WeSenseNET - Plataforma de Visualização, Gestão Remota e Agregação Múltipla de Dados

António Pedro Domingues Leite

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

2012

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: António Pedro Domingues Leite, Nº 1060370, 1060370@isep.ipp.pt

Orientação científica: Manuel Carlos Malheiro de Carvalho Felgueiras, mcf@isep.ipp.pt

Empresa: EVOLEO Technologies, LDA.

Supervisão: Rodolfo Manuel Maia da Cruz Martins, rodolfo.martins@evoleotech.com

António João dos Santos Sousa, antonio.sousa@evoleotech.com

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

4 de Dezembro de 2012

Agradecimentos

Várias pessoas e instituições contribuíram directa ou indirectamente para o desenvolvimento deste trabalho e gostaria de aqui lhes exprimir os meus agradecimentos.

Em relação às instituições, gostaria de agradecer à *Evoleo Technologies*, nas pessoas do Eng.º Rodolfo Martins e do Eng.º António Sousa, a oportunidade oferecida para fazer parte de um dos principais projectos da empresa e ao Instituto Superior de Engenharia do Porto, nas pessoas do Eng.º Alberto Sampaio, Eng.º Vieira dos Santos e Eng.º Jorge Silva, que me impulsionaram na realização deste trabalho.

Em relação aos agradecimentos pessoais, não poderia deixar de começar pelo meu orientador, Professor Doutor Carlos Felgueiras, pelo empenho, apoio, compreensão e pela confiança demonstrada no meu trabalho desde o seu início. A todos os colegas e amigos que tão importantes para mim foram ao longo do curso, nomeadamente o Diogo Ribeiro, Nuno Carvalho, e André Oliveira pela amizade, constante confiança e encorajamento demonstrados.

Por último e principalmente, agradeço à minha família e à minha namorada, por todo o apoio, confiança e, por vezes, sacrifícios que fizeram para me proporcionar melhores condições em vários aspectos da minha vida em particular durante o tempo de estudo. Sem eles não seria possível a conclusão de mais esta etapa.

Resumo

"Inventions have long since reached their limit, and I see no hope for further development."

Julius Sextus Frontinus, Engenheiro Romano, 10 D.C.

Nos últimos anos, o avanço da tecnologia e a miniaturização de diversos componentes de electrónica associados a novos conceitos têm permitido nascer novas ideias e projectos, que até há alguns anos não passariam de ficção científica. Talvez o exemplo mais acabado seja actualmente o *smartphone*, um pequeno bloco de *hardware* e *software*, com capacidade de processamento que ultrapassa várias vezes o dos computadores com uma dúzia de anos. Estas capacidades têm sido utilizadas em comunicações, blocos de notas, agendas e até entretenimento. No entanto, podem ser reutilizadas para ajudar a resolver algumas limitações/constrangimentos da actualidade. Dentro destes destacam-se a gestão de recursos escassos. Com efeito, o consumo de energia eléctrica tem aumentado como consequência directa do desenvolvimento global e aumento do número de aparelhos eléctricos. Uma percentagem significativa de energia eléctrica tem sido produzida através de recursos não-renováveis de energia. No entanto, a dependência energética, associada à subida de preços e a redução das emissões de gases do efeito estufa, estimula o desenvolvimento de novas soluções que permitam lidar com esta situação.

O desempenho energético por sua vez depende não só das características da estrutura, mas também do comportamento do utilizador. O desempenho energético dos edifícios é muito importante, uma vez que os respectivos consumos são responsáveis por mais de metade do total da energia produzida. Desta forma, a fim de alcançar um melhor desempenho é importante não só considerar o desempenho de estrutura, mas também monitorizar o comportamento do utilizador. Esta última questão coloca várias limitações, uma vez que depende muito do tipo de utilizador.

Um dos conceitos actuais emergentes são as chamadas redes de sensores sem fio. Com esta tecnologia, pequenos módulos podem ser desenvolvidos com muitas possibilidades de

conectividade, com elevado poder de processamento e com grande autonomia, sem serem excessivamente caros. Isto proporciona os meios para implementar vários dispositivos em toda a instalação, para recolher uma variedade de dados, sendo posteriormente armazenados num servidor.

Os blocos fundamentais da infra-estrutura de sensores do projecto foram concebidos na Evoleo Technologies em simultâneo com o decorrer do estágio. Estes blocos recolhem dados específicos na instalação, e periodicamente enviam para o servidor central os valores recolhidos, onde são armazenados e colocados à disposição do utilizador. Os dados recolhidos podem então ser apresentados ao utilizador, proporcionando um registo de consumo de energia associado a um dado período de tempo. Uma vez que todos os dados são armazenados no servidor, podem ser efectuados estudos para determinar o uso típico, possíveis problemas em aparelhos, a qualidade da energia eléctrica, etc., permitindo determinar onde a energia está a ser eventualmente desperdiçada e fornecendo dados ao utilizador para que este possa proceder a alterações, tendo por base dados recolhidos num dado período.

O objectivo principal deste trabalho passa por estabelecer a ligação entre o nível máquina e o nível de utilizador, isto é, uma plataforma de interacção entre dispositivos e administrador da instalação. Fornecer os dados de uma forma fácil e sem necessidade de instalação de *software* específico em cada dispositivo que se pretenda utilizar para monitorizar foi uma das principais preocupações das fases de concepção do projecto.

Palavras-Chave

Gestão do consumo energético, Consumo energético em edifícios, Monitorização remota, Configuração remota, Redes de sensores sem fio.

Abstract

"Inventions have long since reached their limit, and I see no hope for further development."

Julius Sextus Frontinus, Roman Engineer, 10 A.C.

In recent years, the advancement of technology and the miniaturization of many electronic components, associated with new concepts have allowed growth of new ideas and projects, which until a few years ago would not exist outside science fiction. Perhaps the best example now is the smartphone, a small piece of hardware and software, with a processing power that exceeds several times that of computers with a dozen years. These capabilities have been used in communications, notebooks, diaries, and even entertainment. However they can be reused to help solving some limitations / constraints in the present day. Within these we highlight the management of scarce resources. Indeed, the electric energy consumption has increased as a direct result of the global development and with the increase in electrical devices numbers. A significant percentage of electricity has been produced by non-renewable resources of energy. However, energy dependence associated with the rising prices and reduction of emissions of greenhouse gases stimulates the development of new solutions to deal with this situation.

The energy performance in turn, depends not only on the characteristics of the structure, but also on user behaviour. The energy performance of buildings is very important, since their consumption accounts for more than half of the total energy produced. Thus, in order to achieve better performance it's not only important to consider the performance of the structure, but also monitor user behaviour. This last question raises several limitations, since much depends on the type of user.

One of the current emerging concepts is called Wireless Sensor Networks. With this technology, small modules can be developed with many connectivity possibilities, with high processing power and with great autonomy, without being overly expensive. This

provides the means to implement various devices throughout the facility to collect a variety of data that are later stored in a server.

The building blocks of the infrastructure (nodes) were developed in the project at Evoleo Technologies concurrently with the internship. These blocks collect specific data on the installation and periodically send the collected data to the server, where they are stored and made available to the user. The data gathered can then be displayed to the user, providing a record of power consumption associated with a given period of time. Since all data is stored on the server, studies can be made to determine the typical use, possible problems in appliances, power quality, etc., to determine if and where energy is being wasted, providing data to the user so that he can proceed and apply changes to the application.

The primary objective of this study is to establish the connection between the machine level and user level, i.e. a platform for interaction between devices and the installation administrator. Providing data in an easy way and without the need to install special software on each device used for monitoring was a major concern of the design stages of the project.

Keywords

Energy management, Energy consumption in buildings, Remote monitoring, Remote configuration, Wireless sensor networks.

Nota ao leitor

A maior parte da literatura referente à área onde se insere este trabalho está publicada em inglês, língua que acolhe a generalidade dos termos que nela se tornaram consagrados e assim entraram na linguagem oral corrente. A escrita em português dum documento desta natureza vê-se deste modo frequentemente confrontada com a necessidade de escolher entre os termos comummente usados ou a respectiva tradução, de forma a não afectar a compreensão e a clareza da exposição. Decidimos por isso adoptar o emprego de termos em português, sempre que não suscitarem dúvidas, e dos termos consagrados em inglês nos restantes casos.

O estilo itálico utiliza-se para distinguir os termos que não pertencem ao vocabulário português e para destacar uma determinada palavra ou expressão.

Os acrónimos são geralmente apresentados em maiúsculas e em estilo normal, independentemente de se referirem a expressões escritas em português ou em inglês, incluindo-se no início deste documento uma lista para facilitar a sua identificação.

Por fim, convém referir que o presente documento não segue o novo acordo ortográfico.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

AJAX	-	*Asynchronous JavaScript And XML*
API	-	*Application Programming Interface*
BD	-	Base de Dados
BSD	-	*Berkeley Software Distribution*
CAD	-	*Computer Aided Design*
CI	-	Circuito Integrado
CSS	-	*Cascading Style Sheet*
CSV	-	*Comma-separated values*
DB	-	*Database*
EES	-	*Energy, Environment and Sustainability*
ECSS	-	*European Cooperation on Space Standardization*
EMC	-	*ElectroMagnetic Compatibility*
ESA	-	*European Space Agency*
ETAR	-	Estação de Tratamento de Águas Residuais
FFD	-	*Full Function Device*
FMEA	-	*Failure Mode and Effects Analysis*
FMECA	-	*Failure Mode, Effects and Criticality Analysis*
GPS	-	*Global Positioning System*
GSM	-	*Global System for Mobile*
GUI	-	*Graphical User Interface*
HSDPA	-	*High-Speed Downlink Packet Access*
HTML	-	*HyperText Markup Language*
http	-	*HyperText Transfer Protocol*
HTTPS	-	*HyperText Transfer Protocol Secure*
HW	-	*Hardware*
IDE	-	*Integrated Development Environment*
IP	-	*Internet Protocol*
I&D	-	Investigação e Desenvolvimento

IMEI	-	*International Mobile Equipment Identity*
ISEP	-	Instituto Superior de Engenharia do Porto
JSON	-	*JavaScript Object Notation*
MAC	-	*Media Access Control*
MEEC	-	Mestrado em Engenharia Electrotécnica e Computadores
MTB	-	*Memory Test Board*
OHA	-	*Open Handset Alliance*
PDA	-	*Personal Digital Assistant*
PDF	-	*Portable Document Format*
PDU	-	*Power Distribution Unit*
PHP	-	*Hypertext Preprocessor*
PLC	-	*Programmable Logic Controller*
PNG	-	*Portable Network Graphics*
PSA	-	*Part Stress Analysis*
RAM	-	*Random Access Memory*
RFD	-	*Reduced Function Device*
SCADA	-	*Supervisory Control and Data Acquisition*
SDK	-	*Software Development Kit*
SGBD	-	Sistema de Gestão de Base de Dados
SO	-	Sistema Operativo
SQL	-	*Structured Query Language*
SW	-	*Software*
TCP	-	*Transmission Control Protocol*
UDP	-	*User Datagram Protocol*
UI	-	*User Interface*
VHDL	-	*VHSIC (Very High Speed Integrated Circuits) Hardware Description Language*
WCA	-	*Worst Case analysis*
WSN	-	*Wireless Sensor Network*
WYSIWYG	-	*What You See Is What You Get*
XHR	-	XMLHttpRequest
XML	-	*Extensible Markup Language*

1. INTRODUÇÃO

O presente documento satisfaz parte dos requisitos da Tese/Dissertação do segundo ano do Mestrado em Engenharia Electrotécnica e Computadores (MEEC), ramo de Automação e Sistemas, do Instituto Superior de Engenharia do Porto (ISEP), correspondendo ao relatório final do trabalho efectuado nesse âmbito. O trabalho desenvolvido inclui-se no desenvolvimento de uma infra-estrutura de informação recolhida por uma rede de sensores com capacidade de comunicação sem fios, permitindo a recolha e armazenamento dos dados recolhidos, bem como a visualização e controlo/reconfiguração remota de cada dispositivo instalado.

Este trabalho foi desenvolvido como parte de um estágio curricular na empresa Evoleo Technologies, sediada na Maia. O trabalho realizado faz parte de um projecto abrangente e de dimensão elevada em desenvolvimento na empresa, tendo sido atribuída a tarefa de definição de proposta, projecto e desenvolvimento da infra-estrutura ao autor do presente documento.

1.1. CONTEXTUALIZAÇÃO

Desde sempre se tem verificado que o desenvolvimento implica um aumento do consumo de energia nas suas diferentes formas, que tem influência nefasta na qualidade ambiental.

Cai-se assim numa situação em que, por um lado, se visa melhorar a qualidade de vida e, por outro, se piora aspectos fundamentais da qualidade de vida dessa mesma sociedade, colocando mesmo em causa a sua sobrevivência no futuro [1]. O consumo energético em edifícios é importante dado que representa globalmente 40% de toda a energia consumida [2]. Estes dados assumem especial importância para Portugal/Europa que é externamente dependente da energia fóssil, sendo particularmente importantes para Portugal que importa cerca de 80% da sua energia primária [3]. A utilização eficiente da energia é de elevada importância devido não só a factores económicos (elevado preço dos combustíveis, instabilidade dos mercados, etc.) mas também ambientais, (e.g. a emissão de gases de efeito estufa, NOx, delapidação de recursos naturais, etc.) e sociais. Por forma a obter um desenvolvimento balanceado, foi introduzido o conceito de *desenvolvimento sustentável*. A União Europeia estabeleceu um conjunto de objectivos ambiciosos até 2020 mas apenas uma pequena parte do seu orçamento foi investido em programas de educação em escolas e campanhas de informação [4]. Como consequência disso, atingiu-se uma boa eficiência energética do ponto de vista de produtos desenvolvidos, mas existem ainda grandes desperdícios de energia que dependem do consumidor final, isto é, a eficiência do utilizador. Existem vários estudos que indicam a eficiência do utilizador como o principal responsável pelos elevados desperdícios energéticos. Nalguns casos, existem relatos de instalações onde mais energia é consumida fora do horário de trabalho (56%) do que durante o horário de trabalho (44%) [5]. Lindelöf e Morel [6] estudaram os padrões de acender e apagar luzes dos ocupantes de uma determinada instalação. Durante as horas de trabalho, verificaram que devido à localização do interruptor (junto à porta, afastado do alcance do braço a partir de qualquer secretária), as luzes eram mantidas acesas durante todo o dia, mesmo quando não eram necessárias. Mahdavi et al. [7] avaliou 48 escritórios de três edifícios, utilizando câmaras digitais e outros dispositivos de recolha de dados. Foi observado que os trabalhadores desses escritórios passavam mais de metade do tempo afastados da secretária de trabalho, contudo, mantendo o equipamento ligado durante todo o dia. Adicionalmente, segundo dados recolhidos relativos à eficiência energética em edifícios, indicam que os edifícios públicos são mais relevantes do que os privados. Por exemplo, considerando o total de energia produzido em Portugal, os edifícios públicos são responsáveis por 71% do consumo face aos 23% dos edifícios privados [8]. De forma a tentar ultrapassar estes problemas, alguns países lançaram programas nacionais de promoção do desenvolvimento sustentável (e.g. PPEC [9]), mas concentram-se

essencialmente na publicação de linhas de orientação para prevenir o desperdício energético.

A necessidade de redução de consumos energéticos em edifícios novos levou a que durante a fase do respectivo projecto sejam tomadas cada vez mais medidas que beneficiem a sua eficiência energética. No entanto, os edifícios já existentes são de longe os que constituem o maior número, pelo que a respectiva racionalização energética apenas se pode fazer através da monitorização de consumos e outras grandezas e respectiva aquisição de dados objectivos. Por um lado, essa aquisição enfrenta frequentemente limitações decorrentes da ausência de infra-estruturas orientadas para esse efeito. Por outro, tais dados são fundamentais para a adequada gestão, i.e. redução de consumos e aumento da eficiência energética.

Do exposto, torna-se evidente o desenvolvimento de uma solução que permita a racionalização energética de um edifício. Pretende-se com este projecto desenvolver uma infra-estrutura que permita medir em *tempo-real* um conjunto de grandezas (energia eléctrica, temperatura, estado de detectores, consumos de água, etc.) cuja adequada gestão permita a respectiva racionalização. Os blocos principais da infra-estrutura a desenvolver serão os módulos de *hardware* que permitam a observação dos valores das grandezas pretendidas, associado a um meio de interface com o utilizador.

Este trabalho faz parte da solução *WeSense Network*, desenvolvida na empresa Evoleo Technologies durante os últimos meses, que pretende ser uma solução para estes problemas. A solução completa envolve componentes de *hardware* e *software*, tendo o presente trabalho maior ênfase na última.

1.2. A EMPRESA EVOLEO TECHNOLOGIES

A Evoleo Technologies foi fundada em 16 de Janeiro de 2007 com o objectivo estratégico de enveredar pela área de negócio do Espaço. Essa opção foi tomada pelo seu fundador,

Figura 1 - Logótipo Evoleo Technologies.

Rodolfo Martins, que detinha *know-how* específico, necessário para trabalhar nesta área. O nível de conhecimento técnico exigido para trabalhar nesta área é elevado, uma vez que se tratam de equipamentos projectados e desenvolvidos para operar no espaço, estando desta forma sujeitos a condições extremas de temperatura e radiação cósmica. A prototipagem de equipamentos para este sector é dispendiosa, chegando alguns circuitos integrados (CI's) certificados para o espaço a custar cerca de € 15000.00 [10], e podem apenas ter uma possibilidade para serem programados. Trata-se assim de uma operação com elevado nível de risco dado que caso haja algum erro, o componente pode ficar inutilizado.

As agências espaciais, em conjunto com as empresas que produzem os satélites enviados para o espaço, lançam concursos para *Demonstration Payloads* (cargas úteis de demonstração), que são essencialmente blocos de experiências científicas que se pretende que sejam realizadas no espaço, obedecendo a especificações de peso, comunicação, não podendo comprometer a integridade do satélite seja em que circunstâncias for [11].

A Evoleo participou no TDP8 (*Technology Demonstration Payload*) contribuindo com vários dos blocos funcionais presentes na Figura 2 como o Data Interface (Data I/F), *Power Distribution Unit* (PDU) e *Memory Test Board* (MTB), sendo que este último pretende ser uma base tecnológica para avaliar o impacto das radiações cósmicas em memórias *Random-Access Memory* (RAM) e Flash que não são produzidas para as condições extremas do Espaço (memórias não *Rad-Hard*) [12].

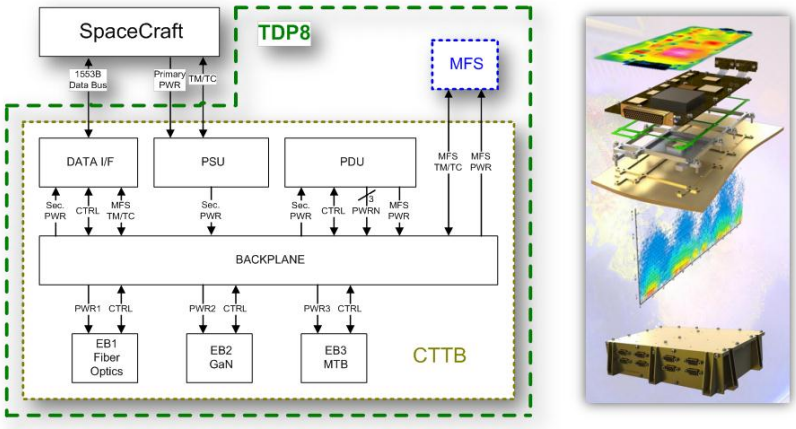


Figura 2 - Esquema representativo do Alphasat TDP8 [10].

Actualmente, a Evoleo opera nos negócios Aeroespacial e Industrial, apostando em competências adquiridas que permitem o desenho de sistemas electrónicos de elevada complexidade, automação e controlo.

No entanto a equipa da Evoleo, que é actualmente constituída por 11 elementos, maioritariamente antigos alunos do ISEP, detêm assim competências em diversas áreas, que permitiram à empresa adquirir um conjunto de conhecimentos técnicos exigentes, decorrentes principalmente de trabalhos desenvolvidos para a *European Space Agency* (ESA), e de grupos de Investigação e Desenvolvimento (I&D), Universidades e Politécnicos [13].

Algumas das competências da equipa incluem [14]:

- Desenho e Implementação de *Hardware* Analógico;
  - Projecto e desenho assistido por computador (*Computer Aided Design* - CAD);
  - Análises de pior caso (*Worst Case Analysis* - WCA);
  - Análise de *stress* de componentes (*Part Stress Analysis* - PSA);
  - Análise de modos de falha (*Failure Mode and Effects Analysis* - FMEA);

- Desenho e Implementação de *Hardware* Digital;
  - Projecto e CAD;
  - *Field Programmable Gate Array* (FPGA) das marcas Altera, Xilinx e Actel;
  - Microcontrolador ou Microprocessador 8/16/32 bit (simples e *multi-core* sobre FPGA);
  - *VHSIC (Very High Speed Integrated Circuits) Hardware Description Language (VHDL)* e *firmware* em C (FW-C);

- Implementação de *Software* para Sistemas Embebidos;
  - Utilização de sistemas operativos de tempo real;
  - Desenvolvimento em plataforma NI LabView;
  - Desenvolvimento em ambiente Windows ou Linux;

- Sistemas de instrumentação, automação e controlo;
  - Utilização de autómatos;
  - Utilização de tecnologia *National Instruments* de rápida implementação e fácil utilização;

- Engenharia de Sistemas;
  - Apoio à integração e concretização de projectos de áreas abrangentes;
  - Teste e qualificação eléctrica, funcional e *ElectroMagnetic Compatibility* (EMC);

- Processo de Desenvolvimento e Qualidade;
- Processos internos de acordo com normas ESA-ECSS;
- Adaptação das normas/processos para aplicações industriais;

No que concerne às normas ESA-ECSS, estas dizem respeito ao cliente principal na área Espacial, a ESA, que obriga ao cumprimento de diversas normas e regulamentos, este conhecimento é considerado uma mais-valia para a Evoleo [15].

1.2.1. ÁREAS DE NEGÓCIO

As principais áreas de negócio da Evoleo são quatro. Algumas estão focadas no mercado nacional e outras no internacional, como é o caso do sector Aeroespacial. As áreas prioritárias definidas pela empresa são as seguintes [15]:

- Aeroespacial;
- Energia;
- Transportes;
- Saúde;

A área aeroespacial caracteriza-se pelo desenvolvimento de sistemas críticos de elevada complexidade a nível de *hardware* e *software*, para serem colocados em satélites (e.g. Alphasat TDP8). Este sector é o *core business* da empresa, resultado da experiência acumulada nesta área de negócio com os anteriores trabalhos realizados para o principal cliente (ESA). Os requisitos dos produtos desenvolvidos para este sector são mais elevados, acrescidos de normas e regulamentos específicos.

A área de Energia tem em conta a crescente dependência sobre recursos energéticos não renováveis, e a consciencialização geral da necessidade de uma utilização da energia de forma mais sustentável pelo que a Evoleo Technologies identifica este sector como uma área principal de negócio, desenvolvendo equipamentos que permitam a monitorização de consumos energéticos e recursos, qualidade da energia eléctrica e optimização energética em edifícios.

A área de transportes caracteriza-se pelo desenvolvimento de sistemas que permitam a determinação do estado das linhas ferroviárias. Estes sistemas tornam-se relevantes na prevenção de acidentes e ajuda à manutenção das linhas. A Evoleo Technologies desenvolveu vários trabalhos para este sector, designadamente a monitorização constante das vibrações dos *bogies* do alfa pendular na linha Lisboa-Porto, com geolocalização, por forma a determinar quais os sectores que mais urgentemente precisam de manutenção [16].

A área da saúde foi também identificada como importante face à observação do crescimento de alguns segmentos da população que necessitam de cuidados médicos na hora, mas que têm dificuldade em obtê-los.

1.2.2. PRODUTOS

A empresa levou a cabo o desenvolvimento interno de alguns produtos, que se deram por finalizados na fase de criação de um protótipo. A capacidade produtiva associada a estes protótipos implica um investimento elevado, pelo que até ao momento ainda não houve produção comercial de nenhum produto. A existência de produtos próprios é, no entanto, um dos objectivos estratégicos da empresa no médio prazo [15].

São exemplos dos produtos desenvolvidos pela empresa o We Alert e o We Find.

O We Alert é um sistema de alerta de afastamento e prevenção de afogamento para crianças e cujo protótipo electrónico se apresenta na Figura 3.

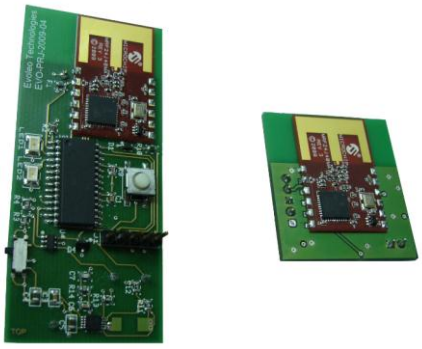


Figura 3 - Cartas electrónicas do We Alert produzidas na Evoleo.

O We Alert é um produto que tem por objectivo evitar que as crianças se afastem dos cuidadores - família, responsáveis. O equipamento alerta o responsável do afastamento, quando a criança se afasta mais do que uma distância de segurança de cerca de 30 metros.

Para além desta função, o We Alert alerta o responsável para possíveis situações de afogamento eminente.

O sistema We Find é um produto que permite a obtenção de uma localização exacta de pessoas, viaturas, transportes, entre outros, através do sistema de posicionamento global (*Global Positioning System* - GPS). A Figura 4 apresenta o módulo desenvolvido.

Figura 4 - Pormenor do módulo We Find.

O We Find é um localizador GPS compacto que permite a obtenção de uma localização através do envio de uma mensagem para o número de telemóvel associado. O equipamento responde à mensagem enviada da mesma forma, isto é, enviando uma mensagem com a localização (latitude e longitude) para um número que solicitou esta localização.

