação *Web* uma outra forma de disponibilizar os dados recolhidos pelos sensores, por exemplo, apresentar as informações num formato “CSV”, para depois ser tratado pelos especialistas.

Outra direção poderá se a inclusão de outros sensores como o de leitura de humidade do solo e o de leitura de temperatura do solo na nossa rede de sensores, tendo em conta que como esses sensores poder-se-ia dar outro tipo de informação sobre a exploração agrícola, como por exemplo, quando se proceder à rega dos terrenos ou determinar as condições propícias a efetuar uma sementeira.

Anexo A – Mapa de Gantt



	Nome da Tarefa	Duração	Início	Conclusão	Pre
1	Dissertação	259 dias?	06-06-11	30-05-12	
2	- Julho	32 dias	06-06-11	19-07-11	
3	PHP	6 dias	06-06-11	13-06-11	
4	Sensores de Arduino	10 dias	14-06-11	27-06-11	
5	Instalação do ambiente de desenvolvimento	3 dias	28-06-11	30-06-11	4
6	análise de requisitos	12 dias	04-07-11	19-07-11	5
7	- Agosto	30 dias	20-07-11	30-08-11	
8	Início do desenvolvimento	30 dias	20-07-11	30-08-11	6
9	- Setembro - Dezembro	123 dias	31-08-11	17-02-12	
10	Base de dados	15 dias	31-08-11	20-09-11	
11	Aplicações Web	15 dias	21-09-11	11-10-11	10
12	Programar Redes de sensores	30 dias	12-10-11	22-11-11	
13	Middleware Muffin	24 dias	23-11-11	26-12-11	12
14	Testes	30 dias	09-01-12	17-02-12	
15	- Janeiro - Abril	115 dias	20-02-12	27-07-12	
16	Avaliação dos testes	12 dias	20-02-12	05-03-12	
17	Instalação do protótipo	10 dias	07-03-12	20-03-12	16
18	Escrever	90 dias	26-03-12	27-07-12	

Anexo B – Comunicação entre *waspmote* e *gateway*

Para começarmos vamos colocar a pilha (bateria auxiliar), a bateria (bateria principal) e também conectar o cabo USB no dispositivo *waspmote* e no computador e por fim ativar o modo ON. Agora vamos abrir o *software waspmote*-IDE e começar a programar o dispositivo *waspmote*, lembre-se que o módulo *XBee* não pode estar conectado no *waspmote board* durante o período de programação. Agora é possível colocar o *waspmote board* no modo OFF e só depois desligar o cabo USB. A partir deste momento, pode-se conectar o *XBee* com a antena no dispositivo *waspmote* e ativar o modo ON.

Para recebermos as informações programadas nas redes de sensores o *gateway* necessita de estar com o *XBee* e a antena encaixadas e a seguir conectado a uma porta USB.

No caso de não recebermos as mensagens, devemos configurar os nossos *Xbees* usando um *software* X-CTU.

O X-CTU ajuda-nos a verificar e definir as configurações do *ZigBee*. É muito importante ter em mente que todos os *ZigBee* têm que estar na mesma rede. Verificar se o *ZigBee* do *waspmote* está a transmitir para o endereço (MAC) do *ZigBee gateway*.

Em primeiro lugar, é feita uma configuração do *ZigBee* que vai permanecer ligado ao *gateway* que posteriormente será conectado à porta USB. Temos que definir todos os *ZigBee* com o mesmo *baud rate*, e depois carregar em *test/query*, nesse momento abre uma janela e vamos carregar no *read*. Nesta altura temos que ter em atenção a escolha do *modem* certo para a configuração. Seguidamente é escolhido qual o *function set*, sendo muito importante saber qual o mais apropriado para o *modem* em questão, existem três situações possíveis: *ZigBee Coordenador*, *ZigBee Router*, *ZigBee End Device*.