1.3. MOTIVAÇÃO E ENQUADRAMENTO DO TRABALHO A DESENVOLVER

Tendo em conta o actual estado da tecnologia e a miniaturização de diversos componentes, abrem-se novas perspectivas para utilização dos sistemas modernos para resolução de problemas actuais. O trabalho desenvolvido pela Evoleo nos últimos meses vem abordar o assunto relacionado com a eficiência energética e a sustentabilidade. De uma forma sucinta, o objectivo do projecto consiste em desenvolver uma infra-estrutura que permita monitorizar eventos, devendo ser a interface de fácil utilização. Aplicações possíveis desta solução seriam para o sector industrial. A determinação de consumos dos motores através dos módulos medidores de energia, detecção de vibrações que indiquem anomalias nos motores, através do módulo de análise de vibrações, monitorização de níveis de depósitos em Estações de Tratamento de Águas Residuais (ETAR) por exemplo, de abertura e fecho de válvulas, através dos módulos de aquisição múltipla. Genericamente trata-se de uma solução que se pretende completa de *hardware* e *software* que permita a monitorização de uma instalação mesmo que complexa.

Aquando do início do estágio na Evoleo em 19 de Março de 2012, a equipa estava a dar os primeiros passos no desenvolvimento de um equipamento de aquisição múltipla de dados analógicos e digitais, estando ainda numa fase conceptual. Estando a equipa já orientada com o desenho do *hardware* e *firmware* para o equipamento, foi dada prioridade ao desenvolvimento de um *software* de visualização dos dados recolhidos. Seria este o passo e rumo inicial do estágio. Com o desenrolar da fase de investigação durante a elaboração da proposta inicial para solução do problema, os níveis de exigência do projecto subiram e o trabalho deixaria apenas de focar na visualização de alguns dados através de um *software* dedicado e específico, passando agora a ser uma plataforma de recolha e visualização de dados sensoriais obtidos do módulo já em fase de projecto, bem como de equipamentos futuros que viessem mais tarde a ser desenvolvidos.

Estimulado pelas diversas redes sensoriais que começam agora a surgir, colocou-se de parte a ideia de um *software* para um equipamento, e tomou-se a decisão de enveredar por um portal agregador de dados sensoriais, tendo por base os equipamentos que até agora se têm vindo a desenvolver para esta solução. O conceito fundamental e a arquitectura do sistema foram determinados tendo por base a simplicidade/facilidade de acesso, interactividade, atractividade visual e controlo da instalação dos sensores, fornecendo ao mesmo tempo a capacidade de expansão futura a novos equipamentos. Em conjunto com a equipa da Evoleo, esta plataforma foi-se adaptando às necessidades que foram surgindo, e aos equipamentos que foram sendo adicionados. Adicionalmente foram incluídas potencialidades de depuração por se revelarem essenciais durante as fases de teste e calibração dos equipamentos em desenvolvimento, numa interacção próxima com os responsáveis do desenvolvimento de *hardware* e *firmware* da equipa.

Com este trabalho desenvolvido, a Evoleo Technologies passa a dispor de uma solução completa de *hardware* e *software* independente e funcional, pronta a ser instalada, algo que sem o *software* desenvolvido não seria possível. A plataforma passa assim a ser uma forma de agregar novos produtos desenvolvidos na Evoleo, sob uma interface simples e comum a todos os equipamentos e que oferece ao utilizador final a possibilidade de monitorização e reconfiguração remota dos equipamentos, a partir de qualquer PC, *smartphone* ou *tablet* com um *browser* de Internet.

1.4. CALENDARIZAÇÃO

Todo o processo de investigação e desenvolvimento da plataforma foi documentado e controlado durante os oito meses de estágio. Durante este período foram desenvolvidos vários documentos intercalares de estudo e investigação, apresentados aos orientadores na empresa, por forma a expor os avanços efectuados e novas propostas e ideias que iam surgindo de ambas as partes no decorrer do projecto. Grande parte das tarefas propostas e realizadas durante o estágio, bem como as diferentes etapas e opções de projecto, estão presentes num *log book* com cerca de 150 páginas, onde se definiu diariamente metas a atingir e metas atingidas no final de cada dia. Desta forma obteve-se um controlo sobre as funcionalidades implementadas e as propostas, uma metodologia de trabalho diária por objectivos incrementais sugeridos ou idealizados, e uma base sólida de informação sobre o desenrolar do estágio, que permitiria um pormenor consideravelmente maior no esquema apresentado na Figura 5, mas que se optou por manter algo simplificado.

	Task Name	Duration	Start	Finish
1	Análise Conceptual	10 days	19-03-12	01-04-12
2	Análise de Requisitos	15 days	02-04-12	20-04-12
3	Desenho Preliminar	21 days	02-04-12	30-04-12
4	Testes iniciais - Javascript, PHP, MySQL, bibliotecas	26 days	09-04-12	14-05-12
5	Desenvolvimento do Core v1	25 days	23-04-12	25-05-12
6	Páginas de teste conceptuais com hardware deenvolvido	19 days	15-05-12	08-06-12
7	Restruturação completa do design e navegação do portal	14 days	08-06-12	27-06-12
8	Desenvolvimento da página de Reconfiguração remota	21 days	27-06-12	25-07-12
9	Desenvolvimento do Core v2 com suporte ao novo módulo Energ	30 days	15-06-12	26-07-12
10	Testes de validação reconfiguração	9 days	26-07-12	07-08-12
11	Desenvolvimento página de visualização de gráficos	4 days	07-08-12	10-08-12
12	Férias	15 days	13-08-12	31-08-12
13	Desenvolvimento página de visualização de gráficos (continuaçã	10 days	03-09-12	14-09-12
14	Desenvolvimento página de visualização de grupos	14 days	13-09-12	02-10-12
15	Preparação para a conferência EES 2012	6 days	21-09-12	28-09-12
16	Testes de validação do portal	25 days	03-09-12	06-10-12
17	Elaboração do Relatório	72 days	10-07-12	17-10-12

Figura 5 - Calendarização do trabalho.

1.5. ORGANIZAÇÃO DO RELATÓRIO

No capítulo um, capítulo introdutório, é feita uma apresentação e contextualização do assunto deste trabalho. É apresentado também o problema existente e os objectivos do trabalho.

No capítulo dois, são introduzidos alguns conceitos relevantes para o desenvolvimento do trabalho, abordando áreas como as ferramentas utilizadas para a programação *web*, infra-estruturas possíveis para a comunicação das redes de sensores e configurações possíveis, e outros conceitos importantes, ainda que apresentados de forma sucinta.

O capítulo três apresenta o estado da tecnologia para as redes de sensores sem fio, e projectos que se encontram em desenvolvimento que partilham semelhanças com o proposto. São abordados os principais projectos actuais e analisadas as suas características e funcionamento geral, sendo essencialmente o capítulo de estudo desenvolvido durante o estágio por forma a avaliar as opções existentes.

No capítulo quatro é apresentada a solução proposta à empresa, onde se procura apresentar o funcionamento do sistema em alto nível. Os requisitos definidos inicialmente durante o estágio, são também aqui apresentados.

No capítulo cinco detalha-se a solução desenvolvida, tanto quanto possível, por questões de confidencialidade relacionadas com a empresa. Neste capítulo, apresentam-se as principais funcionalidades da plataforma, de uma forma detalhada, explicando o funcionamento de todo o sistema por detrás da interface gráfica.

No capítulo seis, apresenta-se a validação do trabalho efectuado, através de vários testes realizados durante o estágio. Adicionalmente, apresentam-se alguns comentários realizados pelos participantes da conferência *Energy, Environment and Sustainability 2012* (EES 2012) sobre o trabalho desenvolvido.

No capítulo sete são apresentadas conclusões sobre o trabalho desenvolvido, e direcções de desenvolvimento futuro.

2. CONCEITOS FUNDAMENTAIS

Neste capítulo apresentam-se alguns conceitos fundamentais relativos ao trabalho, importantes para a compreensão das opções tomadas no desenrolar do projecto.

São vários os conceitos abordados ao longo deste trabalho, uma vez que se trata de toda a estrutura da solução, desde o *hardware* até à interface com o utilizador. Neste capítulo são apresentados alguns dos conceitos mais relevantes, relacionados com a visão do projecto e a própria implementação do trabalho desenvolvido.

2.1. REDES DE SENSORES SEM FIO

As redes de sensores sem fio (*Wireless Sensor Network* - WSN) consistem essencialmente em redes sem fios de dispositivos autónomos distribuídos por uma determinada área, que utilizam sensores para monitorizar condições físicas ou ambientais. Estes dispositivos autónomos ligam-se a *routers* e *gateways* para criar o sistema típico de uma rede WSN. Os dispositivos autónomos distribuídos numa determinada zona comunicam com uma *gateway* central, que serve de interligação entre a rede sem fios e a rede com fios,

disponibilizando assim os dados para serem recolhidos, analisados, processados e apresentados ao utilizador do sistema [17].

As WSN são constituídas por *nós* ligados a um ou mais sensores. Cada *nó* da rede é tipicamente constituído por um *transceiver* por radiofrequência, com antena interna ou conexão para uma antena externa, um microcontrolador, um circuito electrónico de condicionamento de sinal para interface do microcontrolador com os sensores e uma fonte de alimentação [18] (e.g. bateria, painel solar), conforme ilustrado na Figura 6.

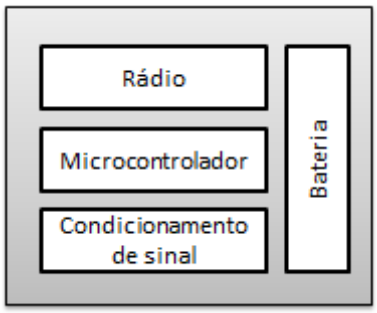


Figura 6 - Representação de um *nó* de uma rede WSN.

O tamanho destes *nós* varia consoante a aplicação e requisitos definidos. O mesmo se aplica para o preço destes *nós*. As topologias destas redes WSN podem variar desde uma rede em estrela até uma rede *multi-hop* [19][20].

2.1.1. APLICAÇÕES

As aplicações práticas da monitorização sem fios são muito variadas conforme apresentado na Figura 7. Estas podem ser justificadas em situações em que, por motivos de limitações da infra-estrutura, ou questões relacionadas com custos, uma implementação com fios não seria viável [20]. São exemplos disso aplicações nas seguintes áreas:

- Agricultura;
- Rede eléctrica;
- Monitorização de estruturas;
- Monitorização na indústria;
- Monitorização ambiental;

Figura 7 - Exemplos de aplicação de redes WSN [20].

Na agricultura, um sistema WSN torna-se particularmente interessante para aplicações como monitorização dos campos de cultivo, nas quais os requisitos impõem uma solução distribuída, para adquirir informações relacionadas com a utilização, tais como a humidade, temperatura atmosférica ou mesmo do solo do cultivo [21]. Os sistemas de rega por efeito de gravidade podem ser monitorizados usando sensores de pressão para determinar os níveis dos tanques de água. A rega automática permite também uma utilização mais eficiente do uso da água, reduzindo o desperdício. As bombas de água podem ser controladas utilizando entradas e saídas do dispositivo sem fios e o consumo de água pode ser medido e transmitido para uma central de controlo que posteriormente faça a gestão de custos [22].

Na rede eléctrica, as WSN proporcionam um sistema de baixo custo para recolha de dados que permitem a redução dos gastos de energia e uma melhor utilização dos recursos, nomeadamente, iluminação das ruas e gastos energéticos em edifícios públicos [22] [23]. As WSN permitem fazer uma monitorização constante dos consumos em energia de um determinado edifício, e podem gerar alarmes ou avisos caso se detecte algum valor fora do esperado numa certa zona do edifício, avisando o utilizador por correio electrónico ou mensagem SMS (*Short Message Service*) do sucedido [26].

Na monitorização de estruturas, as redes de sensores podem ser utilizadas para monitorização constante da conservação estrutural de edifícios, pontes, túneis, etc. [27]. Os sensores sem fios podem ser utilizados para monitorizar movimentos nestas estruturas, fornecendo aos engenheiros dados constantes de análise da mesma, sem necessidade de

deslocação ao local (custos) e com uma recolha de dados constante (várias medições diárias) face às medições semanais ou mensais que poderiam ocorrer.

Na monitorização na indústria, as redes WSN podem também ser utilizadas para determinar o estado das máquinas e monitorização dos processos [28]. As redes de sensores permitem obter informação sobre o possível desgaste das máquinas. A instalação de *nós* sem fios permite ainda aceder a partes antes dificilmente acessíveis ou mesmo inacessíveis, com sensores com fio.

Na monitorização ambiental, as redes de sensores têm vindo a ser utilizadas para monitorizar uma grande variedade de situações, tais como a análise da qualidade do ar, a detecção de fogos florestais, a detecção de deslizamentos de terra, a monitorização da qualidade da água e a prevenção de desastres naturais [24][25][29]. Um sistema de detecção de deslizamentos de terra pode utilizar uma rede de sensores sem fios para detectar movimentos no solo e mudanças em diversos parâmetros que podem ocorrer antes ou durante um deslizamento de terra [29]. Adicionalmente, ao ser recolhida informação sobre estes eventos e posteriormente analisada, pode eventualmente ajudar na detecção futura dos mesmos. Existem também algumas redes WSN para detecção de actividade sísmica ou erupção perto de vulcões [25]. Um grupo de investigadores da Universidade de Harvard implementou três redes de sensores sem fios em torno de três vulcões activos, conforme apresentado na Figura 8, para recolha de infra-sons que possam fornecer indícios de iminente actividade sísmica ou vulcânica [29][30].

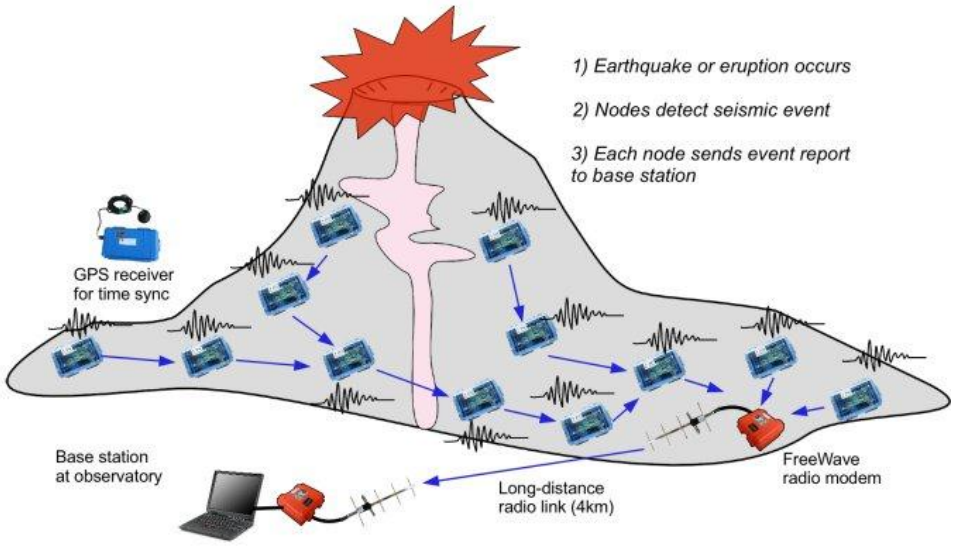


Figura 8 - WSN de *Harvard Sensors LAB* de detecção de actividade sísmica [30].

Em Londres e Estocolmo, já foram implementadas várias redes WSN para monitorizar as concentrações de gases nocivos para a população [31]. Nestes casos de ambientes urbanos, as redes WSN podem já tomar partido dos vários pontos de acesso Wi-Fi disponíveis na área para realizar o *upload* dos dados recolhidos para a Internet.

A utilização destes tipos de redes pode também ser aproveitada para detecção de incêndios nas florestas [32]. As unidades distribuídas podem estar equipadas com sensores que permitam a medição da temperatura, humidade e gases produzidos pelos fogos nas árvores ou vegetação. A detecção precoce destes incêndios é crucial para potenciar o sucesso da intervenção por parte dos bombeiros.

2.1.2. ARQUITECTURA DE UMA REDE WSN

Numa arquitectura WSN genérica, os *nós* de medição são distribuídos espacialmente para adquirir medições relacionadas com temperatura, tensão, ou oxigénio dissolvido, por exemplo [23]. Os *nós* fazem parte de uma rede *wireless* administrada por uma *gateway*, que controla determinados aspectos, tais como os relacionados com autenticação do cliente e segurança dos pacotes. A *gateway* recolhe a informação dos vários sensores e envia-os através de uma ligação com fios ou sem fios para um computador que irá disponibilizar a informação recolhida [22]. Neste computador, o *software* em execução pode analisar e processar os dados fornecidos e apresentar os dados da forma pretendida. A Figura 9 apresenta um diagrama de blocos representativo deste processo.

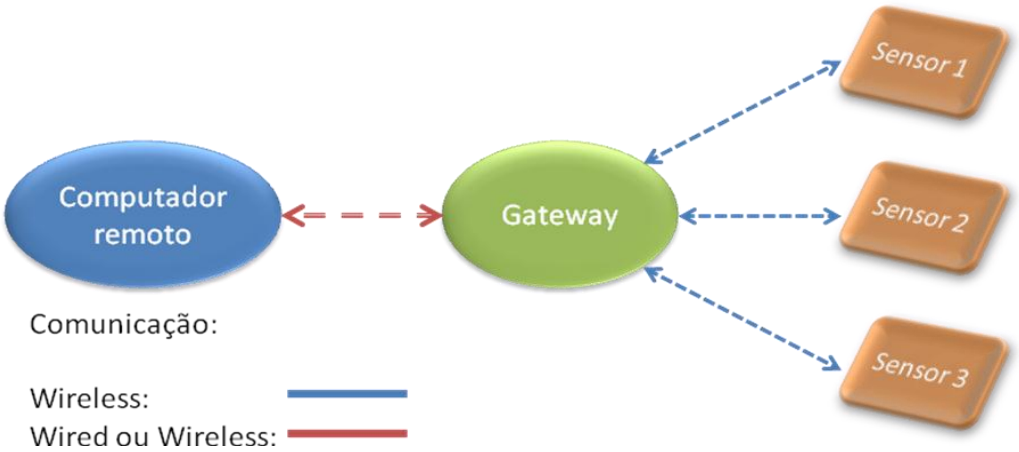


Figura 9 - Arquitectura de uma WSN e interligação com o PC.

Alternativamente à configuração disposta na Figura 9, a informação recolhida pela *gateway* pode ser armazenada num servidor ao qual o computador remoto acede para monitorizar os dados recolhidos. Esta configuração é representada na Figura 10.

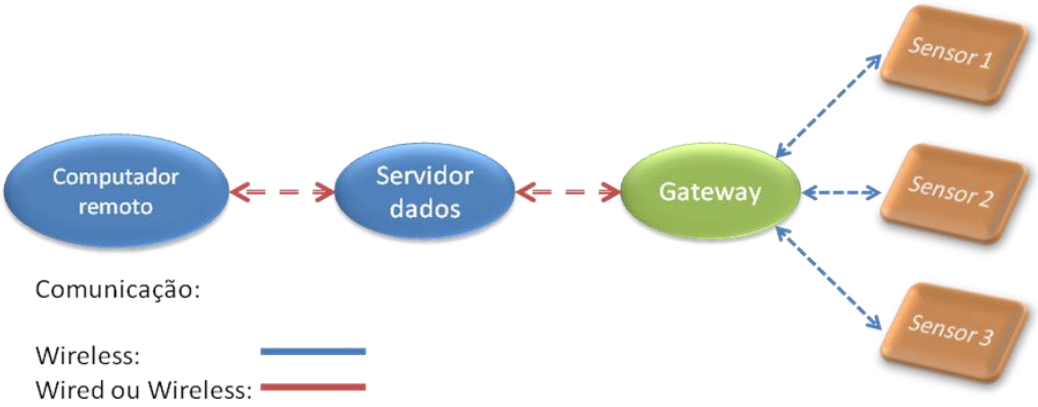


Figura 10 - Arquitectura alternativa.

Neste caso, a informação passa a ser disponibilizada pelo servidor para qualquer cliente remoto que pretenda monitorizar os dados recolhidos.

2.1.3. TOPOLOGIAS DE REDES WSN

Várias topologias de rede podem ser utilizadas para coordenar a *gateway* da rede WSN, *nós* e *nós routers* da rede. Os *nós routers* são semelhantes aos *nós sensores* dado que podem também recolher informação dos sensores próprios mas adicionalmente podem também ser utilizados para passar os dados recolhidos para outros *nós*. Os três tipos de topologia de redes WSN aqui descritos apresentam-se sob a forma de esquema na Figura 11 [19][23].

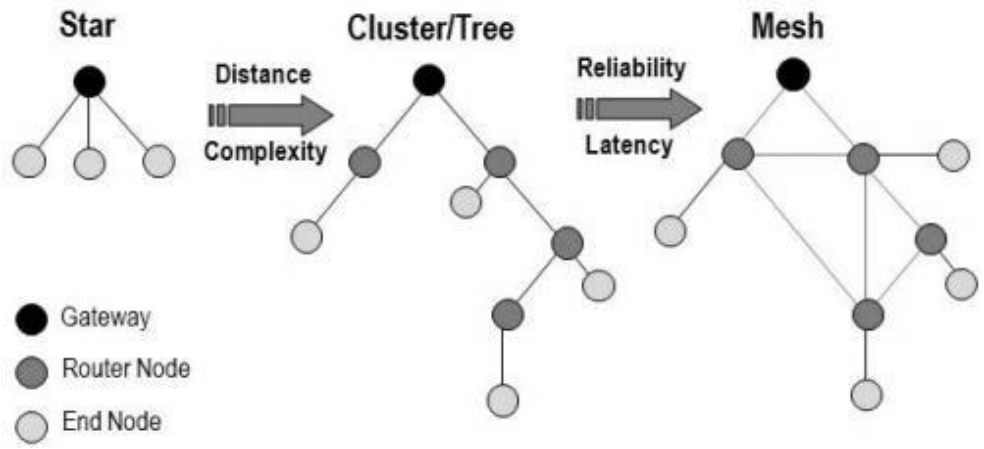


Figura 11 - Topologias de redes WSN [19].

A primeira topologia e a mais simples é a topologia em estrela, na qual cada *nó* mantém uma única comunicação directa com a *gateway*. Esta topologia tem a vantagem de ser simples mas restringe a área máxima total que a rede pode cobrir.

De forma a aumentar a área que a rede pode cobrir, pode-se implementar uma topologia em árvore. Nestes tipos de arquitectura mais complexa, cada *nó* mantém mesmo assim um único canal de comunicação para a *gateway*, mas pode usar outros *nós* para redireccionar os dados através desse caminho. A desvantagem desta topologia é que, caso um dos *routers* avarie, todos os *nós* ligados a este *router* ficam sem poder comunicar com a *gateway*.

Uma rede WSN com topologia em *mesh* resolve esta limitação ao utilizar caminhos de comunicação redundantes de forma a aumentar a confiabilidade do sistema. Numa rede *mesh*, os *nós* mantêm múltiplos caminhos de comunicação com a *gateway*, para que caso um *nó router* deixe de funcionar, a rede automaticamente redireccione os pacotes por uma rota diferente. A topologia *mesh* oferece mais garantias de entrega dos pacotes mas sofre de maior latência uma vez que os pacotes têm de percorrer vários *nós router* até chegar à *gateway* [19].

A escolha da topologia de rede para cada aplicação depende assim da área, complexidade e confiabilidade necessárias para cada caso.

2.1.4. PRINCIPAIS CARACTERÍSTICAS

As principais características dos elementos constituintes de uma infra-estrutura WSN são as seguintes [23]:

- Limitações nos consumos de energia;
- Fácil transporte e remoção/instalação;
- Tem de gerir falhas na comunicação;
- Elevada autonomia;
- Facilidade de utilização;

A limitação no consumo energético de cada elemento justifica-se pela fonte de energia associada, isto é, baterias ou mesmo captação de energia no local (e.g. painéis solares).

A facilidade de transporte de cada elemento torna-se relevante para a reutilização ou relocalização deste na instalação.

A gestão de falhas de comunicação deve também ser avaliada, verificando métodos alternativos caso a transmissão de dados não seja concretizada.

A autonomia de cada módulo é também relevante uma vez que podem ser instalados em locais sem rede eléctrica, como no caso das redes de sensores que monitorizam variáveis ambientais distribuídas por campos de cultivo, ou mesmo módulos que podem ser instalados no interior de motores para adquirir dados referentes a temperatura ou vibrações, onde o acesso pelo exterior se torna difícil.

A facilidade de utilização de cada elemento deve também ser tida em conta, nomeadamente na manutenção do módulo e mesmo na parametrização e visualização dos dados recolhidos por este. A interface *Homem-Máquina* deve permitir consultar todos os dados recolhidos sem necessidade de conhecimentos técnicos específicos.

2.2. DESENVOLVIMENTO *WEB*

O desenvolvimento de aplicações *web* envolve vários conceitos e ferramentas que se abordam nesta secção. Pode-se dividir o desenvolvimento *web* em duas partes, i.e. o desenvolvimento *client-side*, e o desenvolvimento *server-side* [33]. O primeiro é executado do lado do cliente após o acesso à página e o segundo, que não é visível ao cliente, é executado no servidor e só apresentado o resultado da operação. Cada um é descrito de seguida de modo mais detalhado.

2.2.1. DESENVOLVIMENTO *CLIENT-SIDE*

O desenvolvimento *client-side* oferece quer *design* como funcionalidade à página. O *design* da página deve ser apelativo e simples, tendo contudo todas as funcionalidades necessárias definidas no requisito do projecto.

O aspecto visual da página é realizado essencialmente através de *HyperText Markup Language* (HTML) e as folhas de estilo (*Cascading Style Sheet* – CSS). O HTML oferecerá a estrutura da página e o CSS o aspecto gráfico (cores, gradientes de cor, sombras, etc.)[33][34].

HyperText Markup Language (HTML)

O HTML, que pode ser traduzido por Linguagem de Marcação de Hipertexto é uma linguagem utilizada para o desenvolvimento de páginas *Web*. Essencialmente, esta linguagem pode ser vista como um conjunto de etiquetas (*tags*) que servem para definir a forma na qual se apresentará o texto e outros elementos na página [33].

Os documentos HTML são ficheiros de texto simples que podem ser criados a partir de qualquer editor de texto. Existem adicionalmente *software*'s de edição de páginas baseados no conceito *What You See Is What You Get* (WYSIWYG), como é o caso do *Dreamweaver*, onde o código HTML é gerado dinamicamente, de acordo com as alterações efectuadas ao *design* da página [34].

Estrutura básica de um Documento HTML

A estrutura básica de uma página *web* consiste num conjunto de instâncias semelhante ao exemplo apresentado na Figura 12. Conforme se pode observar, temos a inclusão das folhas de estilo, um *script JavaScript* para criar um pequeno alerta e a estrutura típica da disposição do documento.

```
<!DOCTYPE html>
<html>
    <head>
        <title> Título do documento <title>
        <!-- Inclusão do ficheiro CSS -->
        <link rel="stylesheet" type="text/css" href="class.css" />
    </head>
    <body>

    <h1>Heading da página</h1>
    <p>Parágrafo da página</p>

    </body>
    <script language="javascript" type="text/javascript">
        <!-- Ao abrir a página, cria popup com a mensagem olá mundo -->
        alert("Olá Mundo");
    </script>
</html>
```

Figura 12 - Estrutura básica do código HTML.

As *tags* <html> e </html> representam o inicio e fim do código HTML respectivamente. O preenchimento das *tags* não é *Case Sensitive*. Todas estas *tags* são compreendidas como código HTML e apresentadas na página de acordo com as suas características predefinidas.

Como foi dito, as páginas *Web* são geralmente constituídas por um título principal, que é colocado no cabeçalho do código HTML entre as *tags* <head> e </head> dentro da *tag* <title>, como mostra a Figura 12. De seguida temos geralmente o corpo da página, onde aparece todo o conteúdo do sítio que nos encontramos, habitualmente constituído por texto, imagens, hiperligações, entre outros. O corpo da página é colocado entre as *tags* <body> e </body> pode conter outras *tags* para estilizar o texto da página, apresentar imagens, hiperligações, etc..

***Cascading Style Sheets* (CSS)**

O CSS consiste essencialmente numa linguagem de estilo desenvolvida para separar a formatação do documento, da própria estrutura do documento e é responsável por definir a apresentação de cada elemento da página [33]. A sintaxe é relativamente simples dado que se trata de um conjunto de palavras que representam propriedades às quais se pode atribuir um valor, associado a um selector que identifica a que elemento da página se aplicam essas novas regras. A estrutura do CSS pode-se representar da seguinte forma:

```
#selector1 {       //ID do elemento a formatar

  Propriedade 1: valor 1;     //Atribuição de
  Propriedade 2: valor 2;     //novos parâmetros
  Propriedade 3: valor 3;     //ao elemento

}
```

Se for necessário atribuir um conjunto de estilos ao elemento identificado por *#selector1*, basta então indicar qual a propriedade que se pretende alterar e o valor pretendido.

O seguinte exemplo representa a atribuição de vários estilos à classe *teste_classe*.

```
.teste_classe {
    text-decoration: underline;    //Sublinhado
    color: #00FF00;   //Definição da cor da letra
    font-family: "courier";   //Tipo Letra
    font-size: 14pt;          //Tamanho da letra
    font-weight: heavy;       //Negrito
}
```

A definição dos estilos de cada elemento é geralmente feita num documento à parte do que contêm o código HTML, sendo posteriormente incluído na página.

JavaScript

O *Javascript* é uma linguagem que oferece a funcionalidade à página do ponto de vista do cliente. É assim possível realizar transições, animações, resultados, validações e mesmo interactividade com a página para além do tradicional *submit* do formulário e obter a resposta do servidor noutra página [33]. É actualmente a linguagem mais usada para programação *client-side* em navegadores *web* [35]. A sintaxe é muito semelhante a C/C++ e possui várias funções e métodos de alto nível geralmente disponíveis em linguagens como o Java ou o C++, facilitando assim a tarefa do programador [33].

A vantagem do *Javascript* em relação a outras linguagens com potencialidades semelhantes como o Flash é fundamentalmente a compatibilidade com navegadores actuais. O Flash está a ser abandonado por grande parte da comunidade de programadores (Refira-se que a Apple recusa-se a suportar Flash no iOS, e Android apesar de já ter suportado, começa também a abandonar o suporte a partir da versão 4.1).

Uma vez que o código é executado do lado do cliente, a execução torna-se mais rápida, dependendo só do processamento do computador do cliente. Pode-se assim criar uma aplicação *web*, interactiva, com efeitos visuais simples e com toda a funcionalidade que se pretenda adicionar.

O código seguinte trata-se de um pequeno exemplo da sintaxe *Javascript*:

```
<script type=”text/javascript”>

  //Declaração de função somar
  var somar = function() {

    //Declaração de variáveis para cálculo
    var i, x = 0;

    //ciclo for para percorrer e somar valores
    //passados por parâmetro
    for (i = 0; i < arguments.length; i++) {
        x  = x + arguments[i];
    }

    //Retorno do valor da somar realizada
    return x;
  }


  //Invocação da função de soma e apresentação do
```

```
  //valor num popup de alerta no ecrã
  alert( somar(5, 13, 10) ); // retorna 28

</script>
```

O código apresentado trata-se de uma função de soma de valores passados por parâmetro, que recorre a um ciclo *for* para efectuar a soma dos valores. A função é chamada ao abrir a página pela invocação da função *alert(),* que retorna então para uma mensagem de *popup* no ecrã o valor da soma.

***Asynchronous JavaScript And XML* (AJAX)**

O *AJAX* é essencialmente um método de enviar e receber dados para um servidor, sem necessidade de efectuar o *refresh* da página. Apesar do nome, o XML não é obrigatório como codificação dos dados enviados e recebidos (JSON também é frequentemente utilizado) nem o pedido tem de ser assíncrono [36].

A popularidade do AJAX surgiu em 2005 quando a Google utilizou AJAX na caixa de texto de pesquisa, para retornar sugestões do que se estava a escrever conforme se iam inserindo novos caracteres. Não se tratava de uma nova tecnologia, mas sim uma nova forma de utilizar a tecnologia existente. Com efeito, os *browsers web* apresentavam esta tecnologia desde o ano 2000.

Algumas características do AJAX são as seguintes:

- Permite carregar uma página com informação mais rapidamente e apresentá-la aos utilizadores com menos tempo de carregamento;
- Baseado em *JavaScript* e pedidos HTTP;
- Troca de informação utilizando JSON ou XML;
- Alteração dinâmica da estrutura da página com base nos novos dados recebidos;
- Recuperação assíncrona de dados utilizando o objecto *XMLHttpRequest* e *XMLHttpResponse*;

2.2.2. DESENVOLVIMENTO *SERVER-SIDE*

O desenvolvimento *server-side* irá permitir a interface entre dados do servidor e o cliente. O código executado no servidor permitirá gerar a página dinamicamente, sendo possível gerar o código HTML, *Javascript* e CSS, fornecendo-o ao cliente para apresentação da

página. São várias as linguagens de programação disponíveis para o desenvolvimento *server-side* tais como:

- ASP;
- PHP;
- PERL;
- Python;
- Ruby;
- Java;
- SMX;
- .NET;
- Lua;
- WebDNA;

A Figura 13 representa a interacção entre cliente e servidor. O pedido efectuado pelo cliente é processado pelo interpretador PHP. Dependendo do pedido, o *script* pode aceder à base de dados para obter informação que lhe permita criar a resposta para o cliente.

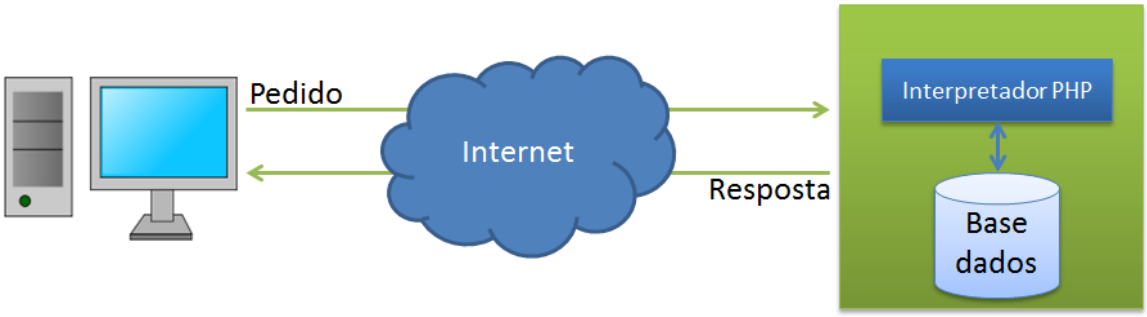


Figura 13 - Representação da interacção cliente-servidor.

Entre as tarefas mais comuns, realizadas pelos *scripts* executados *server-side* podemos identificar o acesso à base de dados e leitura ou escrita de dados, a validação dos pedidos efectuados pelos clientes, apresentação de respostas de erro se necessário e assegurar a confidencialidade da informação existente na base de dados.

2.2.3. SQL

A linguagem utilizada para a criação e manipulação de bases de dados mais utilizada actualmente é o *Strutured Query Language* (SQL), que em língua portuguesa se pode traduzir por Linguagem de Consulta Estruturada. Esta linguagem diferencia-se de outras linguagens de consulta a bases de dados, uma vez que uma consulta especifica a forma do resultado e não o caminho [37].

Por exemplo, considere-se a situação representada na Figura 14, isto é, uma tabela presente numa base de dados fictícia, com duas colunas, *C1* e *C2*, e dois registos.

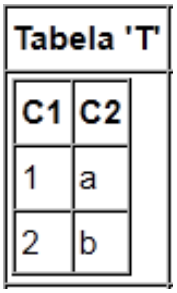


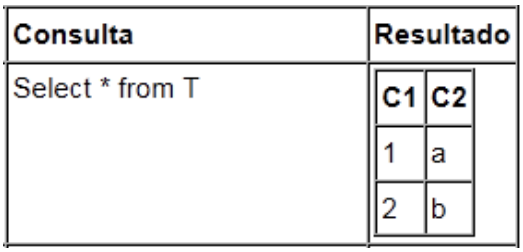
Tabela 'T'	
C1	**C2**
1	a
2	b

Figura 14 - Tabela exemplificativa *T* [38].

Quando se efectua a pesquisa com o comando

SELECT * FROM T

temos como resultado todos os elementos de todas as linhas da tabela chamada T, como mostra a Figura 15.



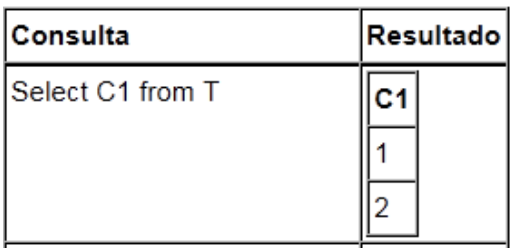
Consulta	Resultado	
Select * from T	**C1**	**C2**
	1	a
	2	b

Figura 15 - Resultado da consulta *SELECT * FROM T* [38].

Repare-se no entanto que ao realizar a pesquisa utilizando o comando,

SELECT C1 FROM T

o resultado já será apenas o que se ilustra na Figura 16.



Consulta	Resultado
Select C1 from T	**C1**
	1
	2

Figura 16 - Resultado da consulta *SELECT C1 FROM T* [38].

Considere-se agora a seguinte pesquisa a que corresponde o comando seguinte:

SELECT * FROM T WHERE C1=1

O resultado desta pesquisa será o apresentado na Figura 17

Consulta	Resultado
Select * from T where C1=1	C1 C2 1 a

Figura 17 - Resultado da consulta SELECT * FROM T WHERE C1=1 [38].

Do exposto pode-se então compreender, como foi referido anteriormente, que uma consulta em SQL vai de encontro ao resultado que se pretende, uma vez que dizemos apenas o que pretendemos do resultado da pesquisa, e no caso de este existir é exibido.

De um modo geral, para fazer a gestão das Bases de Dados, existem sistemas próprios conhecidos como Sistemas de Gestão de Bases de Dados (SGBD), cuja explicação será feita na subsecção seguinte.

2.2.4. SISTEMAS DE GESTÃO DE BASES DE DADOS (SGBD)

Um Sistema de Gestão de Base de Dados trata-se essencialmente de um pacote de *software* que permite a criação, manutenção e utilização de uma base de dados. Um sistema deste tipo permite que várias aplicações acedam a base de dados concorrentemente sem perda de informação. As linguagens de programação de bases de dados também simplificam a respectiva organização, bem como recolher e apresentar a informação obtida. Assim, um SGBD possibilita o controlo do acesso aos dados, reforça a integridade dos dados, faz a gestão do acesso à informação concorrente, permite a recuperação da base de dados depois de falhas e o seu restauro a partir de ficheiros de *backup* e também mantêm a segurança da base de dados.

Para permitir aos utilizadores atingir objectivos como consultar dados, inserir novos dados, entre outros, um SGBD disponibiliza linguagens de:

- **Definição de dados:** para criação e alteração da estrutura da Base de Dados (*Data Definition Language* - DDL) [39];
- **Manipulação de dados:** para acrescentar dados novos e modificar dados existentes (*Data Manipulation Language* - DML) [39].
- **Consulta de dados:** que permite obter e processar os dados armazenados (*Data Query Language* - DQL) [39];

Actualmente os SGBD têm, na maioria dos casos, um sistema um interface com o utilizador que esconde estas linguagens (DDL, DML, DQL) atrás de botões activados com um simples *click*, como por exemplo *Consultar*, *Inserir*, *Apagar*, entre outros. Um desses exemplos é o PHPMyAdmin para o SGBD MySQL [37].

2.3. ANDROID

A *Open Handset Alliance* (OHA) foi criada com intuito de desenvolver uma plataforma *open-source* para telemóveis que visasse preencher algumas lacunas presentes neste sector do mercado. Inicialmente teve por base uma aliança sólida de empresas líderes nos seus respectivos mercados como TI, Google, Nvidia, SE, Vodafone, num total de 79 companhias de *hardware*, *software* e telecomunicações [40].

Com efeito, mais de metade da população mundial possui actualmente um telemóvel [41] e procuram para além das funcionalidades tradicionais de envio de SMS ou realização de chamadas, outros recursos tais como câmara, jogos, acesso a Internet, GPS entre uma miríade de outras funcionalidades. Para além da utilização comum, existem também casos de empresas que tentam agilizar os seus processos, recorrendo a aplicações móveis para tal efeito. Nesse sentido, o Android oferece assim um sistema operativo robusto rodeado de ferramentas de desenvolvimento de aplicações baseadas nas linguagens JAVA e XML, que são totalmente *open-source* e que permite aos desenvolvedores de *software* uma rápida adaptação ao ambiente de desenvolvimento [42].

2.3.1. A PLATAFORMA ANDROID

Em Novembro de 2007, em simultâneo com o anúncio da formação da OHA, foi publicada a primeira distribuição Android, tendo sido disponibilizado o primeiro *kit* de desenvolvimento dias depois. O primeiro aparelho surgiu em Outubro de 2008, pela HTC, o G1, com a versão do Android 1.0. Desde então, surgiram várias versões deste Sistema Operativo (SO), desde o 1.0/1.1 (*Alfa/Beta*), seguindo uma ordem de nomenclatura peculiar: 1.5 *Cupcake*, 1.6 *Donut*, 2.0/2.1 *Eclair*, 2.2 *Froyo*, 2.3 *Gingerbread*, 4.0 *Ice-Cream Sandwich* até à mais recente, 4.1 *Jelly Bean*.

Actualmente a plataforma Android usa como base o *kernel* do Linux 3.1.10, tendo sido efectuadas várias alterações para que o sistema fosse adaptado aos dispositivos móveis. É responsável pelas tarefas fundamentais do sistema tais como segurança e gestão de memória.

A pilha de *software* consiste em aplicações Java que trabalham através das bibliotecas *core* de Java numa máquina virtual *Dalvik* com compilação JIT (*Just-in-Time*). As bibliotecas escritas em linguagem C incluem o gestor do ambiente gráfico, sistema gestão de base de dados SQLite, OpenGL para gráficos 3D, SGL para 2D, etc., totalizando em 12 milhões de linhas de código do SO.

Arquitectura

A arquitectura do sistema operativo Android é dividida em cinco partes fundamentais [42] conforme se ilustra na Figura 18.

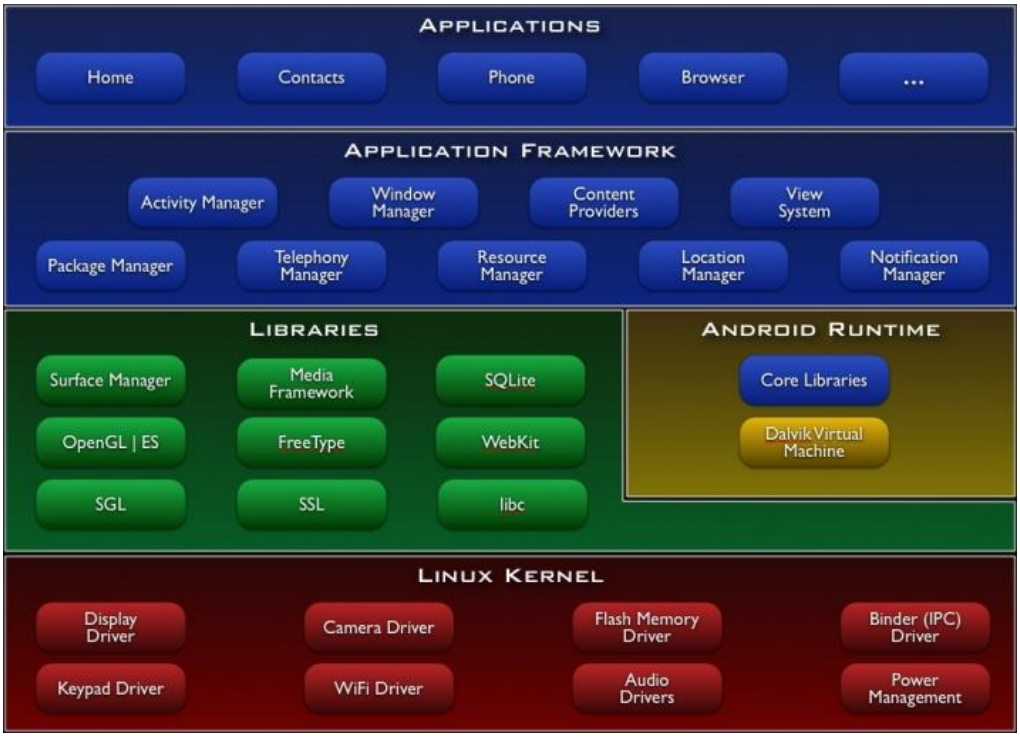


Figura 18 - Arquitectura do Sistema operativo Android [43].

Cada bloco colorido da Figura 18 será detalhado nas subsecções seguintes.

Kernel Linux

A camada do *Kernel* Linux é baseada no Linux 3.1. É responsável por tarefas fundamentais como gestão de processos, memória, pilha protocolar de rede, segurança, protocolos de comunicação, etc. conforme se ilustra na Figura 19.

Figura 19 - *Kernel* Linux [43].

O *kernel* serve também como uma camada de abstracção do *hardware* para as camadas superiores. De forma a compatibilizar o SO com o novo ambiente em que este se integra, foi necessário realizar algumas alterações ao *kernel*, nomeadamente adição de *device drivers* (e.g. rádios GSM, HSDPA, GPS), melhoramentos a nível de gestão de energia (*wake locks*) e um sistema que permite terminar processos de uma forma criteriosa quando há pouca memória disponível (*lowmem killer*).

Bibliotecas

O Android inclui um conjunto de bibliotecas C/C++ usadas por vários componentes do SO conforme se ilustra na Figura 20.

Figura 20 - Camada Bibliotecas [43].

Estão incluídas nesse conjunto a biblioteca C padrão (Libc) do *Berkeley Software Distribution* (BSD) adaptada para dispositivos com Linux. As bibliotecas suportam os mais populares formatos de áudio e vídeo, bem como imagens estáticas, visualização de camadas 2D e 3D, funções para navegadores *web*, gráficos, aceleração de *hardware*, 3D *render*, fontes *bitmap* e vectorizadas e funções de acesso a base de dados SQLite. Todos estes recursos estão disponíveis no *framework* para o desenvolvimento de aplicativos. Ao contrário da maioria das distribuições Linux, o ambiente gráfico (*Graphical User Interface* - GUI) do Android não é baseado em X-window mas sim desenvolvido pela Google e optimizado para dispositivos móveis.

Android Runtime

Apesar do Android usar como linguagem de programação Java, o SO não possui uma máquina virtual Java (JVM), mas sim a máquina virtual *Dalvik*, optimizada para execuções em dispositivos móveis. A Figura 21 apresenta os componentes do Android Runtime.

Figura 21 - Camada Android *Runtime* [43].

Cada aplicação Android corre o seu próprio processo numa instância da máquina virtual *Dalvik*. Esta máquina virtual foi escrita para ser executada eficientemente com múltiplas

instâncias e executa arquivos *.dex*, um formato de *bytecode* desenhado para ocupar menos espaço e carregar rapidamente, os arquivo *.dex* e outros recursos como imagem. Estes ficheiros são compactados num único arquivo com a extensão *.apk* (*Android Package File*) que representa a aplicação final, pronta para ser instalada.

Application *Framework*

Nesta camada, conforme é apresentado na Figura 22, os desenvolvedores têm acesso à mesma *framework* das *Application Programming Interface*'s (API's) utilizadas para as aplicações Android, o que simplifica a reutilização de componentes e abstrai parte dos procedimentos necessários para se fazer uma aplicação funcionar.

Figura 22 - Camada *Application Framework* [43].

Uma grande vantagem da *framework* de aplicação do Android é a de que cada aplicação pode publicar suas capacidades para serem utilizadas por outras aplicações que estiverem em execução.

Aplicações

A camada de aplicações é a mais alta da arquitectura da plataforma Android, onde se encontram aplicações tais como correio electrónico, *browser*, mapas, calendários e jogos, conforme ilustrado na Figura 23.

Figura 23 - Camada Aplicações [43].

Estas são escritas na linguagem Java e convertidos para arquivos .dex, um *bytecode* da instância da máquina virtual *Dalvik*, em que trabalham.

2.3.2. AMBIENTE DE DESENVOLVIMENTO

O ambiente de desenvolvimento do Android (*Android SDK manager*) é composto pela API das classes Java específicas do Android e um emulador, que permite executar e testar as aplicações desenvolvidas, com o intuito de incentivar o desenvolvimento de aplicativos. O ambiente gráfico do emulador é visível na Figura 24.

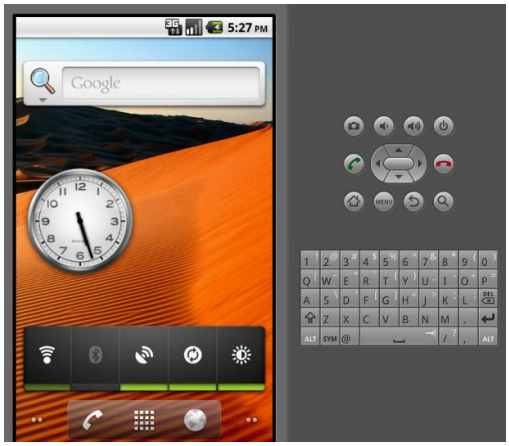


Figura 24 - Visualização do emulador Android.

De forma a facilitar a adaptação dos desenvolvedores a interfaces já popularizados, é possível integrar o SDK em ambientes de desenvolvimento tais como Eclipse, Netbeans, IntelliJ IDEA, ou JDK 5 ou JDK 6, Apache Ant 1.6.5 ou superiores para Linux e Mac, 1.7 ou superiores para Windows, sendo o Eclipse o ambiente de desenvolvimento recomendado pela Google. De forma a integrar o SDK no Eclipse, a Google disponibiliza o *plugin Android Development Tools* (ADT). A interface de desenvolvimento de aplicações Eclipse é apresentada na Figura 25.

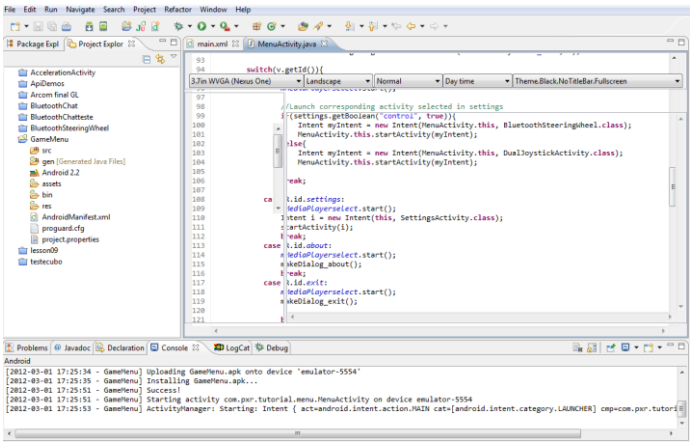


Figura 25 - Ambiente gráfico do Eclipse.

Conforme se pode observar, o menu de navegação das aplicações em desenvolvimento apresenta-se à esquerda, a janela de programação no centro e vários *tabs* de *debug* na janela inferior.

2.3.3. CICLO DE VIDA DA APLICAÇÃO

À semelhança com outros sistemas operativos, na plataforma Android uma aplicação pode ser utilizada em qualquer altura, assim como ser terminada quando não for necessária. Durante o tempo de vida de uma aplicação Android, esta pode encontrar-se em vários estados, como mostra a Figura 26.