Neste caso escolhemos o *ZigBee Coordenador*, e para o *ZigBee* que vai permanecer no *waspmote board* deverá ficar definido como *ZigBee Router*. Temos que verificar as

configurações e depois poderemos carregar no *write* do *software* X-CTU para terminar a configuração. Neste momento, é feita uma nova configuração para o outro *ZigBee* (que vai ser conectado no *waspmote board*), após este processo, é possível fazer a comunicação entre eles.

Bibliografia

[1] Akyildiz, I.F. and Weilian Su and Sankarasubramaniam, Y. and Cayirci, E. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.

[2] Alexandre Pereira, Carlos Poupa. Linguagens WEB, (2011).

[3] André de Castro, Luiz Dias, Pedro Botelho, Rafael Faria. Redes de Sensores Sem Fio. http://www.gta.ufrj.br/grad/10_1/rssf/introduo.html#Topic5, 2010.Recurso acedido em Outubro de 2011.

[4] Antônio Rogério Messias.Controle remoto e aquisição de dados via XBee/ZigBee (IEEE 802.15.4). http://www.rogercom.com/ZigBee/ZigBee.htm, 2008.Recurso acedido em Dezembro de 2011.

[5] Apache Friends. XAMPP. http://www.apachefriends.org/en/xampp.html. Recurso acedido em Agosto de 2011.

[6] Arthur Na, Mark Priest. Sensor Observation Service. http://portal.opengeospatial.org/files/?artifact_id=26667, 2007. Recurso acedido em Março de 2012.

[7] Bruno Valente. Serviços para a programação de redes de sensores, Dissertação para obtenção do grau de Mestre em Informática, Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa 2011.

[8] Carlos Serrão, Joaquim Marques. Programação com PHP 5.3, 2009.

[9] Daniel Viera. Introdução a Sistemas de Bancos de Dados, 2004.

[10] David E. Culler and Hans Mulder. Smart Sensors to Network the World. http://web.sau.edu/lilliskevinm/wirelessbib/CullerMulder.pdf, pages 84–91, 2004. Recurso acedido em Novembro de 2011.

[11] Digi – Your M2M solutions expert. Wireless Mesh Networking ZigBee vs. DigiMesh. http://www.digi.com/pdf/wp_ZigBeevsdigimesh.pdf, 2008. Recurso acedido em Março de 2012.

[12] Emmerich, W. Software Engineering and Middleware: A Roadmap. In the future of software engineering - 22nd int. conf. on software engineering (ICSE2000), pages 117 – 129. ACM Press, May 2000.

[13] Francisco Martins and Luís Lopes and João Barros. Towards the Safe Programming of Wireless Sensor Networks. In Proceedings of Programming Language Approaches to Concurrency and Communication-cEntric Software (PLACES), vol.17 of EPTCS, pages 49–62, 2010.

[14] G. Anastasi, M. Conti, M. Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. Ad Hoc Networks, 7(3):537–568, May 2009.

[15] I. F. Akyildiz, T. Melodia and K. R. Chowdhury.A survey on wireless multimedia sensor networks. Computer Networks, v. 51, n. 4, pages 921–960, 2007.

[16] Ingo Simonis and Johannes Echterhoff. Draft OpenGIS Web Notification Service-Implementation Specification. http://portal.opengeospatial.org/files/?artifact_id=18776, 2006. Recurso acedido em Dezembro de 2011.

[17] J. A. Stankovic, "A network virtual machine for real time-coordination", The Real-Time Computing Laboratory, University of Virginia, http://www.cs.virginia.edu/nest, 2002. Recurso acedido em Dezembro de 2011.

[18] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin and D. Ganesan, "Building efficient wireless sensor networks with low-level naming", In Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles, Banff, Alberta, Canada, ACM Press, pages 146 – 159, 2001.

[19] J. Yick, B. Mukherjee, and D. Ghosal, Wireless sensor network survey, Computer Networks, vol. 52, no. 12, pages 2292–2330, 2008.

[20] José Luís Pereira. Tecnologia de Bases de Dados, (1998**).**

[21] Kris Pister. My view of sensor networks in 2010. http://www.eecs.berkeley.edu/%7Epister/SmartDust/in2010. Recurso acedido em Maio de 2012.