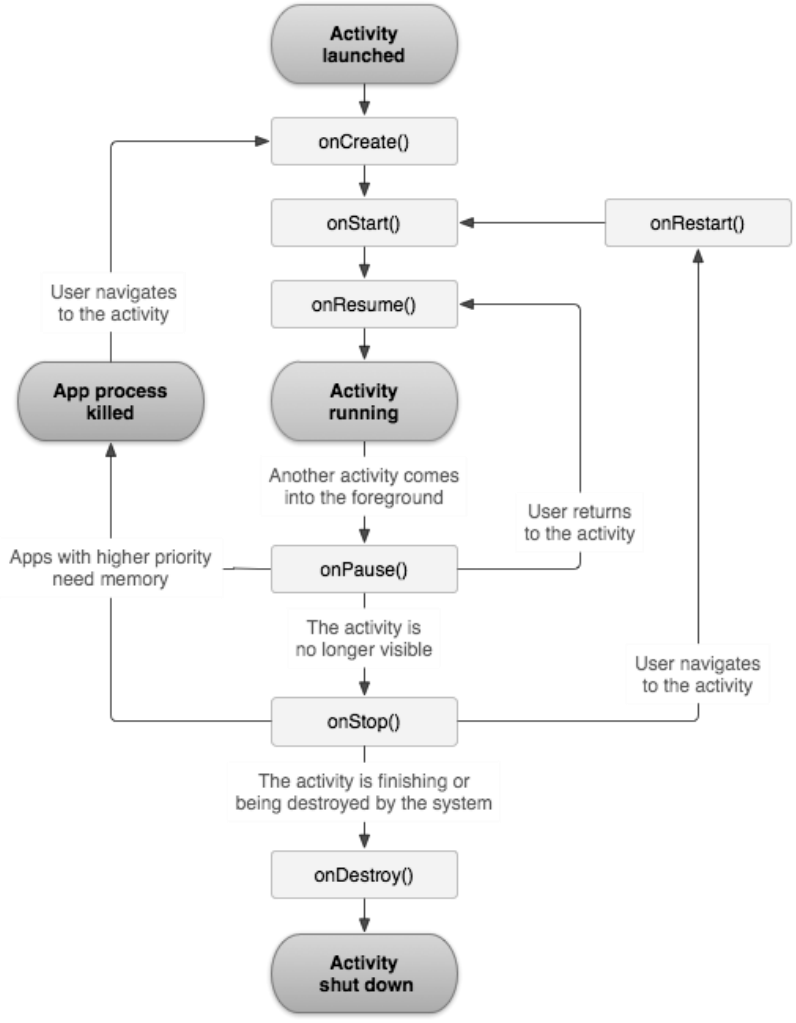


Figura 26 - Ciclo de vida de uma aplicação Android [44].

O estado em que se encontra uma aplicação é totalmente gerido pelo SO, não tendo o utilizador da aplicação que se preocupar com a sua mudança dos estados, para além de definir o código a executar em cada mudança.

Como para correr uma nova aplicação é sempre necessário uma chamada à classe *Activity*, usar-se-á este último termo para nos referirmos à actividade de uma aplicação.

O programador não tem controlo sobre o estado em que a aplicação se encontra, podendo no entanto ser notificado quando o estado está prestes a mudar através da chamada aos métodos *onResume( ), onPause( )*, etc.. O programador deve portanto saber como gerir os métodos na classe *Activity*, sendo da responsabilidade do SO utilizar estes métodos na altura certa. Apresentam-se de seguida alguns dos métodos mais utilizados.

- **onCreate():** este método é chamado quando a *activity* é iniciada. Pode receber como parâmetro o valor NULL ou algumas informações sobre o estado guardado anteriormente pelo método *onSaveInstanceState()*.

- **onStart():** este método indica que a *activity* está em execução e a ser mostrada ao utilizador.

- **onResume():** é chamado quando a aplicação pode começar a interagir com o utilizador. Geralmente é aqui que se fazem arrancar algumas animações e músicas.

- **onPause():** é chamado quando uma *activity* que está a ser executada se encontra prestes a ir para plano de fundo, geralmente porque uma outra foi iniciada.

- **onStop():** este método é chamado quando uma aplicação deixa de ser visível para o utilizador e não deverá ser necessária por algum tempo. Apesar de neste estado a aplicação não estar visível, continua a correr em memória, assim sendo o estado *onStop()* não se deve utilizar se a memória for pequena, pois o SO como protecção pode simplesmente terminar o seu processo.

- **onRestart():** este é o método chamado para arrancar uma *activity* que se encontra em modo *onStop( )*.

- **onDestroy():** é chamado imediatamente antes de uma *activity* ser encerrada.

- **onSaveInstanceState():** o SO invoca este método para permitir à *activity* guardar um estado num determinado instante.

- **onRestoreInstanceState():** este método é chamado quando uma *activity* está a ser reiniciada a partir de um estado guardado anteriormente pelo método *onSaveInstanceState( )*.

Como no SO Android apenas uma *activity* pode correr em plano principal de cada vez, as actividades que estiverem a ser executadas em plano de fundo são forçadas a parar, podendo o processo destas ser destruído se houver falta de memória para correr novas actividades.

Uma vez que este acontecimento ocorre com bastante frequência, o desenvolvimento de aplicações para Android deve levar este aspecto em conta desde o início, havendo sempre a necessidade de usar o método *onPause()* para guardar os dados, uma vez que este pode ser o último método a ser chamado antes do processo relativo a uma *activity* ser destruído.

2.4. DESENVOLVIMENTO MULTIPLATAFORMA

De forma a agilizar o desenvolvimento de uma aplicação para múltiplas plataformas móveis, existem alguns tipos de *software* que permitem o desenvolvimento de uma aplicação única, compatível com várias plataformas.

As soluções de desenvolvimento multiplataforma oferecem essencialmente uma camada de abstracção da sintaxe própria de programação de cada sistema operativo, sendo o código fonte único. Na fase de compilação, é então gerado código específico para cada SDK que por sua vez cria a aplicação final.

Graficamente pode-se representar este processo conforme a Figura 27.

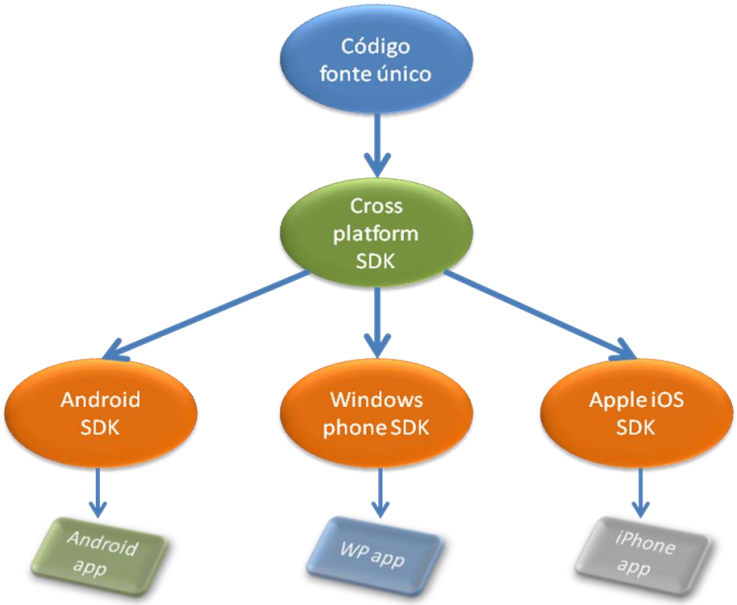


Figura 27 - Representação do desenvolvimento de *software* multiplataforma.

Algumas das principais *frameworks* nesta área são o Phonegap, Worklight, QT, Mosync, AppFurnace e Canappi.

2.5. TESTE E DEPURAÇÃO DE SISTEMAS

O teste de sistemas (*hardware* ou *software*) tem lugar no final do ciclo de desenvolvimento do trabalho e apresenta um resultado binário, i.e. o produto passa ou é rejeitado. Os conceitos de *defeito*, *falta* e *erro* são subjacentes ao teste e relacionam-se como se apresenta na Figura 28.

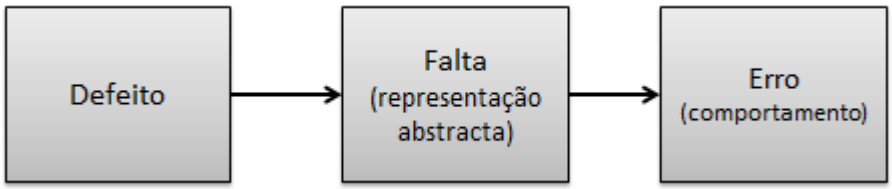


Figura 28 - Relação entre *defeito*, *falta* e *erro*.

O *defeito* é um fenómeno susceptível de impedir o correcto funcionamento de um sistema. A *falta* representa uma condição abstracta que pode levar o modelo dum sistema a apresentar uma resposta incorrecta. Finalmente, o *erro* representa um desvio face ao comportamento esperado do sistema. Para realizar o teste de um sistema é necessário

aplicar estímulos na suas entradas (sejam eles sinais ou pacotes de dados cujo conteúdo é conhecido), capturar as respostas nas suas saídas e compará-las com o resultado esperado.

A depuração está essencialmente associada à etapa de prototipagem e inclui três tipos de operações associadas à presença de *erros* / *defeitos*:

- A *detecção* pretende revelar a sua presença e produz um resultado binário (sim/não).
- O *diagnóstico* pretende efectuar a sua caracterização (identificar, localizar, etc.).
- A *correcção* pretende removê-los.

A *depuração* é realizada sobre um protótipo do sistema em desenvolvimento e pode-se apoiar em equipamentos específicos para este efeito, ou noutros sistemas desenvolvidos propositadamente para facilitar este processo.

3. ESTADO DA ARTE

Neste capítulo apresentam-se algumas soluções semelhantes à que se pretendeu desenvolver, do ponto de vista de *hardware* e *software*. Essencialmente são apresentados os elementos físicos constituintes da rede WSN bem como os *software'*s de visualização de dados e controlo remoto da infra-estrutura.

3.1. SISTEMAS DE MONITORIZAÇÃO E CONFIGURAÇÃO WSN

Nesta secção são apresentadas algumas redes de sensores distribuídos semelhantes à do trabalho desenvolvido, com a descrição dos seus constituintes.

3.1.1. SENSBEE

SENSbee é um exemplo de uma solução de sensores sem fio de baixo consumo, desenvolvido com a plataforma tecnológica da empresa espanhola SAYME. Esta solução permite a criação de redes completas para monitorizar e controlar, à distância, variáveis de diferentes tipos, tais como níveis de poluição, consumos energéticos, vibrações em edifícios ou em motores na indústria. O resultado final são aplicações de fácil instalação que resolvem, de forma bastante precisa, as necessidades do projecto em que se inserem e que, para além disso, as suas características podem facilmente ser expandidas.

No extremo da rede encontram-se os elementos denominados SPOT's, que consistem num sensor, um módulo de condicionamento de sinal que integra os sinais do sensor na plataforma electrónica e um módulo de comunicações sem fios de baixo consumo, conforme ilustrado na Figura 29.

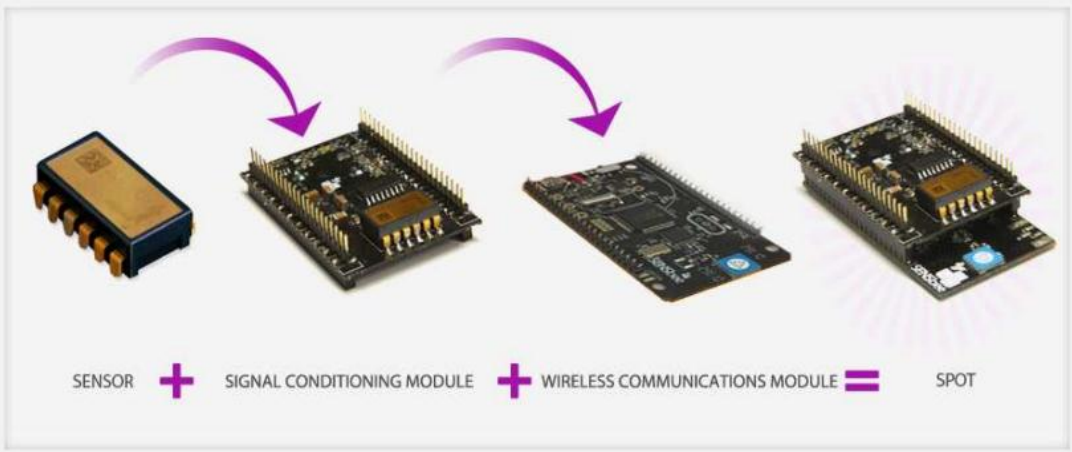


Figura 29 - Módulo RFD SPOT da SENSbee [45].

Os *SPOT*'s são então distribuídos espacialmente de acordo com as variáveis que se pretende monitorizar. Tratam-se dos elementos *Reduced Function Device (RFD)* da rede, podendo a estes serem ligados até seis sensores [45]. Após a variável ser medida pelo sensor e tratada pelo módulo de condicionamento de sinal, o SPOT transmite ao *MASTER* da rede através da tecnologia *SAYME Wireless Area Network* (SWAN), baseada no *standard* IEEE 802.15.4, para que a informação possa ser armazenada e utilizada. O módulo *MASTER* facilita a comunicação local e remota através de múltiplas interfaces *standard*, de forma a adaptar-se de um modo mais eficaz à aplicação em causa. Trata-se do módulo *Full Function Device* (FFD) estando ilustrado na Figura 30.

Figura 30 - Módulo FFD MASTER da SENSbee [45].

A arquitectura fundamental do SENSbee está apresentada na Figura 31.

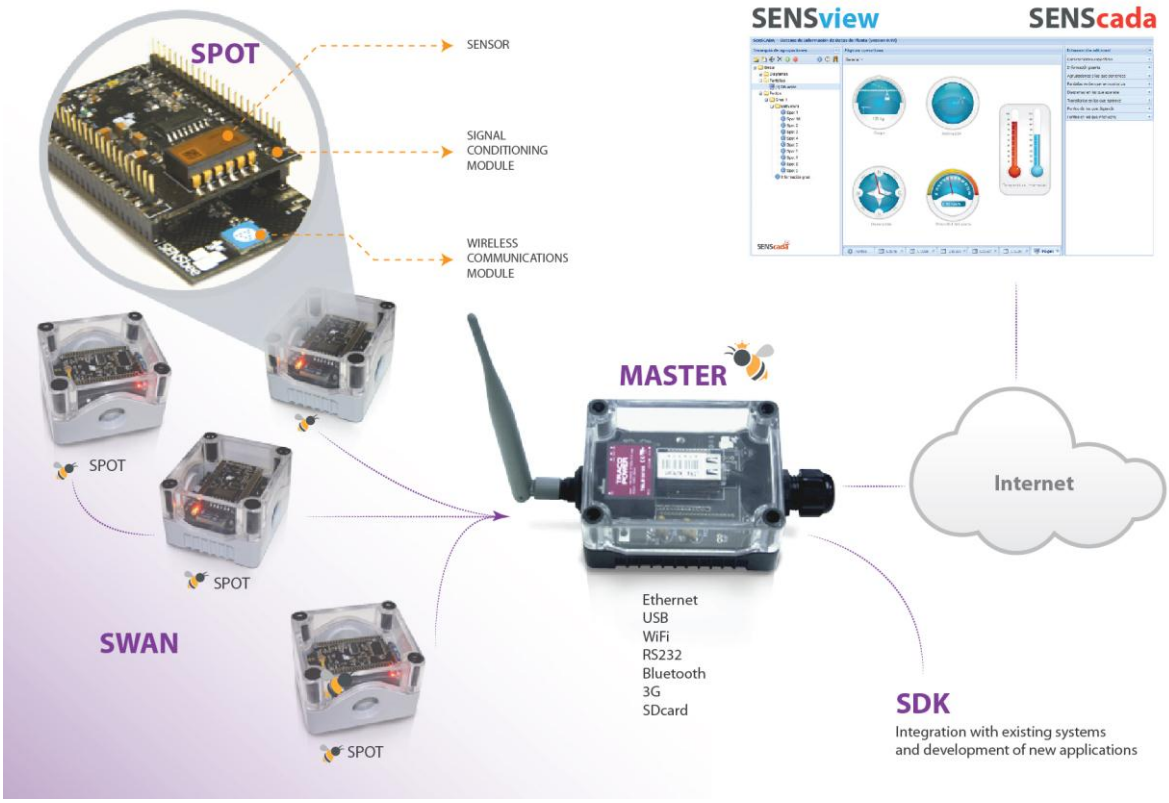


Figura 31 - Arquitectura fundamental do sistema SENSbee [45].

Os desenvolvedores do projecto SENSbee proporcionam também vários *softwares* de gestão e visualização dos dados captados pelos sensores e transmitidos pelos SPOT's. Estes podem ser acedidos em tempo real a partir de qualquer *browser* da Internet, e permitem a configuração de vários parâmetros como alarmes caso determinado parâmetro ultrapasse um valor limite, gestão da informação e recolha em base de dados e visualização gráfica.

3.1.2. AQUAMON

O sistema de monitorização Aquamon surgiu no âmbito da constante demanda de recursos hídricos pelos países em crescimento quando, nalguns casos, estas reservas estão fixas ou a diminuir. Para além disso, os custos associados com a recolha e distribuição de água para fins relacionados com a agricultura continuam a aumentar. Desta forma, monitorizar o solo torna-se essencial para reduzir custos, minimizar o consumo de água, e aumentar a produtividade dos terrenos e colheitas. Assim, conseguir produzir mais, gastando menos água, é o objectivo fundamental deste projecto.

O sistema Aquamon trata-se assim de uma rede de sensores e actuadores que permitem monitorizar e controlar algumas características do solo alvo tais como a temperatura e humidade. A arquitectura deste sistema é apresentada na Figura 32. Estruturalmente existem três tipos de elementos funcionais:

- Unidades sensor (*sensor nodes*);
- Unidade central (*base station*);
- *Software* Aquamon RSVP baseado na *Web*;

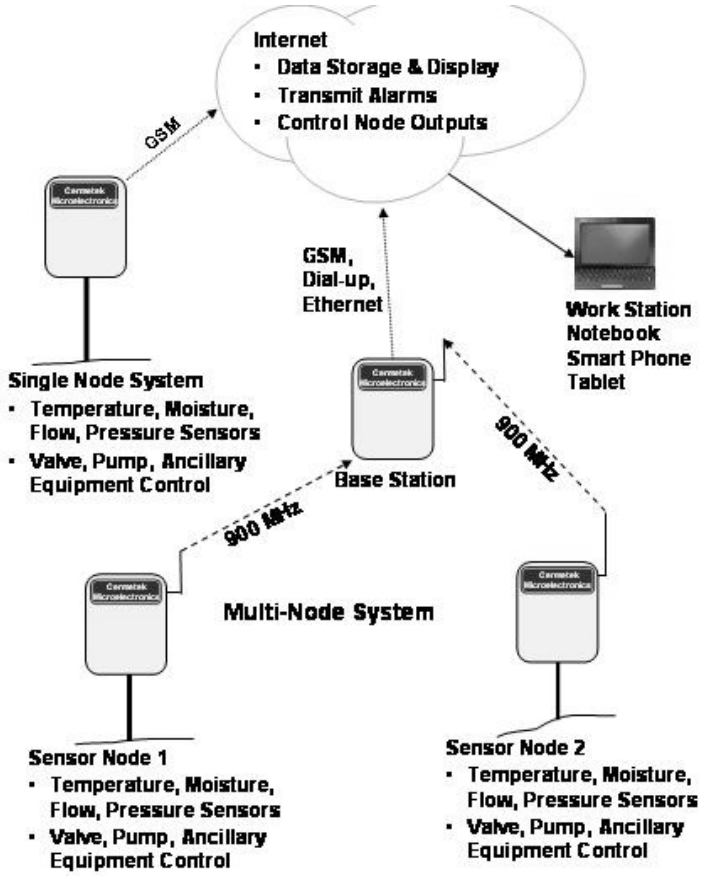


Figura 32 - Arquitectura da rede WSN Aquamon [50].

De forma a monitorizar as variáveis pretendidas, são distribuídos vários *nós sensores* pela zona de cultivo alvo, com capacidade de comunicação *wireless* com uma unidade central denominada *Base station*. Esta pode, por sua vez, enviar os dados para a Internet através de uma rede *Ethernet*, *Dial-up* ou GSM. O acesso a estes dados pode ser efectuado depois através de uma página de Internet a partir de qualquer PC ou *smartphone*.

O sistema Aquamon recolhe informação através dos sensores espalhados pelo terreno de cultivo (*Sensor Nodes* na Figura 32), e transmite estes dados para a unidade central através de uma rede *wireless*. A comunicação entre unidade de sensor e unidade central é possível em distâncias de até 16 Km sem obstáculos físicos entre os dois elementos. De forma a ultrapassar obstáculos tais como montes, pode-se utilizar a capacidade de repetidor presente em cada unidade sensor. Cada *nó sensor* possui cinco entradas analógicas para ligar sensores (humidade, temperatura, pressão atmosférica, etc.) e doze entradas/saídas (*Input/Output* – I/O) digitais para controlar dispositivos externos tais como válvulas para corte ou fornecimento de água. A unidade central recolhe a informação fornecida pelos *nós sensor*, e faz *upload* desta para a Internet, através de GSM/GPRS, *Modem* ou *Ethernet*.

A disponibilização destes dados através da Internet possibilita que estes sejam acedidos a partir de qualquer lugar em qualquer instante. Os dados são guardados num servidor seguro com controlo através de *password* definida pelo utilizador. O *software* Aquamon RSVP permite assim que quaisquer dados sejam acedidos em MAC, Windows PC, iPad, ou qualquer *smartphone* com acesso à Internet.

O *software* Aquamon permite fazer o *download* da informação recolhida para análise posterior. Este *software* permite também ao utilizador controlar válvulas de rega, bombas de água ou outro equipamento a partir das doze entradas e saídas digitais disponíveis.

O sistema Aquamon possibilita também ao utilizador definir limites para os sensores gerando alarmes específicos. Caso esteja assim configurado, o sistema irá transmitir para o utilizador *e-mails* ou SMS para o número de telefone ou *e-mail* definido.

3.1.3. LIBELLIUM WASPMOTE

Libellium é uma empresa espanhola responsável pelo desenvolvimento do Waspmote e Meshlium. Estes dois dispositivos são responsáveis pela rede WSN da Libellium. O Waspnote funciona como o elemento responsável pela captura e condicionamento do sinal e transmissão para o Meshlium, um *router wireless* que pode permitir ligação à Internet para disponibilização da informação. A Figura 33 apresenta o pormenor das cartas do Waspmote.

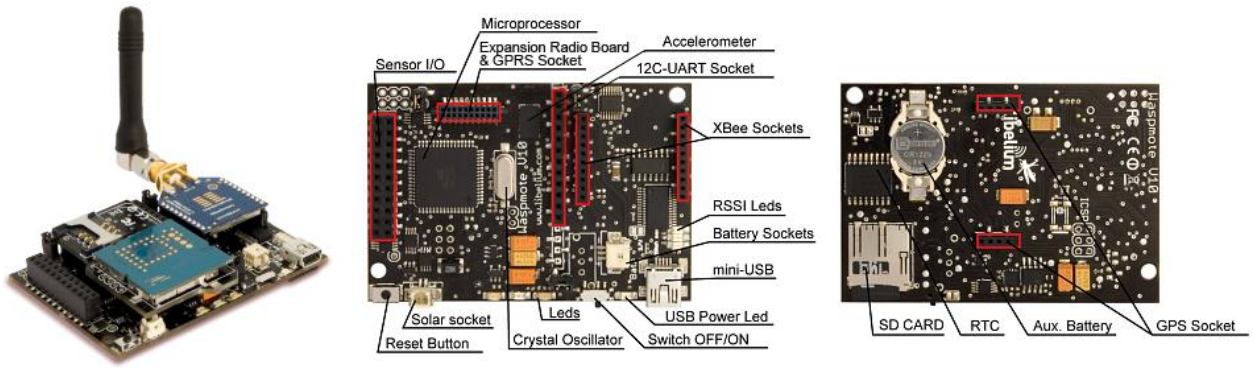


Figura 33 - Fotografias das cartas Libellium Waspmote [48].

Os dispositivos Waspmote têm várias opções de funcionamento e de ligação à rede. Podem-se ligar a um *router* WiFi *Standard* (norma b/g), ou realizar comunicação directa para dispositivos Android ou iOS sem necessidade de *router*, isto é, estabelecendo uma ligação *ad-hoc*. Adicionalmente podem-se ligar a um servidor *web* através dos protocolos HTTP ou HTTPS, ou utilizando *sockets* TCP/IP e UDP/IP.

3.1.4. SENSORCLOUD

O SensorCloud, desenvolvido pela Microstrain, é essencialmente um centro de armazenamento de informação recolhida pelos sensores compatíveis, que foram registados na sua base de dados. Os dados enviados para a Internet são guardados continuamente e é possível visualizar a informação via *web* na página da marca.

Originalmente, o SensorCloud foi desenvolvido para permitir o suporte da marca a instalações de sensores realizadas, durante um longo período de tempo, tendo entretanto evoluído para suportar qualquer dispositivo com acesso à Internet que forneça leituras provenientes de um sensor. O serviço SensorCloud oferece espaço de memória para dados recolhidos virtualmente infinito, com protecção contra falhas com *backups* da informação, sem custos na configuração básica, uma solução para armazenamento de dados dos sensores simples e sem custos para o utilizador. Este serviço permite também o envio de alertas por SMS e *e-mail*, totalmente configuráveis pelo utilizador. Os dispositivos suportados por este serviço incluem os sensores *wireless* de redes WSN disponibilizados pela Microstrain que, de origem, suportam o acesso ao serviço, sensores inerciais também da marca, e mesmo *smartphone*s Android e iOS. Para a recolha e *upload* dos dados dos

acelerómetros, magnetómetro e giroscópio, é necessário descarregar a aplicação do *Market* bem como manter uma ligação à Internet.

3.1.5. COSM

A plataforma Cosm, fundada em 2008, parte do conceito *Internet of things* - a presença de sensores em objectos físicos que lhes permitem atribuir alguma inteligência artificial. A conectividade com a Internet e consequentemente de uns dispositivos com os outros permite então criar um mundo à parte - a *Internet das coisas*.

O Cosm suporta algum *hardware* desenvolvido por empresas independentes ou pela comunidade e *hardware* já existente, como o *Arduino*. Um desses exemplos é o sistema desenvolvido pela empresa Arexx, ilustrado na Figura 34.

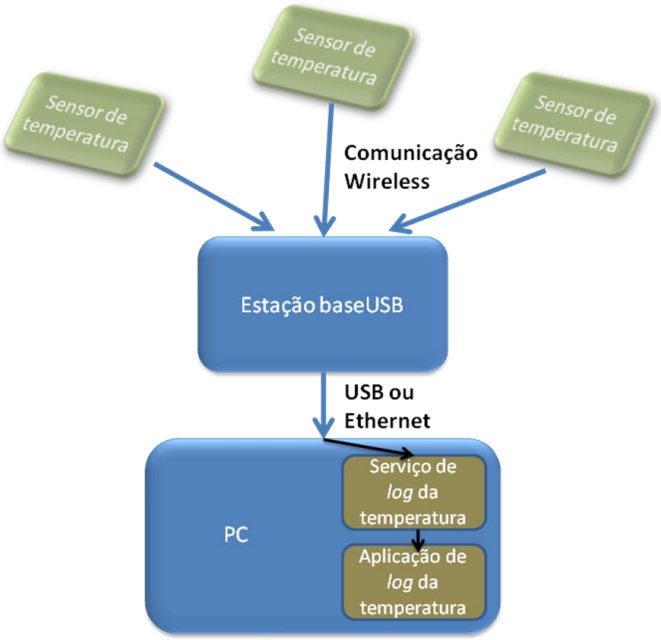


Figura 34 - Esquema de funcionamento do sistema desenvolvido pela Arexx.

A este módulo é possível ligar até 60 sensores de temperatura. A ligação é feita ao PC por USB ou LAN e, no caso do Cosm, é então enviado para os seus servidores a informação recolhida constantemente. Note-se que a empresa também fornece o *software* para consulta da informação no PC.

3.2. *SOFTWARE* DE DISPONIBILIZAÇÃO DE DADOS

Nesta subsecção pretende-se analisar de modo comparativo a interface gráfica dos produtos da subsecção anterior procurando realçar as vantagens relativas.

3.2.1. INTERFACE SENSVIEW

A empresa Sayme desenvolveu três *software*'s para a gestão e controlo da sua rede *wireless*, o SENScada, SENSview e um SDK de desenvolvimento e configuração dos seus componentes de *hardware* SPOT's e MASTER's. O SENSview consiste numa aplicação *web* que permite apresentar uma interface gráfica para visualização animada em tempo real dos dados obtidos a partir dos SPOT's. A Figura 35 apresenta o interface de utilizador do SENSview.

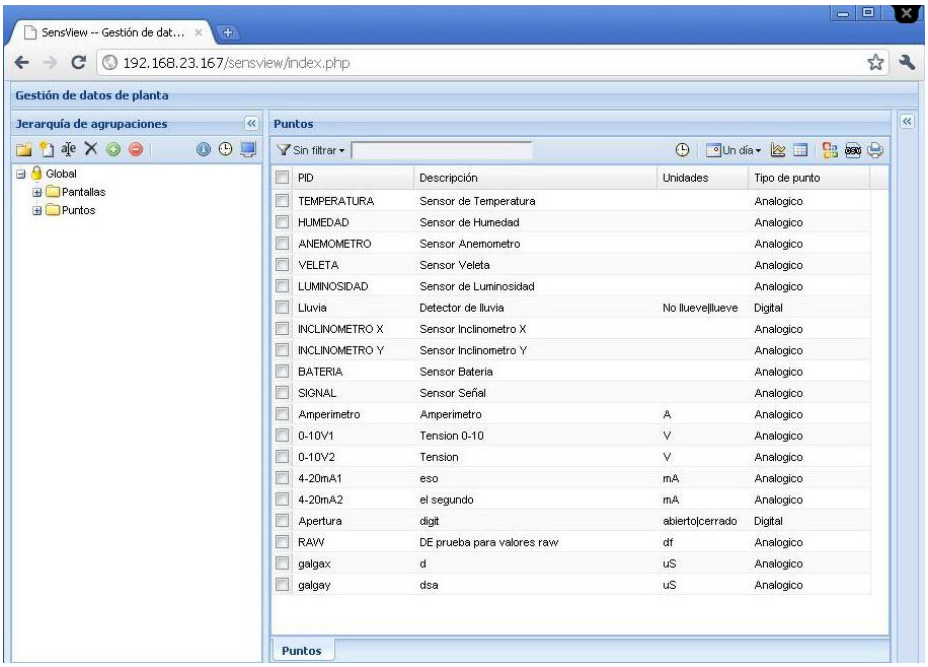


Figura 35 - Interface gráfico *web* do SENSview [45].

O SENScada trata-se de uma solução *Supervisory Control And Data Acquisition (SCADA)* para gerir os dados provenientes dos *nós* SPOT e a sua integração com dados provenientes de outras redes que utilizem *Programmable Logic Controllers* (PLC). O SENScada trata assim de interligar, através de uma aplicação de *software*, dados provenientes do sistema SCADA que pode já estar implementado na fábrica com os fornecidos pela WSN. A Figura 36 apresenta a interface de utilizador do SENScada.

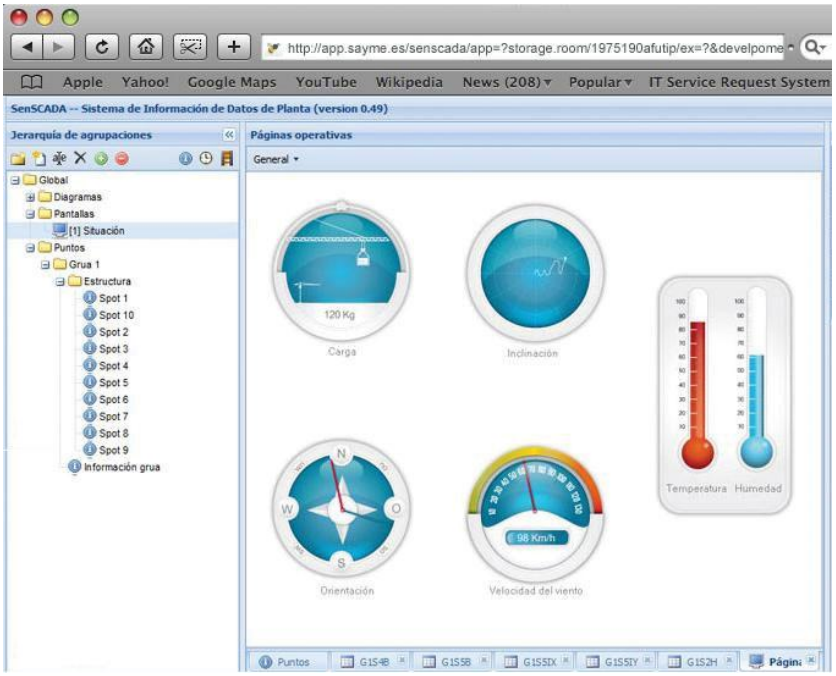


Figura 36 - Interface *web* SENScada [45].

Como é possível observar, o SENScada apresenta do lado esquerdo da interface uma hierarquia de pastas com todos os dispositivos SPOT que estejam ligados à rede WSN, para que se possa aceder a estes e visualizar os dados recebidos por esse *nó sensor*.

Adicionalmente, a empresa desenvolveu um *Software Development Kit* (SDK) de acesso ao *hardware* que permite o uso do SENSbee por todos os tipos de utilizadores, desde pessoal qualificado em desenvolvimento de *software* até àqueles que pretendem controlar o *nó sensor* ao mais alto nível com o SENScada e o SENSview mesmo sem terem formação extensiva neste campo.

3.2.2. INTERFACE AQUAMON

Apesar de ainda não estar disponível, o *software* de visualização sofreu já algumas modificações. Inicialmente previsto para correr através do *browser* do *smartphone*, PC ou MAC, a versão actual funciona somente em Windows 7, Vista ou XP. O ambiente gráfico está apresentado na Figura 37.

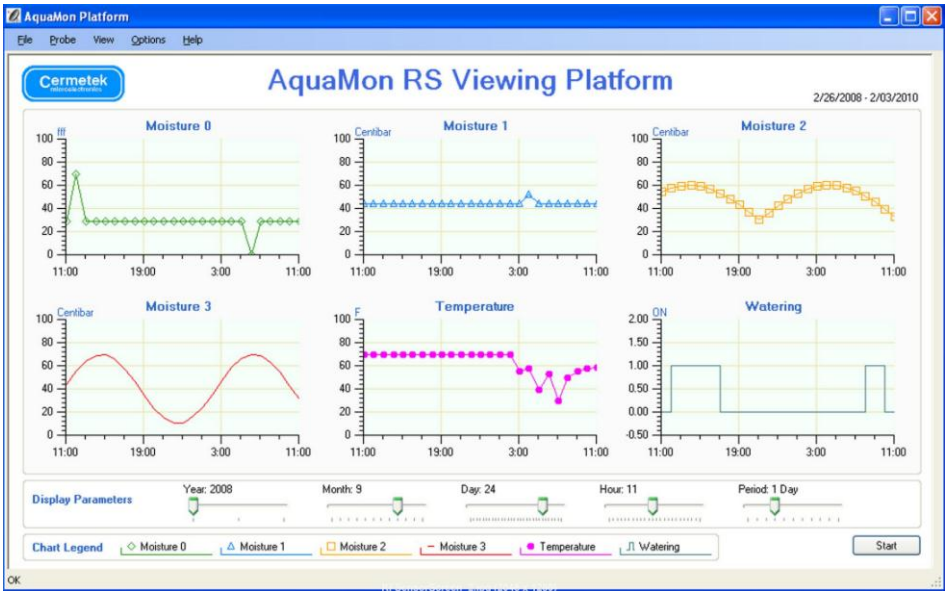


Figura 37 - Interface gráfica Aquamon RS [50].

Essencialmente este *software* recolhe a informação do servidor *web* onde está guardada toda a informação recolhida pela unidade central em campo e disponibiliza-a sob a forma de gráficos para o utilizador. É possível definir o período que se pretende verificar através do campo *display parameters*.

3.2.3. INTERFACE LIBELLIUM

A Libellium desenvolveu já aplicações para Android e iOS, para monitorizar as leituras realizadas pelos Waspmote que se interligam directamente com o *smartphone* através de uma rede *ad-hoc*. Além de monitorização permitem ainda o controlo de dispositivos aos quais o Waspmote esteja ligado [48]. O aspecto geral da aplicação está evidenciado na Figura 38.

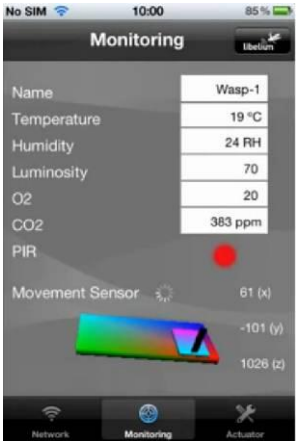


Figura 38 - Aplicação iPhone Libellium Waspmote [32].

A aplicação é composta por três *tabs*: um de identificação e ligação à rede pretendida, um de monitorização dos valores recebidos e outro de controlo de saídas que estão ligadas ao Waspmote. O *tab* de monitorização neste caso disponibiliza em caixas de texto os valores recebidos pelo Waspmote, como temperatura, humidade, luminosidade, níveis de oxigénio e dióxido de carbono. Adicionalmente indica a detecção de movimento num alcance de até oito metros e as coordenadas enviadas pelo acelerómetro. O polígono no ecrã consiste numa animação em OpenGL e é movido de acordo com os valores recebidos do acelerómetro. Para tal são usados três Waspmotes conforme indicado na Figura 39.

Figura 39 - Detectores compatíveis com a aplicação [48].

Cada módulo Waspmote estabelece ligação com o iPhone, apresentando os dados recolhidos pelos sensores acoplados.

3.2.4. INTERFACE SENSORCLOUD

O interface desenvolvido pelo SensorCloud (apresentado na Figura 40) é acedido via *web*.

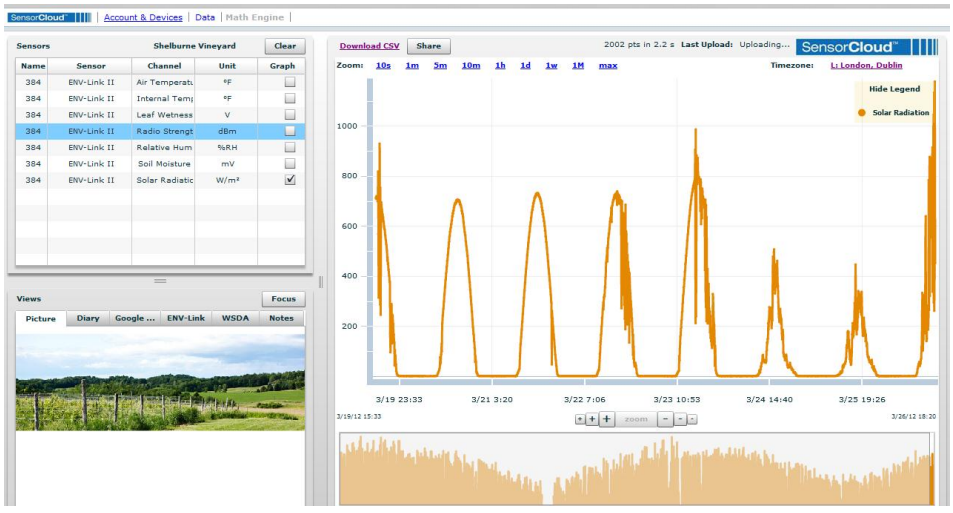


Figura 40 - Interface gráfico SensorCloud [51].

Toda a aplicação corre no *browser* utilizado e acede à informação guardada na nuvem. São apresentadas três janelas principais ao utilizador: a de selecção das medidas a visualizar no canto superior esquerdo, a janela de informação genérica sobre os dados que se estão a analisar e uma janela principal de visualização dos dados sob a forma de gráficos.

Na janela principal de visualização de dados é possível definir qual o período que se pretende observar, desde os últimos dez segundos até desde o início da gravação dos dados no servidor.

3.2.5. INTERFACE COSM

O interface Cosm de visualização dos dados recolhidos é feito também no *browser*. Contudo, os desenvolvedores optaram por disponibilizar as leituras sob a forma de *feeds* dos utilizadores, semelhante a um fórum onde cada elemento pode partilhar as suas leituras. Um exemplo de um *post* deste tipo está apresentado na Figura 41.

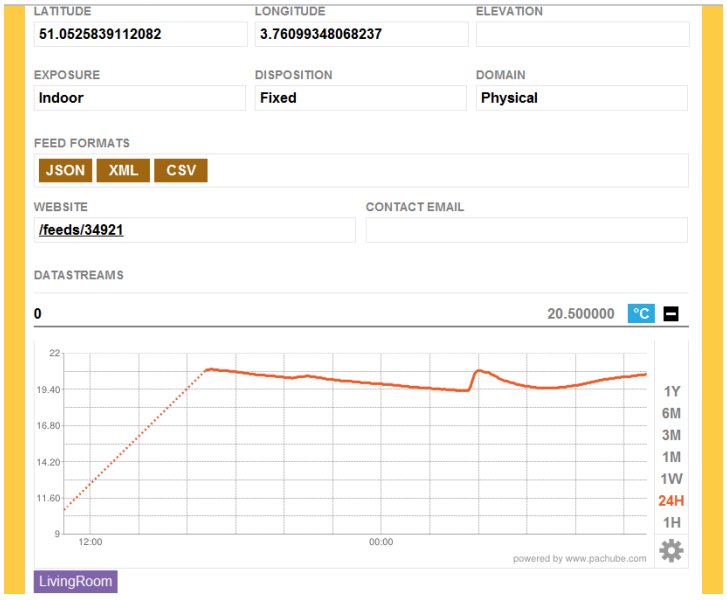


Figura 41 - Interface *web* Cosm [47].

Neste *post* o utilizador opta por partilhar a sua localização e a leitura de temperatura de uma divisão da casa realizada ao longo das últimas 24 horas. Opta também por partilhar as leituras dos sensores com o público em geral, sem restrições de acesso.

4. SOLUÇÃO WESENSENET

Este capítulo descreve a solução *WeSense Network* desenvolvida pela Evoleo, que consiste em *hardware* e *software* desenvolvido durante o estágio na empresa. Esta fase do trabalho revelou-se particularmente difícil dado que o *hardware* estava ainda em desenvolvimento e como tal, não havia ainda uma definição clara dos objectivos a atingir ao nível da interface gráfica. Durante a parte inicial deste trabalho, houve assim a necessidade de definir quais os objectivos a cumprir pelo protótipo global (*hardware* e *software*).

Será assim apresentado neste capítulo o *hardware* desenvolvido pela equipa na Evoleo Technologies, a definição do projecto a desenvolver, as opções de projecto, os requisitos do sistema e finalmente será apresentada a solução projectada para a plataforma.

4.1. MÓDULOS WESENSE

Os módulos responsáveis pela recolha da informação na instalação são a essência da solução *WeSenseNet*. Actualmente estão já em fase de produção dois módulos diferentes, um de aquisição genérica de dados, o WeSense DAQ, e outro de aquisição de dados referentes aos consumos energéticos dos dispositivos, o WeSense ENERGY. As características gerais de cada um dos módulos são apresentadas nesta subsecção.

WeSense DAQ:

As características do WeSense DAQ são as seguintes:

- Comunicação *wireless* para transmissão dos dados – GSM;
- Módulo GPS para determinar a posição de cada módulo;
- Configuração inicial por USB e *software* próprio;
- Possibilidade de alimentação externa ou por bateria interna com autonomia para três anos;
- Sensor de temperatura interna;
- Memória Flash de 4 Mb;
- Quatro entradas analógicas: duas para entradas em tensão (0 a 10 V) e duas para entradas em corrente (0 a 20 mA ou 4 a 20 mA, configurável via *software*);
- Oito entradas digitais, com possibilidade de funcionarem como estado ou como contador: no primeiro caso detecta possíveis eventos que causem alarme ou mudança de estado, no segundo, serve para contagem de impulsos);
- Caixa IP-65 para suportar condições industriais;

A Figura 42 apresenta o protótipo inicial desenvolvido na Evoleo Technologies para este *nó sensor*.

Figura 42 - Módulo WeSense DAQ.

WeSense ENERGY:

As características do WeSense ENERGY são as seguintes:

- Comunicação *wired* – Ethernet;
- Configuração inicial por USB e *software* próprio;
- Possibilidade de alimentação externa ou por bateria interna com autonomia para uma semana;
- Sensor de temperatura interna;
- Memória Flash de 4 Mb;
- Leitura de energia activa, reactiva e aparente do dispositivo em análise;
- Caixa IP-65 para suportar condições industriais;

4.2. DEFINIÇÃO DO PROJECTO

Previamente à definição de requisitos do sistema a desenvolver, foi necessário definir em concreto o trabalho a realizar. A lista inicial de requisitos inclui os seguintes tópicos:

- Recolha e armazenamento de dados;
- Visualização dados;
- Gestão remota dos *nós* sensores;

Os dados recolhidos por cada módulo poderiam ser armazenados pela aplicação a desenvolver, tornando assim possível disponibilizá-los para consultas futuras. A visualização de dados seria fundamental, isto é, verificar de forma interactiva os consumos referentes a energia, água, etc. através de gráficos ou tabelas. Pretendia-se também gerir e reconfigurar cada *nó sensor* remotamente a partir da plataforma a desenvolver. Seguidamente procedeu-se a um estudo para determinar quais as opções possíveis e depois quais as que seriam mais vantajosas para empresa, sendo apresentada a proposta para aprovação. As hipóteses propostas podem ser resumidas na Figura 43.

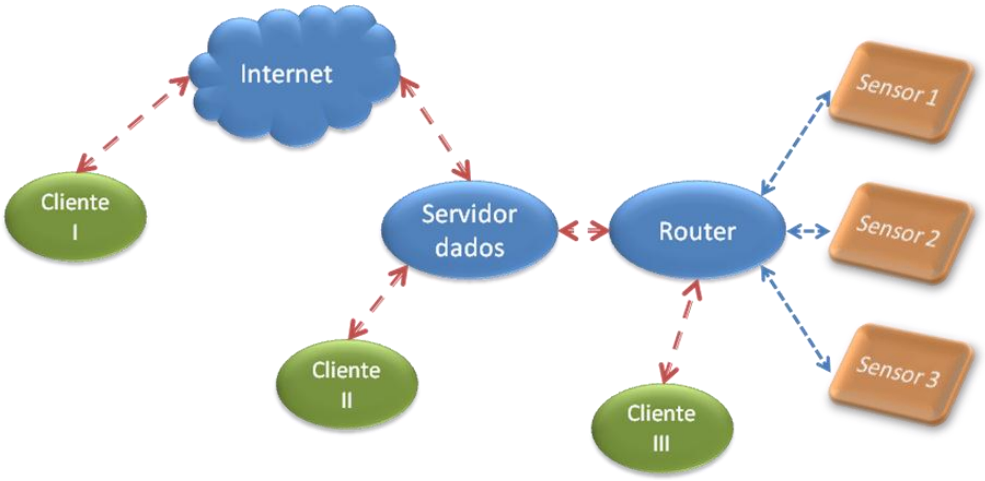


Figura 43 - Opções possíveis de acesso à informação recolhida pelos módulos.

Na Figura 43, podem-se distinguir três pontos de acesso aos dados, simbolicamente representados por Cliente I, Cliente II e Cliente III, que representam hipóteses de diferentes tipos de aplicações e conceitos a desenvolver. De seguida descrevem-se as características de cada caso.

- ***Cliente I*** - Aceder aos dados via *web browser* a partir de qualquer local, guardados no servidor (JavaScript ou Flash);
- ***Cliente II*** - Aplicação instalada no servidor de dados, para controlo e visualização da informação no local da instalação com consulta da informação guardada em base de dados ou simples ficheiro de texto;
- ***Cliente III*** - Acesso directo sem necessidade de servidor de dados – e.g. aplicação Android nativa;

Tendo em conta as alternativas, oferecidas pela situação do *cliente I* e *cliente II*, a solução *cliente III* foi posta de parte desde cedo, uma vez que a ausência de servidor de dados que guardasse constantemente os dados recolhidos pelo módulo, punha de parte a hipótese de ver um histórico das leituras realizadas anteriormente.

Relativamente à solução *cliente II*, não apresenta vantagens do ponto de vista de recolha de informação face ao cliente I, apresentando antes limitações. Com efeito, a informação recolhida só poderia ser acedida no servidor de dados onde o sistema estava instalado.

Desta forma, foi proposto e posteriormente aceite o modelo representado pelo *cliente I*. Com esta solução, torna-se possível o acesso a informação e reconfiguração dos dispositivos instalados a partir de qualquer *browser*, uma vez que os dados são armazenados continuamente no servidor dedicado.

4.3. ESTUDO E OPÇÕES DE PROJECTO

Após a determinação do tipo de aplicação a desenvolver para o sistema, foram colocadas várias hipóteses para tentar resolver o problema. De acordo com a metodologia de trabalho da empresa todas estas hipóteses foram apresentadas seguindo-se depois a sugestão da melhor opção e selecção da solução. Para o desenvolvimento da interface gráfica no *browser* foram também apresentadas algumas hipóteses, nomeadamente em HTML e *Javascript*, Flash, e LabVIEW *Web* UI, que serão analisadas mais detalhadamente nas subsecções seguintes.

4.3.1. INTERFACE GRÁFICA - *JAVASCRIPT*

Uma opção para visualizar a informação é recorrer ao *JavaScript* criando uma *web-app* – essencialmente uma aplicação no navegador. Desta forma, os gráficos podem ser acedidos a partir de qualquer navegador que suporte *JavaScript* (maioria na actualidade, até de *smartphones*). Existem várias *frameworks* que disponibilizam já várias funções, como o desenho de gráficos e interface com utilizador com vários efeitos visuais como transições. Para além disso, algumas destas *frameworks*, permitem com o Phonegap, desenvolver aplicações baseadas na *web* para *smartphones*. As *frameworks* mais populares são o jQuery, Dojo, Enyo, JoApp, MooTools, Ext JS, etc..

De forma a analisar o suporte da comunidade e desta forma a popularidade e possibilidade de resolver problemas que surgissem na fase de implementação, optou-se por consultar o *website StackOverflow* pela ocorrência do nome da *framework* em tópicos diferentes. Obtiveram-se os valores apresentados na Tabela 1.

Tabela 1 Avaliação do suporte da comunidade às várias bibliotecas.

Biblioteca	Nº tópicos
Wakanda	1
Dojo	2782
Mootools	1596
jQuery	166397
enyo	64
joApp	0
Google *web toolkit*	7931
Ext JS	4894
Qooxdoo	125
Echo3	4

Analisando a Tabela 1 pode-se concluir que existe uma clara preferência pela biblioteca jQuery.

4.3.2. INTERFACE GRÁFICA - LABVIEW *WEB* UI E FLASH

Uma possibilidade alternativa consistia na instalação de uma aplicação LabVIEW no servidor que disponibilizasse um *Web service* para ser acedido através da nuvem num *browser*, usando para isso a aplicação desenvolvida no LabVIEW WEB UI. A Figura 44 ilustra o funcionamento desta alternativa.

Figura 44 - Diagrama de blocos exemplificando o funcionamento da proposta.

O desenvolvimento da aplicação com esta ferramenta apresenta vantagens tais como o tempo de desenvolvimento e a acessibilidade do SDK, mas contudo tem limitações relacionadas principalmente com customização e licenças: é necessário licença paga para o

LabVIEW e outra licença, também paga, para o LabVIEW Web UI, o que não é viável uma vez que se pretende utilizar ferramentas livres e a interface do LabVIEW Web UI utiliza o *plugin* SilverLight da Microsoft, não suportado na maioria dos *smartphones* e *tablets.*

4.3.3. INTERFACE GRÁFICA - FLASH

Flash foi também colocado como opção inicial, pelo aspecto visual que poderia ser integrado na aplicação, mas devido à falta de apoio na maioria das plataformas móveis, apresentou-se desde início como uma limitação, sendo por isso rapidamente descartado como primeira opção de desenvolvimento.

4.3.4. DESENVOLVIMENTO MULTIPLATAFORMA DA APLICAÇÃO

Uma alternativa também estudada foi a possibilidade de desenvolvimento de uma aplicação através de *frameworks* de desenvolvimento multiplataforma. Nesta secção apresentam-se as principais *frameworks* analisadas.