[22] Krishnamachari, B. *Networking Wireless Sensor*. Cambridge University Press, 2005.

[23] Loureiro, A. A, Ruiz, L. B, Nogueira, J. M. S, and Mini, R. A. Redes de sensors sem fios. http://homepages.dcc.ufmg.br/~loureiro/cm/docs/sbrc03.pdf. 2003. Recurso acedido em Dezembro de 2011.

[24] Luís Abreu. HTML 5, (2011).

[25] Luís Damas. SQL, (2005).

[26] N. Gross, "21 ideas for 21st century", Business Week, August 1999.

[27] R. Malladi and D. P. Agrawal, "Current and future applications of mobile and wireless networks".http://doi.acm.org/10.1145/570907.570947. vol. 45, pages 144–146, 2002.

[28] S. Megerian, F. Koushanfar, G. Qu, G. Veltri and M. Potkonjak, "Exposure in wireless sensor networks: theory and practical solutions", Wireless Networks, Kluwer Academic Publishers, ISSN 1022-0038, vol. 8, no.5, pages 443–454,2002.

[29] S. Tilak, N.B. Abu-Ghazaleh and W. Heinzelman, "Infrastructure trade offs for sensor networks". http://doi.acm.org/10.1145/570738.570746, ISBN 1-58113-589-0, pages 49–58, 2002. Recurso acedido em Dezembro de 2011.

[30] Serial Programming/Serial Java. http://en.wikibooks.org/wiki/Serial_Programming/Serial_Java. Recurso acedido em fevereiro de 2012.

[31] Wikipédia - A enciclopédia livre. http://pt.wikipedia.org/wiki/Rel%C3% B3gio_de_tempo_real. Recurso acedido em Janeiro de 2012.

[32] Wireless Distributed Communications.http://www.libelium.com/documentation/waspmote/waspmote-zigbee-networking_guide.pdf. Recurso acedido em Fevereiro de 2012.

[33] Wireless Distributed Communications. http://www.libelium.com/documentation/waspmote/waspmote-rtc-programming_guide.pdf. Recurso acedido em Março de 2012.

[34] Wireless Distributed Communications. http://www.libelium.com/forum/viewtopic.php?f=14&t=9553. Recurso acedido em Setembro de 2012.

[35] Wireless Distributed Communications.http://www.libelium.com/documentation/waspmote/waspmote-technical_guide_eng.pdf. Recurso acedido em Outubro de 2011.

[36] Wireless Distributed Communications. http://www.libelium.com/documentation/waspmote/agriculture-sensor-board_eng.pdf. Recurso acedido em Outubro de 2011.

[37] Wireless Distributed Communications. http://www.libelium.com/products/waspmote/hardware. Recurso acedido em Outubro de 2011.

[38] Wireless Distributed Communications. http://www.libelium.com/products/waspmote/OTA. Recurso acedido em Outubro de 2011.

[39] Wireless Distributed Communications. http://www.libelium.com/products/waspmote. Recurso acedido em Outubro de 2011.

[40] Wireless distributed communications. http://www.libelium.com/applications/agriculture. Recurso acedido em Janeiro de 2012.

[41] Wireless Distributed Communications. http://www.libelium.com/development/waspmote/example023.Recurso acedido em Março de 2012.

[42] Wireless Distributed Communications. http://www.libelium.com/development/waspmote/example044. Recurso acedido em Maio de 2012.

[43] Wireless Distributed Communications. http://www.libelium.com/forum/viewtopic.php?f=15&t=7701. Recurso acedido Dezembro de 2011.

[44] World Wide Web Consortium (W3C). SOAP. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/. Recurso acedido no dia 26 de Fevereiro de 2011.

[45] World Wide Web Consortium (W3C). Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl/. Recurso acedido no dia 10 de Outubro de 2011.

[46] Yonggang Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Residual energy scans for monitoring wireless sensor networks. In *IEEE Wireless Communications and Networking Conference (WCNC'02)*, Orlando, FL, USA, March 2002.