PHONEGAP

Phonegap é uma *framework opensource* desenvolvida pela Adobe Systems que permite a criação de aplicações para dispositivos móveis através de Java, HTML5 e CSS3. A aplicação resultante não é assim totalmente nativa (toda a interface gráfica é apresentada utilizando o *webview*) nem tem acesso à totalidade das API's do dispositivo.

A Tabela 2 apresenta as características a que a *framework* tem acesso nos vários sistemas operativos móveis.

Tabela 2 Lista de características *vs* compatibilidade da *framework* Phonegap.

	iOS iPhone / iPhone 3G	iOS iPhone 3GS and newer	Android	OS 4.6–4.7	OS 5.x	OS 6.0+	WebOS	WP7	Symbian	Bada
ACCELEROMETER	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
CAMERA	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
COMPASS	✗	✓	✓	✗	✗	✗	✗	✓	✗	✓
CONTACTS	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓
FILE	✓	✓	✓	✗	✓	✓	✗	✓	✗	✗
GEOLOCATION	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MEDIA	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗
NETWORK	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (ALERT)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (SOUND)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (VIBRATION)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
STORAGE	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗

WORKLIGHT

Worklight pertence à IBM, responsável pela *framework* também com o mesmo nome, Worklight, que de maneira semelhante ao Phonegap, permite o desenvolvimento de aplicações para iOS, Android, Blackberry e Windows Phone. A arquitectura do Worklight é apresentada na Figura 45.

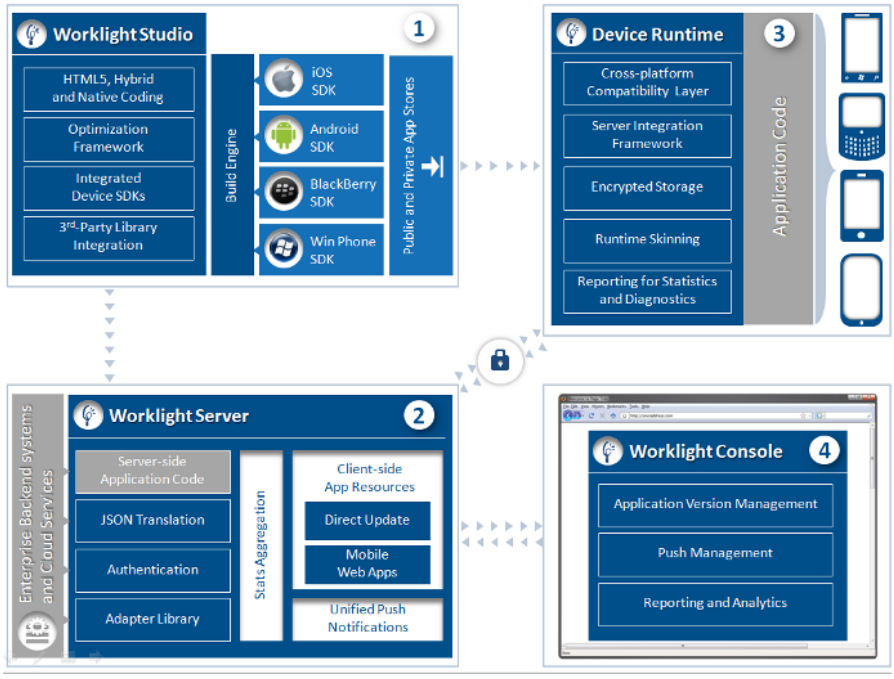


Figura 45 - Arquitectura Worklight [46].

São quatro os principais componentes:

- *Worklight Studio;*
- *Worklight Server;*
- *Worklight Device Runtime Components;*
- *Worklight Console;*

O *Worklight Studio* trata-se de um IDE baseado no Eclipse, que permite aos desenvolvedores de *software* criar todo o código e oferece todas as tarefas de integração necessárias para criar a aplicação. Essencialmente é o Eclipse, com um *plugin* desenvolvido pela empresa.

O *Worklight Server* é um servidor baseado em Java, que consiste numa interligação entre as aplicações, serviços externos e a rede da empresa. O servidor possui características de segurança para permitir conectividade, extracção e manipulação de dados de várias origens, autenticação do utilizador, etc..

Os *Worklight Device Runtime Components* tratam-se de bibliotecas essenciais que complementam o servidor, oferecendo um interface predefinido para aceder às funcionalidades nativas do dispositivo. O Worklight utiliza o Phonegap para este efeito.

O *Worklight Console* é a interface para utilizador em *browser*, que permite a monitorização e administração do servidor Worklight e as aplicações desenvolvidas.

Contudo a licença do Worklight é paga, pelo que desde início se apresentou como uma alternativa dificilmente aceite.

QT

O QT é uma *framework* de desenvolvimento multiplataforma desenvolvida pela Nokia em 1992, e é muito usada para desenvolvimento de *software* multiplataforma com interface gráfico. O interface gráfico do SDK é apresentado na Figura 46.

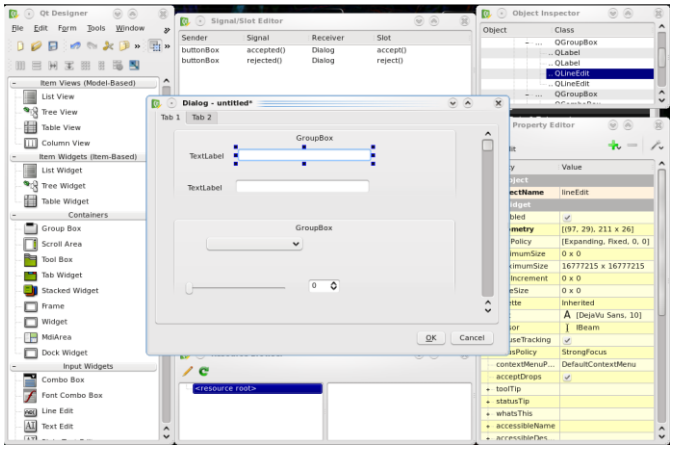


Figura 46 - Ambiente de desenvolvimento do QT [49].

Exemplos de *softwares* desenvolvidos com o QT são o Autodesk Maya, Adobe Photoshop Elements, Skype, VLC Media Player, VirtualBox e Mathematica.

As plataformas suportadas pelo QT actualmente e oficialmente são:

- Windows;
- Windows CE / Mobile;
- Symbian;
- Mac OS X;
- X11;
- Linux para sistemas embebidos;
- Maemo/Meego;
- Wayland;

Depois de a Nokia ter disponibilizado o acesso ao código fonte, as seguintes plataformas ganharam também suporte:

- OpenSolaris;
- Haiku;
- OS/2;
- Amiga OS4;
- iPhone;
- Web OS;
- Amazon Kindle DX;
- Android OS;
- Blackberry OS;

Contudo, surgem alguns problemas com o apoio a estes dispositivos. Tomemos o exemplo do SDK desenvolvido para Android, *Necessitas*. Segundo informação disponibilizada pelos desenvolvedores do *software*, para a execução da aplicação no *Personal Digital Assistant* (PDA) é necessário descarregar outra aplicação, *Ministro*, para descarregar as

bibliotecas específicas necessárias por cada aplicação. A Figura 47 apresenta esquematicamente o desenvolvimento nativo em Android face ao desenvolvimento em QT.

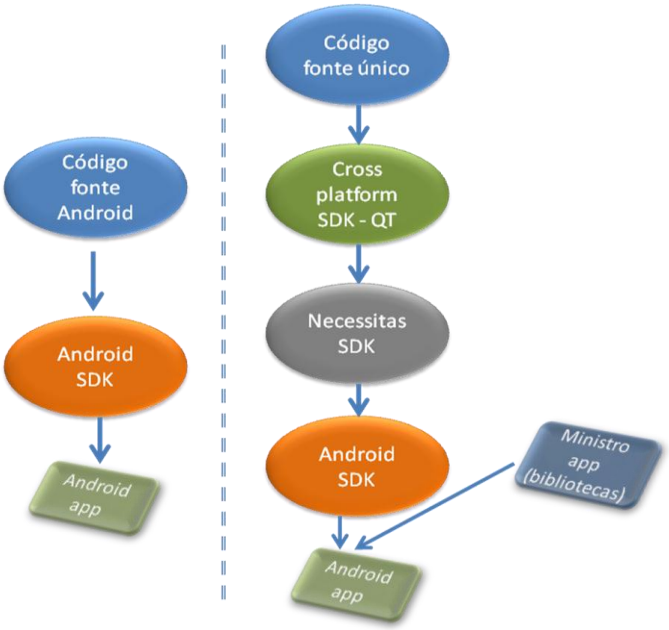


Figura 47 - Desenvolvimento nativo em Android *vs* desenvolvimento em QT para Android.

PROGRAMAÇÃO MULTIPLATAFORMA *VS* DEDICADA

Nesta secção procura-se fazer uma análise comparada entre programação multiplataforma e programação dedicada. A Figura 48 ilustra o desenvolvimento multiplataforma.

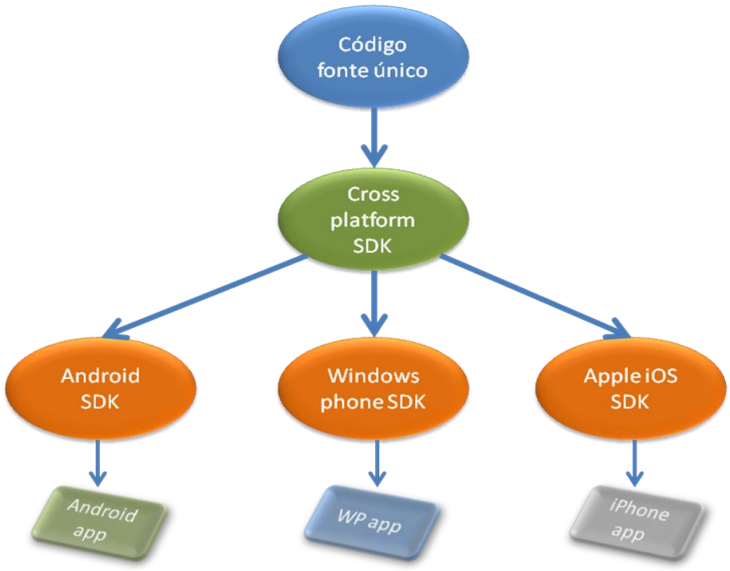


Figura 48 - Desenvolvimento multiplataforma.

O principal e mais óbvio benefício da programação multiplataforma passa pela redução do tempo necessário ao desenvolvimento da aplicação. Ao focar só numa API, poupa-se tempo quer no desenvolvimento, como na subsequente manutenção do código fonte. Adicionalmente, a possibilidade de desenvolver aplicações para várias plataformas móveis recorrendo a ferramentas já bem conhecidas como o HTML, *Javascript* e CSS, torna-se atractivo para novos desenvolvedores de *software*, já que irá fornecer uma abstracção significativa sobre pormenores específicos a cada plataforma e será necessário um menor tempo de aprendizagem por parte do programador, utilizando ferramentas conhecidas. Os factores desfavoráveis que se podem apontar a esta solução são resumidamente os seguintes:

- A dependência de um SDK suportado por terceiros;
- Limitação nas funções suportadas por cada SDK;
- Aprovação das aplicações nos *Markets* principais;
- Grande maioria dos SDK multiplataforma são somente para plataformas móveis;

Relativamente ao primeiro ponto, a dependência de terceiros para suporte a novas funcionalidades que vão sendo adicionadas a cada nova actualização da plataforma em causa, pode nem sempre ser vantajoso. Por exemplo, no caso de o desenvolvedor abandonar o projecto, a adição de novas funcionalidades cessa e torna-se possível que caso sejam feitas alterações a alguma API da plataforma pela marca (Google, Apple, Windows), a aplicação deixe de funcionar. Existem vários casos relatados de SDK's multiplataforma abandonados.

O segundo ponto de limitação torna-se também relevante uma vez que nem todas as funções são suportadas por estes *software'*s. De qualquer modo, e dependendo do projecto, em questão, pode-se facilmente determinar se o *software* cumprirá os requisitos estabelecidos.

Relativamente à aprovação pelos *Markets*, é reconhecidamente maior a política de permissividade de aplicações no *Market* da Google comparando com a da Applestore ou o Marketplace do Windowsphone. O desenvolvimento da aplicação para as três plataformas não garante a aprovação da aplicação pelos três *Markets*, especialmente os últimos dois citados.

Por último, a grande maioria dos SDK's multiplataforma não suporta o desenvolvimento de aplicações para Windows, Mac OS ou qualquer distribuição Linux, sendo maioritariamente orientados ao desenvolvimento multiplataforma móvel.

Sumário do capítulo de opções de projecto

Nesta secção pretendeu-se apresentar todas as opções tomadas na fase inicial de desenho da solução a implementar. Foi consensual a decisão de optar pela interface gráfica em ambiente *web*, através de *JavaScript*. A biblioteca seria o jQuery pelo maior suporte *online* numa área onde não existia muito conhecimento prévio. Uma vez que a aplicação seria baseada na *web*, os *smartphones* e *tablets* também teriam acesso ao mesmo portal, pelo que o desenvolvimento multiplataforma foi descartado.

4.4. LISTA DE REQUISITOS DA PLATAFORMA

Tendo em conta as opções de projecto tomadas, procedeu-se à elaboração da lista de requisitos a cumprir pela plataforma. Alguns requisitos foram adicionados no desenvolvimento da solução por *a posteriori* se terem revelado fundamentais. A lista definida inicialmente é algo extensa e é mostrada na Tabela 3.

Tabela 3 Requisitos do sistema.

REQ ID	Descrição do Requisito	Valor Limite	Método de verificação
FUN 1	Tem de receber pacotes de informação provenientes do WESENSE DAQ e WESENSE Energy.	N/A	Teste
FUN 2	Tem de ser possível configurar o WESENSE-DAQ e WESENSE-Energy através da página *web*.	N/A	Teste
FUN 3	Os dados de configuração do WESENSE-DAQ e WESENSE-Energy têm de ser enviados por TCP/IP.	N/A	Teste
FUN 4	Tem de ser feito o parsing dos pacotes TCP/IP.	N/A	Teste

FUN 5	Tem de guardar os dados recebidos na Base de dados.	N/A	Teste
FUN 6	Tem de permitir Multi-threading / Multi-processing para suportar múltiplos pedidos.	N/A	Teste
FUN 7	Deve ser implementado em servidor APACHE.	N/A	Desenho
FUN 8	Deve-se utilizar MySQL para acesso à base de dados.	N/A	Desenho
FUN 9	A página tem de ser desenvolvida em HTML, *Javascript*, CSS e PHP.	N/A	Desenho
FUN 10	O *web* server tem de funcionar na plataforma Windows.	N/A	Desenho
FUN 11	O módulo de recepção de pacotes do WESENSE tem de ser desenvolvido em C/C++.	N/A	Desenho
FUN 12	Interface gráfico no browser tem de estar disponível para qualquer utilizador	N/A	Teste
FUN 13	Interface de reconfiguração do WESENSE tem de estar disponível só para administrador do sistema.	N/A	Teste
FUN 14	A página *web* tem de permitir a visualização de gráficos de todos os tipos de dados provenientes dos equipamentos WESENSE.	N/A	Teste
FUN 15	O eixo de tempo do gráfico tem de ser parametrizável.	N/A	Teste
FUN 16	Tem de ser possível actualizar os gráficos em tempo real, sem necessidade de fazer refresh da página.	N/A	Teste
FUN 17	Pedidos por parte do cliente têm de ser processados pelo script PHP do lado do servidor.	N/A	Teste
FUN 18	Deve ser utilizada a *framework* Dojo/jQuery para o desenvolvimento de interface gráfica.	N/A	Desenho
FUN 19	O cabeçalho da página deve conter identificação do projecto e empresa.	N/A	Teste
FUN 20	Deve existir um menu horizontal com cinco páginas: Resumo, Gráficos, Vista detalhada, Reconfiguração e Exportação de dados.	N/A	Teste
FUN 21	Tem de ser possível selecionar quais os sensores WESENSE sobre os quais se pretende mostrar informação.	N/A	Teste

FUN 22	Tem de ser possível parametrizar os gráficos.	N/A	Teste
FUN 23	Tem de ser possível realizar o autofit do gráfico no eixo vertical.	N/A	Teste
FUN 24	Tem de ser possível alterar a grandeza observada(V, mA, Wh).	N/A	Teste
FUN 25	Deve ser possível a visualização individual do nível da bateria do módulo WESENSE (Gauges semelhantes aos do LabView).	0 - 3,6 V	Teste
FUN 26	Deve ser possível a visualização individual da temperatura interior do módulo WESENSE (Gauges semelhantes aos do LabView).	de -100º a +155º	Teste
FUN 27	Visualização individual de todas as características de cada WESENSE.	N/A	Teste
FUN 28	Deve-se utilizar as coordenadas GPS fornecidas pelo módulo para o posicionar num mapa na página.	N/A	Teste
FUN 29	Tem de ser apresentada a última hora de configuração realizada.	N/A	Teste
FUN 30	Tem de ser possível reconfigurar o intervalo de transmissão do WESENSE.	de 15 a 10080 min	Teste
FUN 31	O período de amostragem dos contadores tem de ser reconfigurável.	de 1 a 10080 min	Teste
FUN 32	Tem de ser possível exportar dados referentes a um ou vários sensores sob a forma de ficheiros de texto (CSV e XML).	N/A	Teste
FUN 33	Deve ser possível visualizar vários traçados de vários sensores no mesmo gráfico.	N/A	Teste
FUN 34	Tem de existir uma página onde se apresentam factores mais relevantes como os resumos de alarmes e medições mais recentes (home).	N/A	Teste
FUN 35	Tem de existir uma página onde se apresenta o gráfico de todos os WESENSE nessa rede e tabelas de manipulação do gráfico.	N/A	Teste
FUN 36	Tem de existir uma página onde é possível ver informação detalhada sobre cada WESENSE em particular (temperatura, tensão da bateria, posição geográfica)	N/A	Teste
FUN 37	Tem de existir uma página de reconfiguração do WESENSE com todos os parâmetros reconfiguráveis disponíveis.	N/A	Teste

FUN 38	Tem de exisitir uma página de exportação de informação sob o formato CSV e XML.	N/A	Teste
FUN 39	Deve ser possível configurar os módulos através de uma aplicação para *smartphone*, dedicada.	N/A	Teste

4.5. PROPOSTA DA SOLUÇÃO

Nesta secção pretende-se apresentar a arquitectura definida para o trabalho e especificar os seus blocos constituintes, que procuram dar resposta aos requisitos.

4.5.1. ARQUITECTURA DO SISTEMA

A arquitectura do sistema projectada para cumprir os requisitos é apresentada na Figura 49. O bloco sobre o qual o trabalho recai é o servidor de dados. Será este o bloco de análise ao longo desta secção, e o foco do trabalho realizado para a solução.

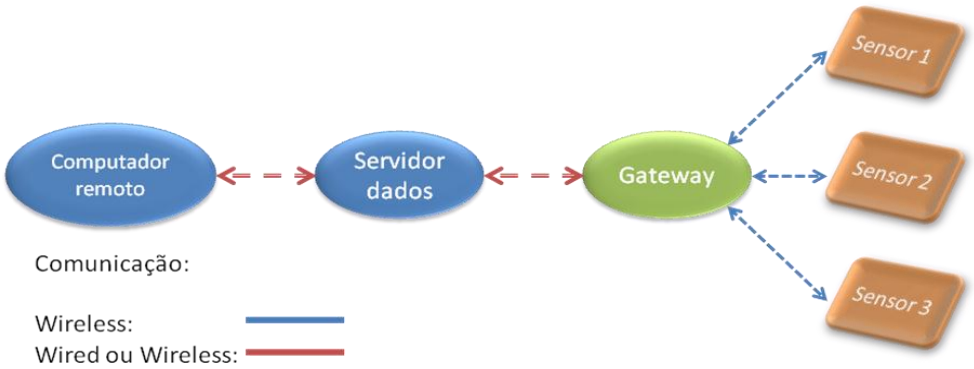


Figura 49 - Arquitectura do sistema.

O servidor de dados deverá realizar duas funções diferentes mas relacionadas. Por um lado, deve aceitar os pedidos de computadores remotos para visualizar a página *Web*, e por outro, deve permitir a recepção constante de pacotes TCP/IP enviados por parte dos módulos.

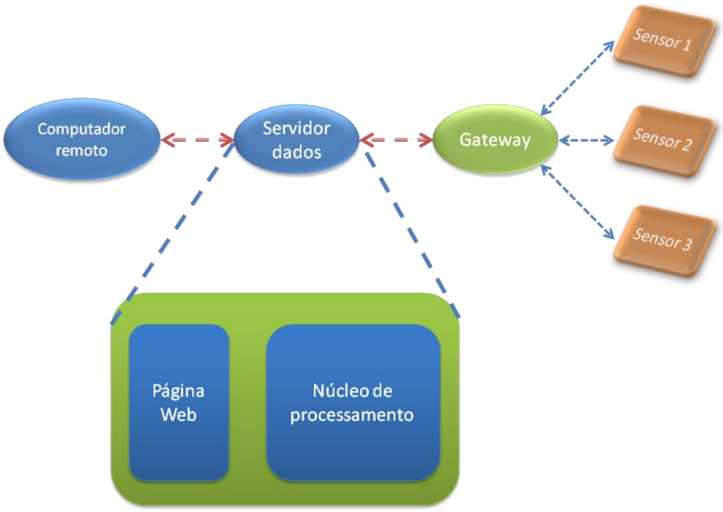


Figura 50 - Blocos fundamentais do servidor.

A página *Web* será desenvolvida utilizando as linguagens de programação mais comuns e a *framework* escolhida será o jQuery. O servidor irá assim esperar pelos pedidos HTTP para disponibilizar as páginas ao cliente. Os módulos WeSense disponibilizam a informação sob a forma de pacotes TCP/IP, a cada leitura efectuada. Desta forma, estes dados devem ser armazenados em algum local depois de descodificados pelo núcleo de processamento. Este núcleo deve então estar activo constantemente, à escuta destes pacotes de dados. Desta forma, de um ponto de vista de arquitectura de *software*, os blocos constituintes do sistema a desenvolver são os apresentados na Figura 51.

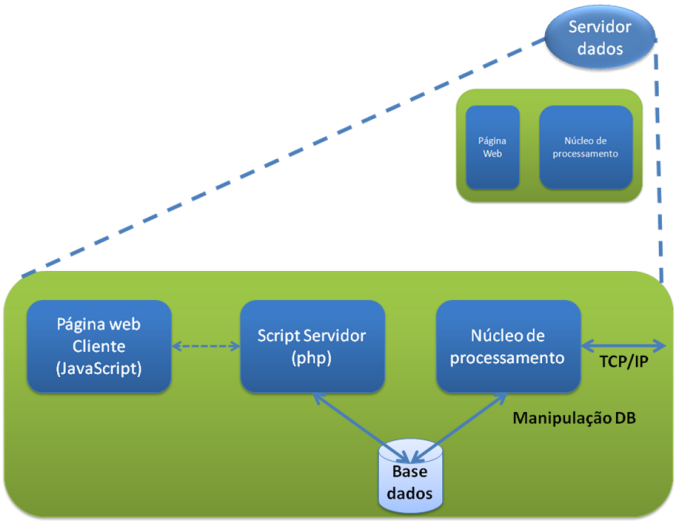


Figura 51 - Bloco funcional do servidor.

Existem dois processos principais: a operação de leitura de informação a partir da base de dados e disponibilização no *browser*, e a operação de reconfiguração dos módulos WeSense. O primeiro processo pode ser representado pelo seguinte diagrama de blocos apresentado na Figura 52.

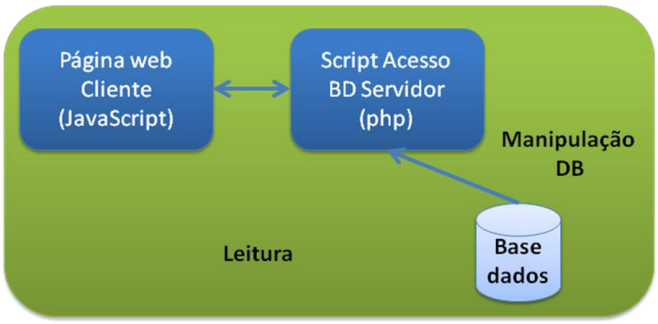


Figura 52 - Bloco referente à leitura/visualização de dados.

Conforme se pode observar, o acesso à base de dados é feito mediante o *script* PHP que acede directamente à base de dados. A escrita na base de dados é feita pelo núcleo de processamento, após recepção da informação por TCP/IP. A questão de pedidos múltiplos neste caso é suportada nativamente pelo *software* APACHE para gestão de execução de novos *scripts* PHP. O *software* de acesso à base de dados escolhido terá isto em conta.

O segundo processo pode ser representado pelo diagrama de blocos apresentado na Figura 53.

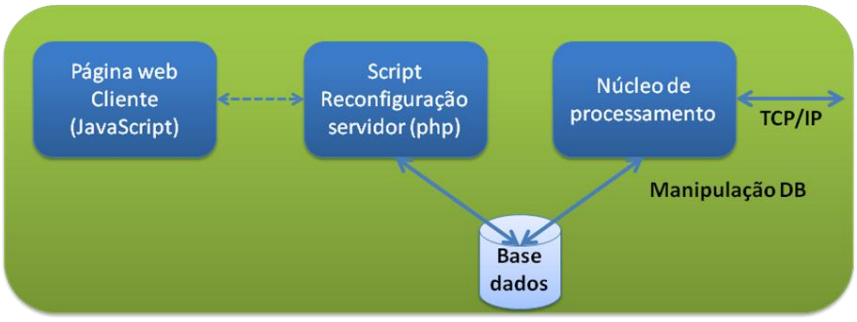


Figura 53 - Bloco referente à reconfiguração do módulo.

Neste caso, o pedido de reconfiguração de determinado sensor é aceite pelo *script* PHP no servidor. O *script* irá aceder à base de dados e alterar a tabela de configuração do dispositivo. Assim, da próxima vez que esse dispositivo transmitir nova informação, receberá a nova configuração do sistema. Adicionalmente, o sistema pode suportar para

além dos *browsers* como cliente, aplicações independentes desenvolvidas para vários sistemas operativos, móveis ou não, tornando-se assim numa plataforma completa que suporte múltiplos pedidos provenientes de múltiplos dispositivos.

4.5.2. NÚCLEO DE PROCESSAMENTO

O funcionamento do núcleo (*core*) visa três funções principais, que são:

- Acesso e manipulação da base de dados;
- Recepção contínua e descodificação dos pacotes TCP/IP recebidos dos módulos;
- Envio de resposta para o dispositivo cliente com as configurações;

Assim, o núcleo pode ser representado pela Figura 54, onde se apresentam os blocos principais a ser desenvolvidos: (*i*) o *listener* que deve estar constantemente activo à espera de ligações efectuadas, (*ii*) os *parsers*, que devem descodificar a mensagem recebida e (*iii*) o módulo de acesso à base de dados que permitirá a escrita e leitura de dados da BD.

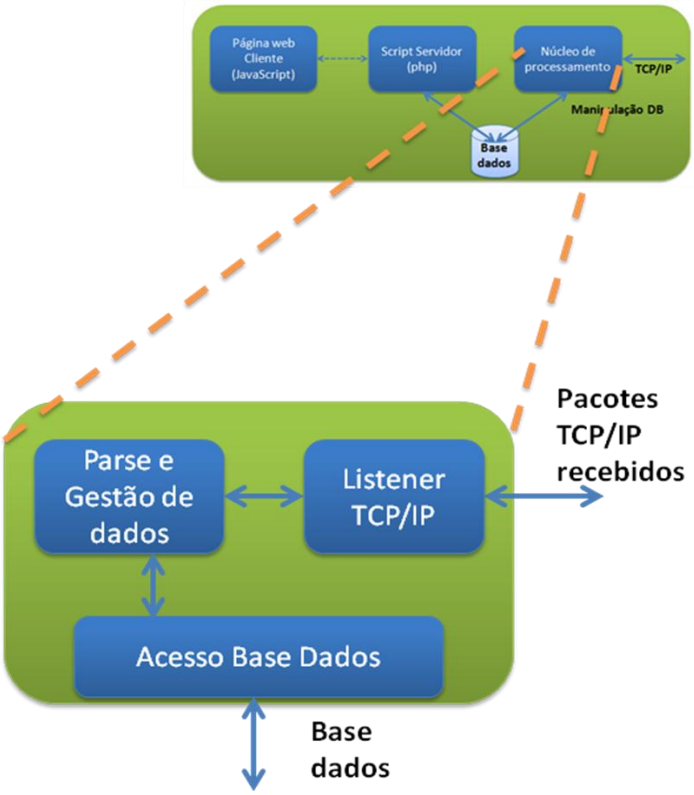


Figura 54 - Pormenor do núcleo de processamento.

5. DESCRIÇÃO DA SOLUÇÃO

Neste capítulo, pretendem-se clarificar quais as opções de projecto tomadas na fase inicial do trabalho, bem como apresentar a respectiva arquitectura de *software*. No decorrer do trabalho, tiveram de ser efectuadas várias alterações ao inicialmente previsto, sendo portanto, aqui apresentada apenas a versão final da arquitectura do sistema.

De acordo com o descrito no capítulo anterior, na secção de proposta da solução, a arquitectura escolhida para a plataforma foi a apresentada na Figura 55.

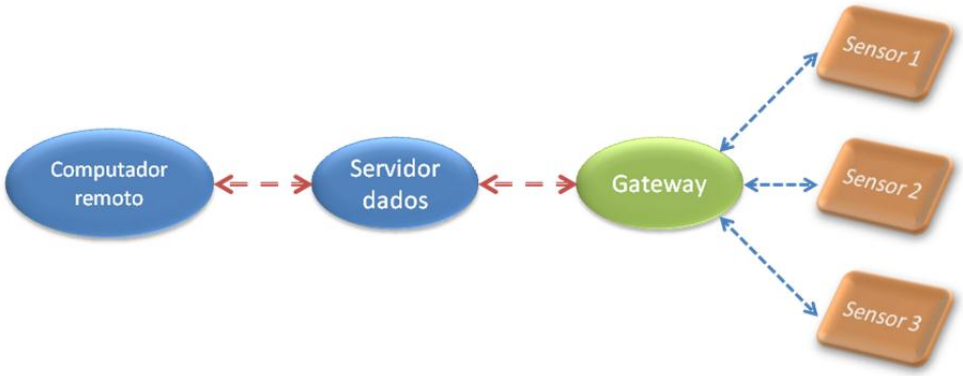


Figura 55 - Arquitectura definida.

Nesta secção, ir-se-á focar em detalhe cada um dos blocos e sub-blocos constituintes do sistema, abordando primeiro as questões de mais baixo nível, e subindo progressivamente para o *software* de mais alto nível, até à interface gráfica, conforme a Figura 56.

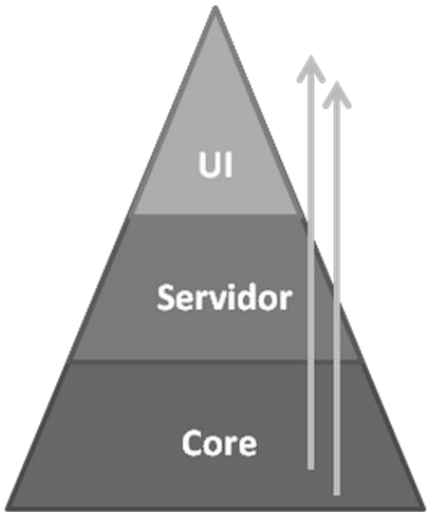


Figura 56 - Representação da abordagem do capítulo.

Resumidamente, as componentes que se pretenderam desenvolver foram as seguintes:

- Criação e organização das tabelas da base de dados;
- Aplicação de processamento e aquisição de dados;
- Aplicação *web* (UI);
- Aplicação Android para reconfiguração;

Cada um destes componentes é detalhado nas subsecções seguintes.

5.1. BASE DE DADOS

De forma a cumprir os objectivos anteriormente propostos, optou-se por armazenar toda a informação recebida numa base de dados, duma forma estruturada. Para esse efeito, utilizou-se o SGBD MySQL e foram criadas as tabelas apresentadas na Figura 57, bem como as respectivas relações. A chave primária entre as tabelas é a identificação única de cada módulo dada pelo endereço MAC (*Media Access Control)* de cada dispositivo ou pelo IMEI (*International Mobile Equipment Identity*). As tabelas criadas são as seguintes:

- *System_config*: armazena a configuração do sistema e a lista de dispositivos na rede;
- *Counters*: armazena os dados referentes a contadores dos módulos;

- *Data*: armazena os dados referentes às entradas analógicas e digitais;
- *System_status*: armazena dados do sistema de cada transmissão;
- *Alarms*: armazena os canais em alarme e a configuração do sistema nesse momento;

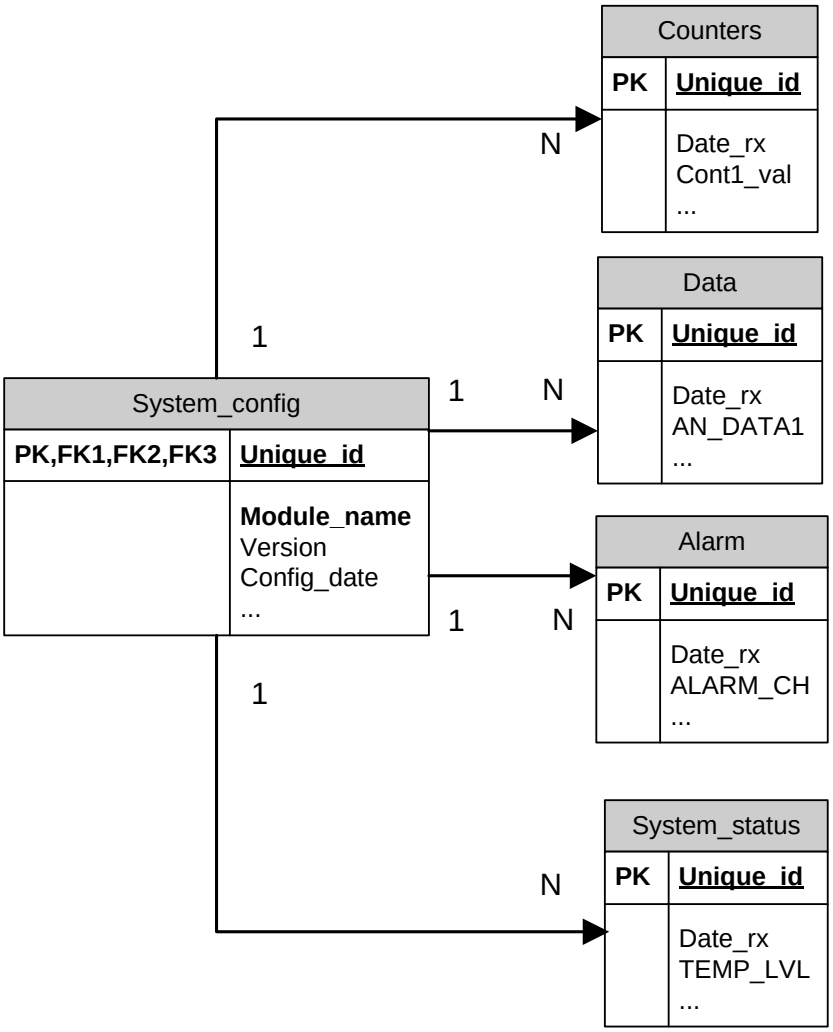


Figura 57 - Diagrama *Entidade-Relação*.

5.2. NÚCLEO DE PROCESSAMENTO (*CORE*)

O núcleo de processamento é o elemento responsável pela recepção dos pacotes de dados provenientes dos módulos, descodificação e armazenamento na base de dados, bem como do envio de resposta para o módulo com as novas configurações do dispositivo. Esquematicamente, pode ser definido como a junção de três sub-blocos conforme se apresenta na Figura 58 e que incluem o bloco *Listener* responsável pela comunicação do *core*, o bloco de *Parsing*, responsável pela interpretação dos dados e validação, e o bloco de acesso à base de dados, responsável pelo armazenamento ou leitura da informação na base de dados escolhida.

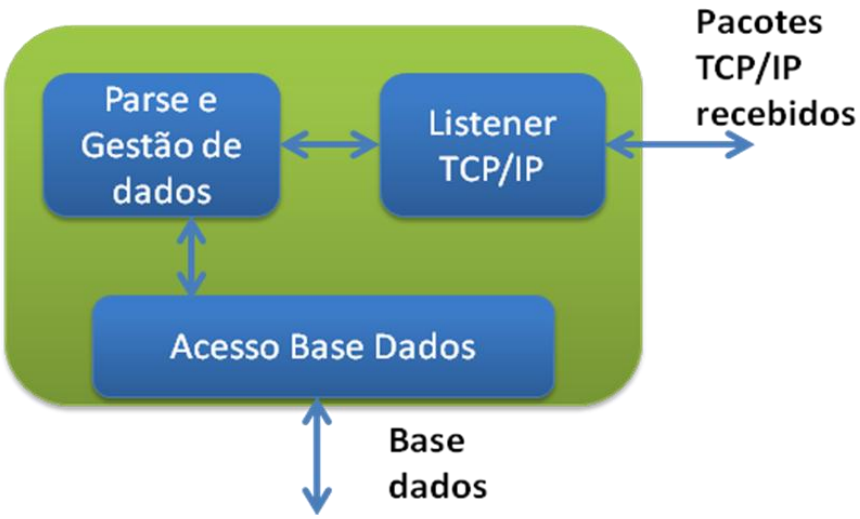


Figura 58 - Bloco do núcleo de processamento.

Este elemento do sistema é assim o intermediário entre o utilizador e os módulos distribuídos pela rede. Foi desenvolvido em C e C++ e comporta cerca de 6000 linhas de código, com o objectivo de correr em servidores Windows.

Tal como o diagrama da Figura 58 indica, os três sub-blocos foram desenvolvidos e testados em separado, inicialmente, numa fase de investigação e avaliação das capacidades e limitações que a solução pudesse oferecer. Algumas questões iniciais que se levantaram foram a rapidez de escrita de novas entradas e a capacidade de resposta a pedidos múltiplos de escrita na base de dados. Desta forma, foram efectuados alguns testes iniciais simples, que serviram de validação e aprovação da proposta, apresentados na tabela seguinte:

Tabela 4 Tempo médio de inserção de novas entradas na BD.

Nº entradas	Tempo de inserção (s)	Entradas por segundo
100000	9	11111
500000	41	12195
1000000	92	10869
4000000	359	11142
5000000	452	11061

Uma vez que o projecto decorreu em simultâneo com o desenvolvimento do *hardware* por elementos da equipa Evoleo, optou-se por desenvolver um cliente de teste para o núcleo de

processamento, por forma a simular uma comunicação proveniente de um dos módulos. Para tal, seguindo o protocolo definido de comunicação, criou-se um simples ficheiro de texto que seria utilizado pela aplicação cliente como o corpo da mensagem a transmitir. A Figura 59 apresenta um dos ficheiros de texto utilizados.

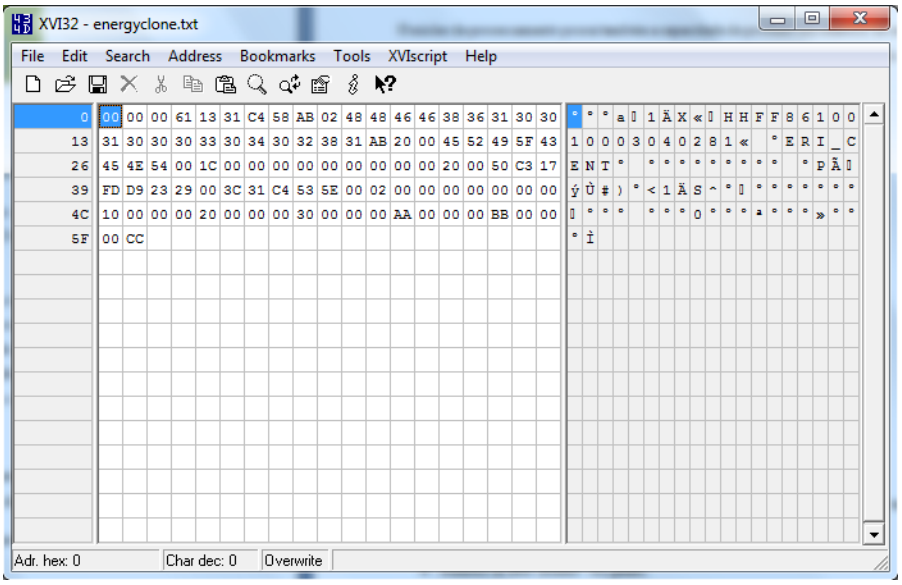


Figura 59 - Exemplo de ficheiro de texto criado para depuração.

Desta forma, conseguiu-se progredir no desenvolvimento sem estar sujeito à necessidade de *hardware* para efectuar os testes de depuração da descodificação. Esquematicamente, os testes iniciais efectuados poderiam ser representados pela Figura 60.

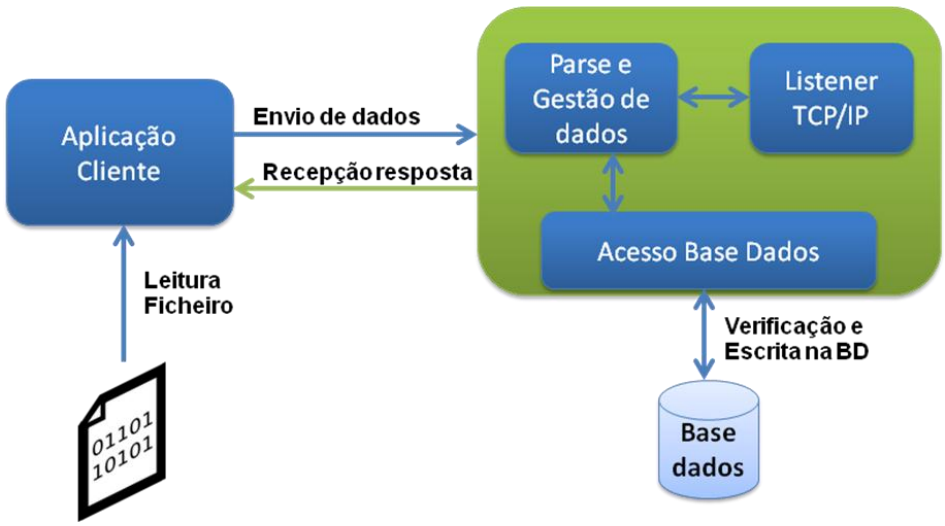


Figura 60 - Representação do sistema de depuração desenvolvido.

Nas versões iniciais do núcleo de processamento, a aplicação exibia o *output* da descodificação para uma consola, sendo possível comparar os valores descodificados com os valores inseridos, efectuando desta forma a depuração do sistema. A resposta era recebida pela aplicação cliente de teste, de forma a também ser possível validar a mensagem recebida gerada pelo núcleo de processamento. Devido à extensão do protocolo e às possíveis opções que permite, que por questões de confidencialidade da empresa não se pode revelar, estima-se que a fase de depuração tenha durado cerca de três quartos do tempo despendido nesta etapa.

Adicionalmente foi desenvolvido um *frontend* gráfico para a aplicação em C# conforme se mostra na Figura 61.

Figura 61 - *Frontend* núcleo de processamento.

Este *frontend* foi desenvolvido por forma a evidenciar o estado actual do sistema, para verificar continuamente os serviços necessários (serviços Apache, MySQL, e o próprio núcleo de processamento), avaliando a resposta a pedidos predefinidos por parte do núcleo de processamento, com o objectivo de validar não só se está em execução, como também para verificar se não ocorreu algum erro e está bloqueado.

O núcleo de processamento possui também a capacidade de produzir um relatório de erros ocorridos durante a execução, alertando para possíveis problemas e falhas, tendo sido útil numa fase de depuração do *software*.

As características do núcleo de processamento podem ser resumidas da seguinte forma:

- Validação, interpretação e armazenamento de dados na base de dados;
- Envio de nova configuração para cada dispositivo ligado na rede;
- *Dummy client* responsável por testar o serviço principal, em intervalos de tempo predefinidos, por forma a verificar se o sistema está a responder;
- Relançamento automático dos serviços do *core*, servidor Apache e MySQL;
- Relatório de erros (ficheiro *.txt*);

Versões com *output* directo para consola deste módulo têm sido usadas para depuração do *firmware* que tem vindo a ser desenvolvido pela empresa para os módulos de aquisição de dados, por forma a comparar rapidamente as saídas obtidas com as esperadas. Desta forma, a plataforma pode funcionar transitoriamente como um mecanismo de apoio à depuração, acelerando o processo de depuração do sistema, quer da parte de *hardware* e *firmware*, desenvolvido pela equipa na Evoleo Technologies bem como do *software* desenvolvido para a plataforma.

A título exemplificativo, a plataforma foi utilizada extensivamente durante o desenvolvimento de *firmware* do WeSense ENERGY para depuração do mesmo e também durante as fases de calibração de cada um dos módulos, comparando o valor apresentado na plataforma com os lidos em aparelhos de medição de referência.

O fluxograma principal do funcionamento do *core* apresenta-se na Figura 62.

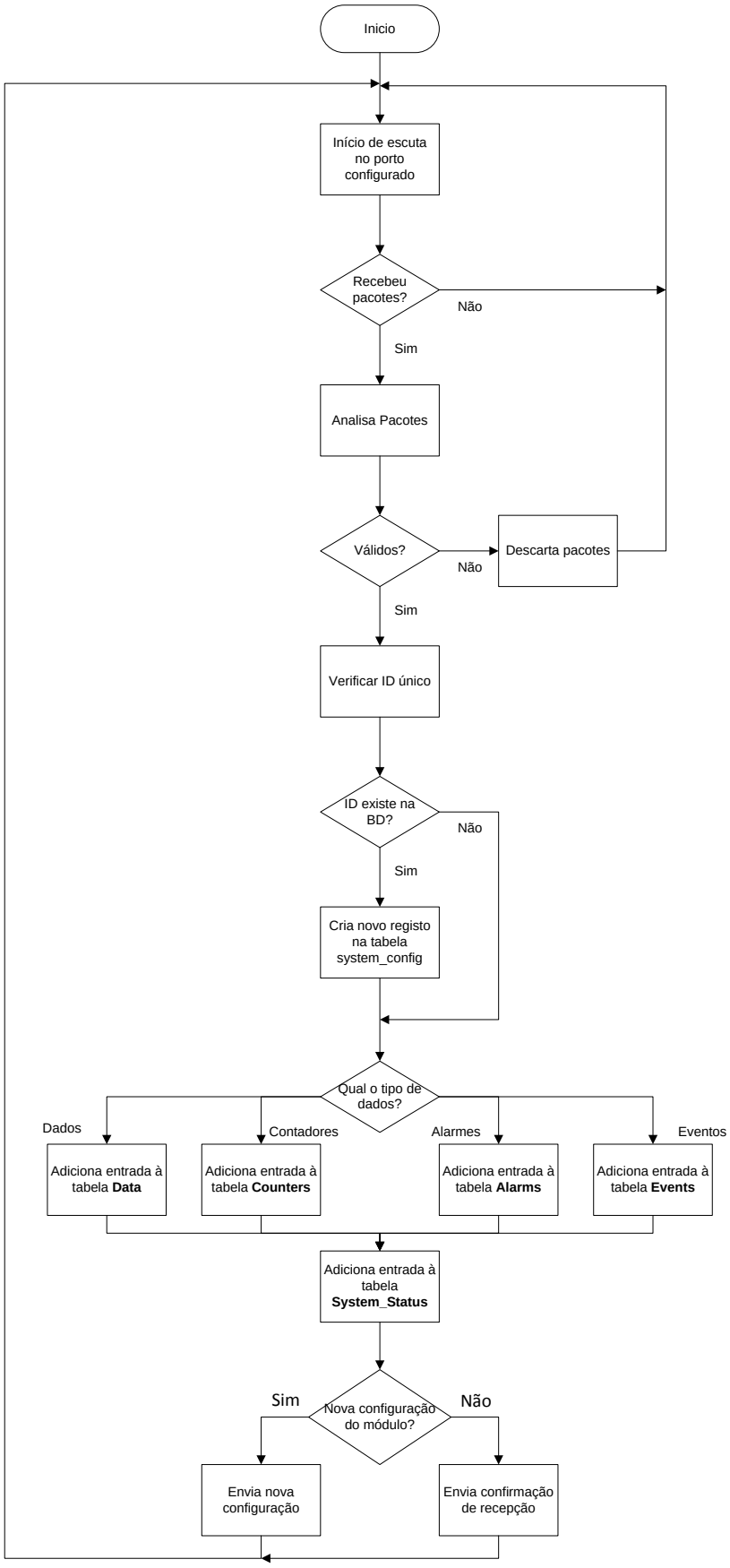


Figura 62 - Fluxograma do funcionamento do núcleo de processamento.

O fluxograma que controla o funcionamento dos serviços e processos necessários apresenta-se na Figura 63.

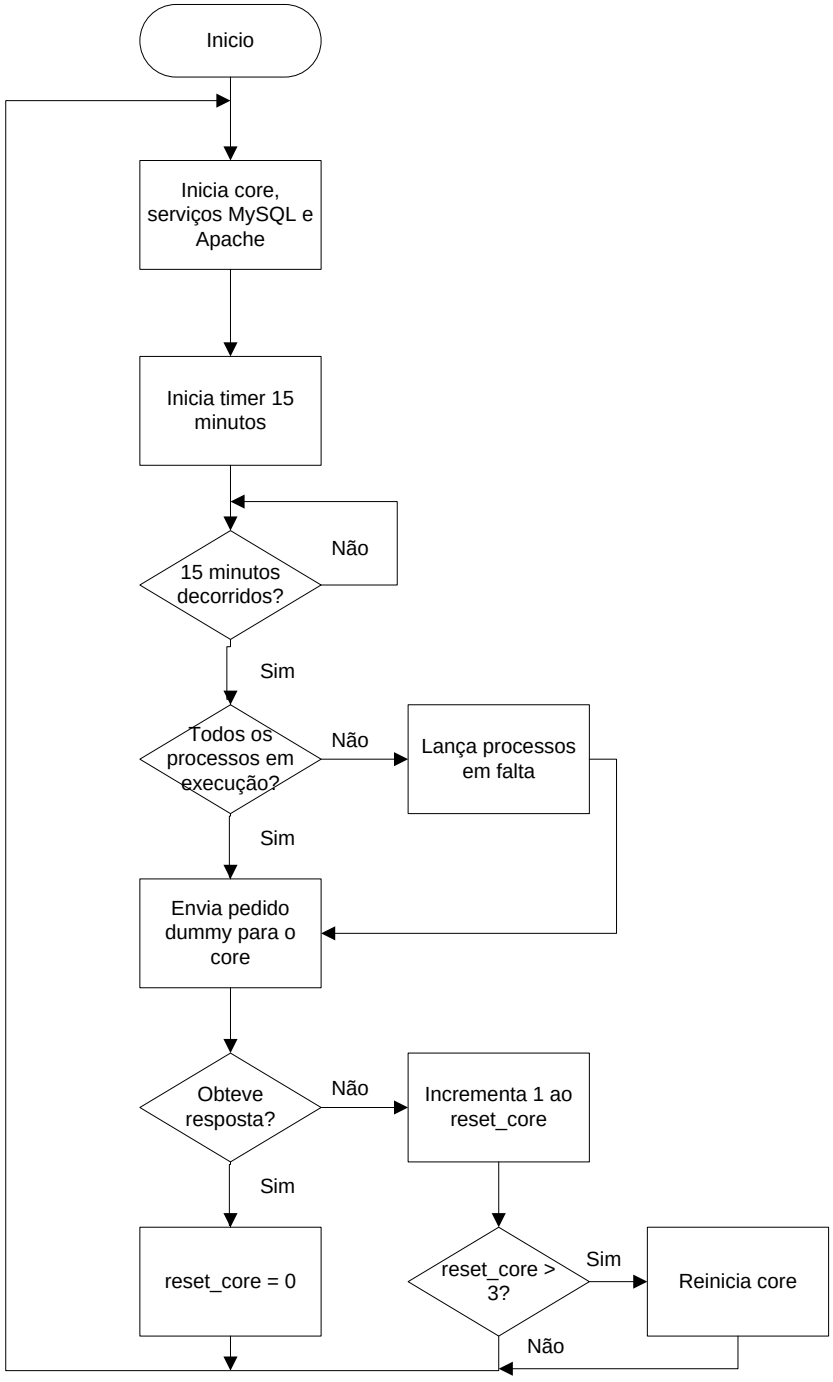


Figura 63 - Fluxograma referente ao *dummy client*.

5.3. SERVIDOR

O bloco referente ao servidor consiste essencialmente num conjunto de *scripts* PHP responsáveis por descodificarem pedidos efectuados ao servidor, obterem a informação pretendida da base de dados (ou inserir conforme for o caso), e enviá-la para o cliente. Esquematicamente podem-se representar algumas das funções principais desempenhadas pelo servidor como se indica na Figura 64.

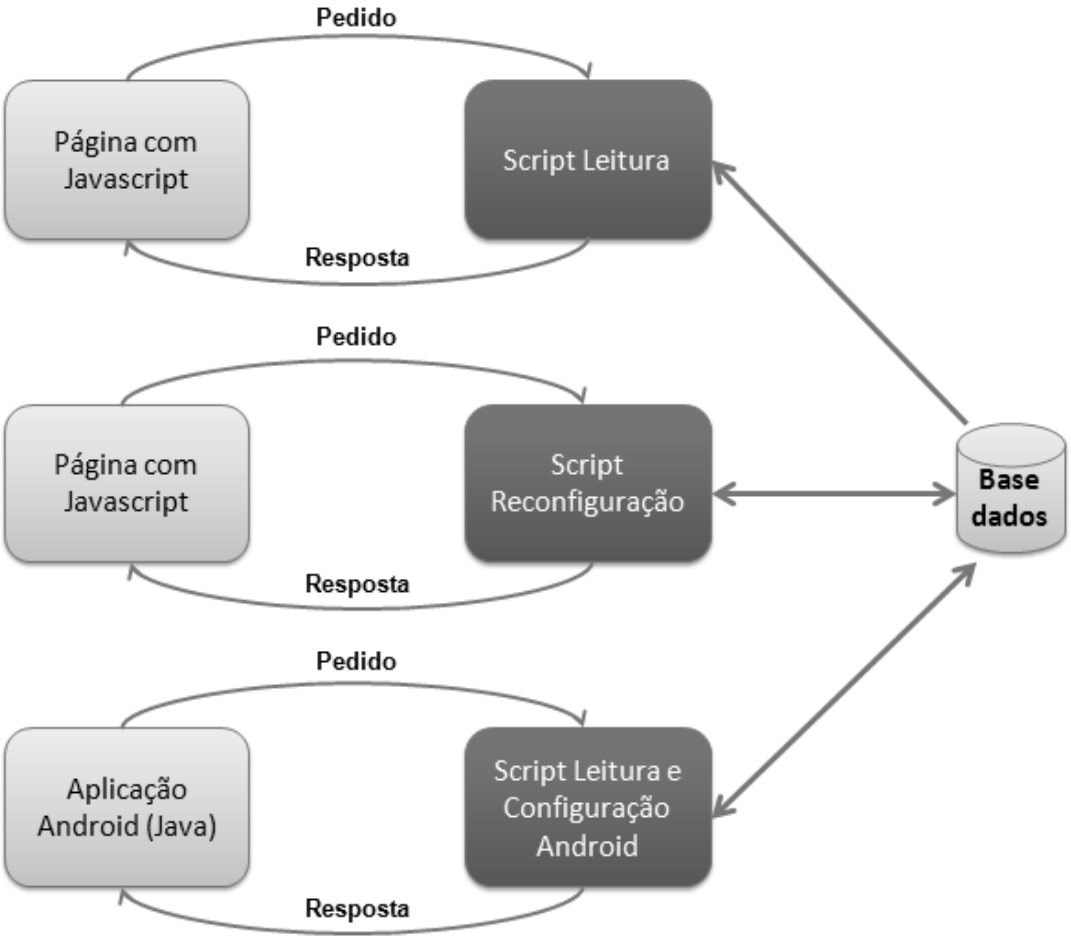


Figura 64 - Esquema representativo da função dos *scripts* PHP no servidor.

Os pedidos efectuam-se através do protocolo HTTP, com pedidos GET ao servidor, e a resposta codificada em JSON. Optou-se por JSON uma vez que a codificação e descodificação é consideravelmente mais simples recorrendo funções já implementadas em bibliotecas como o jQuery. Por exemplo, para efectuar o *parse* utilizando o jQuery, transformando a *string* de resposta num objecto *Javascript*, basta unicamente fazer:

```
var json_object = JSON.parse(data);
```

5.3.1. EXEMPLOS DOS PEDIDOS EFECTUADOS

Os pedidos GET provenientes das páginas são implementados através de AJAX, usando o *wrapper* $.GET da biblioteca jQuery. Estes pedidos são gerados dinamicamente, sem necessidade de *reload* da página, através de AJAX, obtendo informação específica do servidor estritamente quando necessário. Os dados recebidos são então convertidos para um objecto *Javascript* para uma maior facilidade de manipulação. Um exemplo da forma como este sistema de pedidos está implementado, pode ser verificado no extracto de código seguinte:

```
//AJAX GET com destino script.php e definida a
//função function(data){} de execução quando
//houver resposta.

$.get("script.php", function(data){

  //Converter string para objecto Javascript
  var json_object = JSON.parse(data);

  //Print da string recebida
  alert("String recebida: " + data);

  //print na consola do objecto
  console.log(json_object);

});
```

5.3.2. EXEMPLO DA ESTRUTURA DE RESPOSTA JSON

Após o pedido efectuado pelo cliente, o servidor tem como tarefa gerar a resposta, codificada em JSON. O código seguinte representa a estrutura de uma possível resposta do servidor.

```
{
    "0:4:a3:b5:2f:baE_ACTIVA": {
        "DEVICE_GROUP": "POWER_01_03",
        "DEVICE_GROUPDESCRIPTION": " EM_01",
        "DEVICE_ID": "0:4:a3:b5:2f:ba",
        "DEVICE_VAR": "E_ACTIVA",
        "DEVICE_TYPE": "2",
        "DEVICE_NAME": "EMETER03",
        "DEVICE_DATEMIN": "2012-04-01 00:00:00",
        "DEVICE_DATEMAX": "2012-10-18 17:43:11",
        "DEVICE_DATA": [
            {
                "data": "2012-09-05 12:42:38",
                "valor": "10"
```

```
            },
            {
                "data": "2012-09-05 12:43:38",
                "valor": "10"
            }
        ]
    },
    "0:4:a3:b5:2f:79E_ACTIVA": {
        "DEVICE_GROUP": "POWER_01_03",
        "DEVICE_GROUPDESCRIPTION": "EM_01",
        "DEVICE_ID": "0:4:a3:b5:2f:79",
        "DEVICE_VAR": "E_ACTIVA",
        "DEVICE_TYPE": "2",
        "DEVICE_NAME": "EMETER01",
        "DEVICE_DATEMIN": "2012-04-01 00:00:00",
        "DEVICE_DATEMAX": "2012-10-18 17:43:11",
        "DEVICE_DATA": [
            {
                "data": "2012-09-05 12:29:36",
                "valor": "30"
            },
            {
                "data": "2012-09-05 12:30:36",
                "valor": "30"
            }
        ]
    }

}
```


5.3.3. FLUXOGRAMA

O fluxograma relativo a estes *scripts* é relativamente simples, mas ainda assim implica a descodificação do pedido, o acesso a base de dados e recolha dos valores pretendidos, e a codificação da resposta em JSON para o cliente.

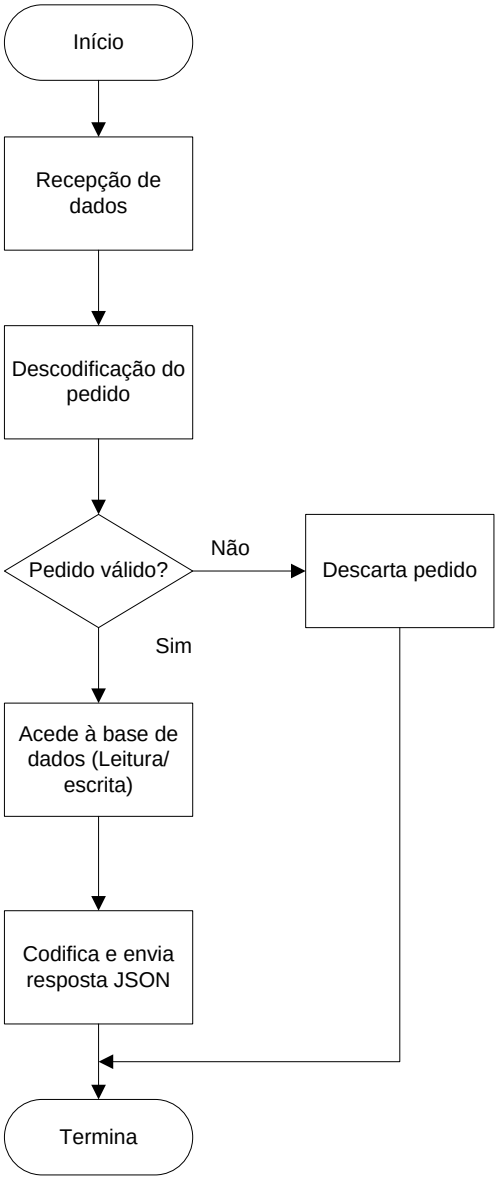


Figura 65 - Fluxograma genérico dos *scripts* PHP desenvolvidos.

5.4. APLICAÇÃO *WEB* (UI)

Nas subsecções anteriores identificamos os componentes responsáveis pela disponibilização dos dados provenientes dos sensores, e que constituem o mecanismo fundamental por detrás da interface gráfica. Nesta secção, pretende-se demonstrar a interface da plataforma desenvolvida, tirando proveito da informação recolhida pelos sensores armazenada na base de dados. Na Figura 66 é apresentado o mapa do portal desenvolvido.

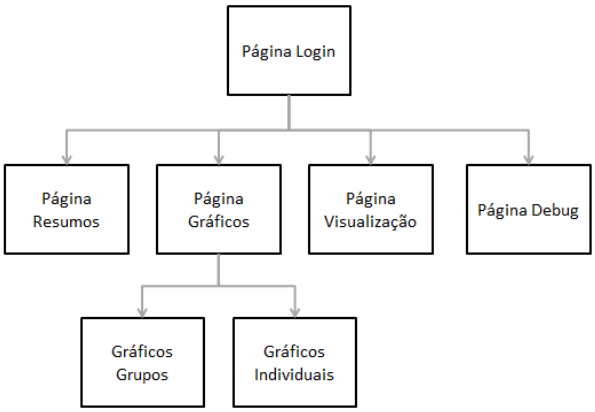


Figura 66 - Mapa do portal desenvolvido.

5.4.1. PÁGINA RESUMOS

Após o *login* no portal, a página de resumos é apresentada. As ideias fundamentais desta página seriam as seguintes:

- Listar últimos alarmes na instalação;
- Listar eventos relevantes (adição de dispositivos à rede, falhas e erros detectados);
- Apresentar lista de dispositivos activos;

A apresentação desta informação de uma forma interactiva, leve, visualmente atractiva e sem ter de recorrer a tabelas excessivas foi uma preocupação dominante desde início desta parte do trabalho. Pretendia-se assim que a página inicial oferecesse informação principal e portanto sem grande detalhe.

Optou-se então por utilizar um painel temporal que permite visualizar todos os eventos definidos com algum detalhe, navegando na linha de tempo. Assim, a estrutura da página ficou definida como a apresentada na Figura 67.

Figura 67 - Estrutura da página de resumos.

Os campos *Dispositivos* e *Grupos* presentes na Figura 67 dizem respeito às listas de dispositivos e de grupos presentes na instalação, oferecendo informação sobre quais os dispositivos *online* e *offline*, nome e tipo (DAQ ou ENERGY). Apresentam-se também os grupos de dispositivos criados pelo administrador da plataforma. O aspecto da página de resumos é apresentado na Figura 68.

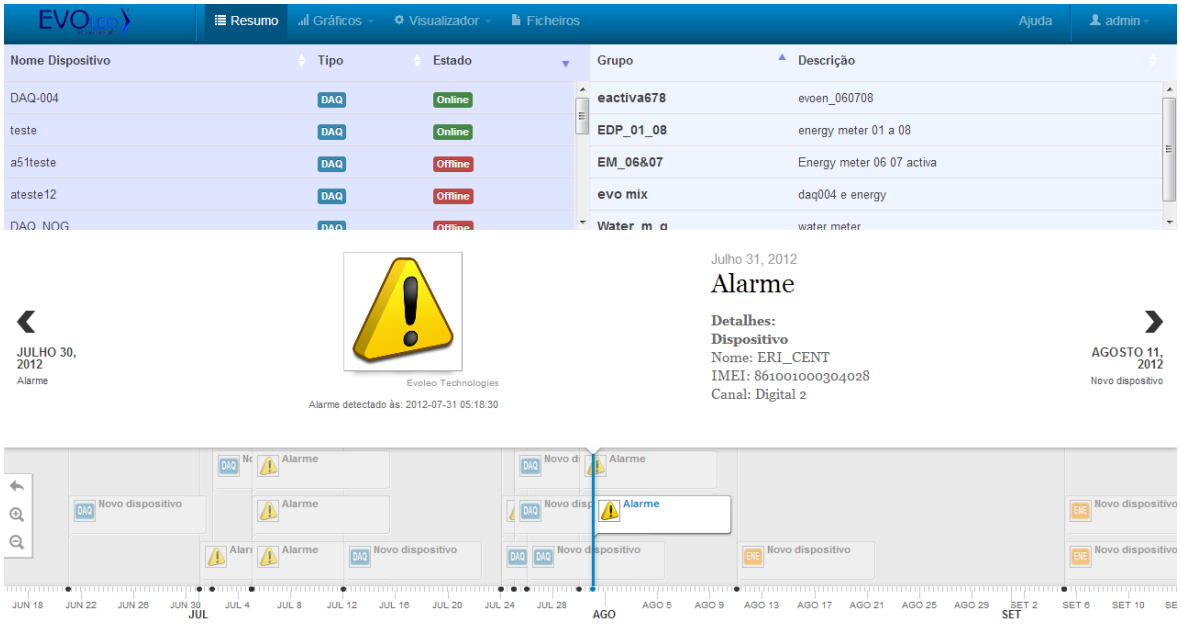


Figura 68 - Página de resumos.

5.4.2. PÁGINA DE VISUALIZAÇÃO

Cada dispositivo ligado à rede possui um determinado número de parâmetros que podem ser remotamente configurados. A vantagem principal prende-se com a simplicidade de utilização dado que ao invés de ser necessário deslocar-se até ao local onde o dispositivo está instalado para o reprogramar com ferramentas adicionais, basta antes ligar um portátil, ou qualquer outro dispositivo com um *browser* à rede para se alterar algum parâmetro pretendido.

Tal como nas anteriores, esta página também teve várias versões até chegar ao *design* e funcionalidade final conforme se apresenta na Figura 69.

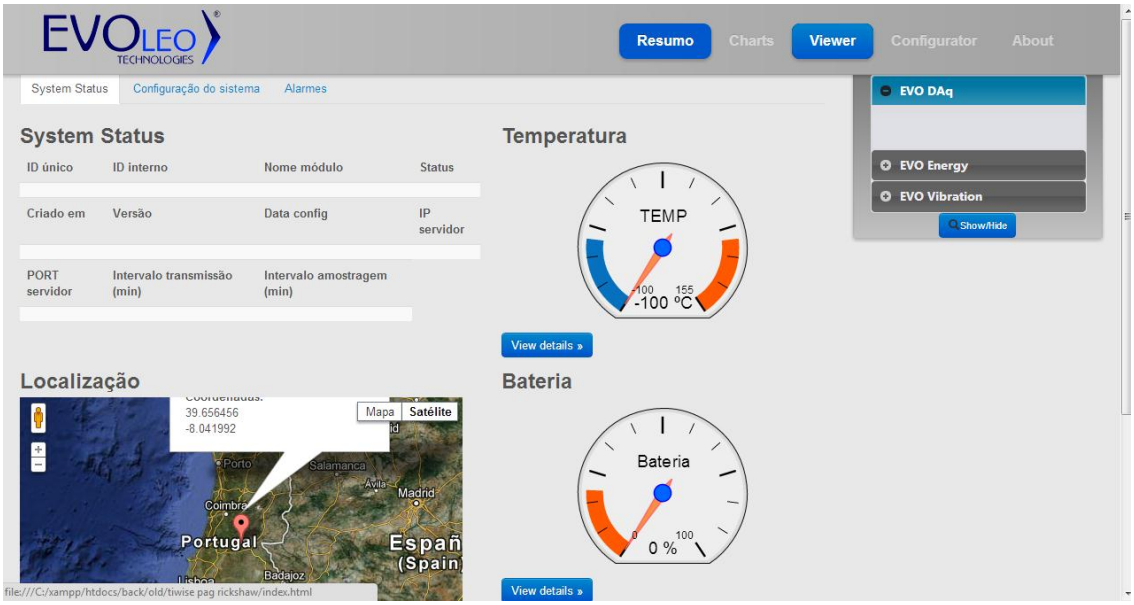


Figura 69 - Protótipo inicial da página de configuração.

Numa fase inicial, as páginas de *configuração* e *visualização* estavam separadas, tendo mais tarde se optado por apresentar na mesma página as mesmas informações, uma vez que a página de *configuração* iria apresentar grande parte da informação que a página de *visualização* tinha. Assim, reuniu-se tudo na mesma página, simplificando do ponto de vista de utilização e de implementação.

Visualmente, os elementos que se quiseram transmitir desde cedo foram as configurações actuais do dispositivo de uma forma simples e visualmente atractiva com *gauges* e mapas como na Figura 69. Podemos resumir os objectivos desta página na seguinte lista:

- Visualizar configuração e reconfigurar remotamente cada dispositivo;
- Lista de alarmes e eventos disparados, referente a cada dispositivo;
- Identificação fácil do tipo de cada dispositivo;
- Pesquisa por nome, tipo ou descrição de dispositivo;
- Edição de campos rápida, sem necessidade de *reloads* da página;
- Apesar da quantidade de parâmetros, manter a navegação visualmente simples e rápida;
- Disponibilizar informação relevante desta secção de forma visualmente fácil de interpretar (gráficos para temperatura, mapas com localização exacta do módulo, etc.);

Depois de cumpridos estes objectivos obteve-se a versão final da página apresentada na Figura 70.

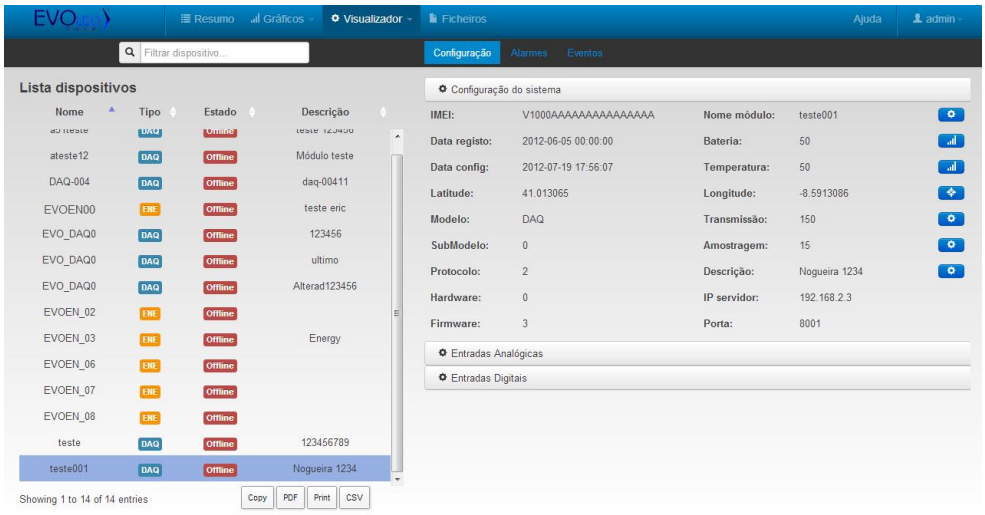


Figura 70 - Aspecto geral da página de visualização dos dispositivos.

A estrutura da página pode ser divida em 3 áreas principais que seguidamente se analisam:

Sub-barra de navegação

Figura 71 - Sub-barra de navegação.

Oferece a navegação entre os três *tabs* principais de navegação, *Configuração* do dispositivo (visível na Figura 70), *Alarmes*, e *Eventos* registados por cada dispositivo. Adicionalmente a caixa de texto presente permite filtrar os elementos da lista de dispositivos pelo nome introduzido.

Lista de dispositivos

A lista de dispositivos apresenta todos os módulos ligados ao servidor, identificados por nome, tipo, estado e descrição. Os campos de tipo e estado presentes na tabela disponibilizam a informação sobre a forma de pequenos *labels* que ao fazer *hover*, disponibilizam informação adicional conforme se pode ver na Figura 72.

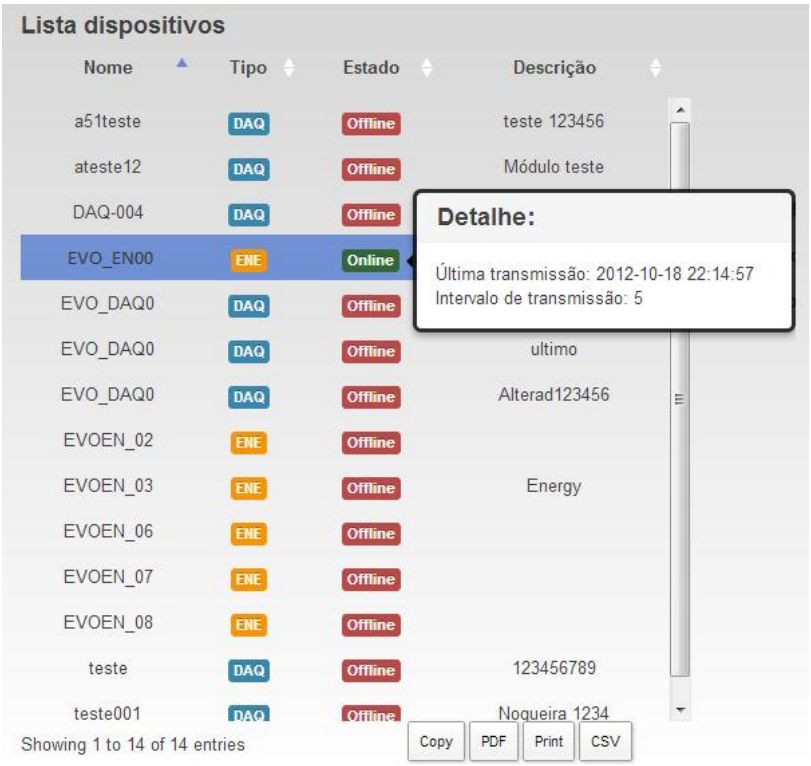


Figura 72 - Pormenor da Lista de Dispositivos.

Relativamente ao estado do dispositivo, é apresentada informação sobre:

- A hora da última transmissão;
- O intervalo de transmissão configurado do dispositivo;

Na *label* do tipo de dispositivo a informação apresentada em *hover* é a seguinte:

- Modelo do dispositivo (Energy, DAQ, Vibrations...);
- Submodelo do dispositivo;
- Protocolo de comunicação;
- Versão de *Hardware;*
- Versão de *Firmware;*

O campo de descrição permite identificar mais facilmente o dispositivo pretendido do ponto de vista do utilizador, uma vez que é possível editá-lo. Toda a tabela pode ser filtrada através do campo de texto na sub-barra de navegação, conforme a Figura 73 onde se apresentam só os dispositivos do tipo DAQ.

É também possível exportar a lista de dispositivos conforme os botões indicados na Figura 73, para os formatos PDF e CSV.

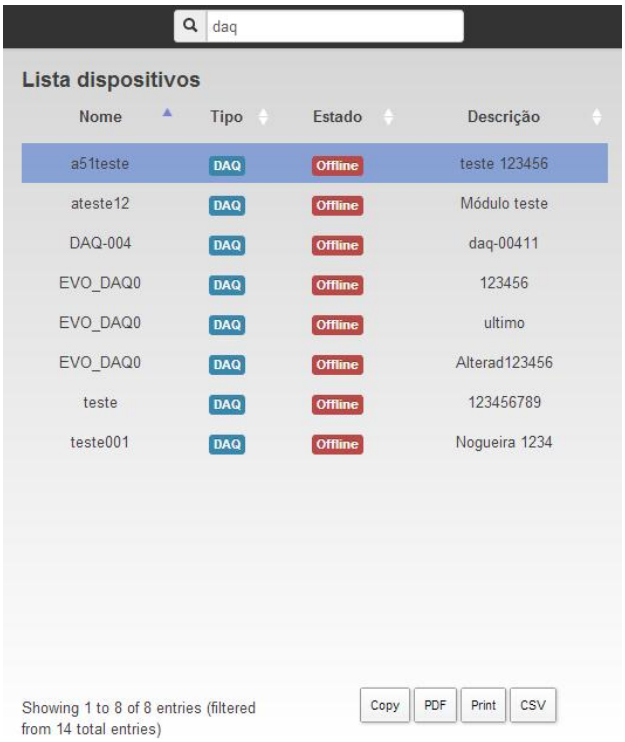


Figura 73 - Resultados da tabela filtrados por tipo de dispositivo.

O botão *print* permite eliminar todos os elementos da página, deixando só a tabela para ser impressa sem o *design* adicional, conforme a Figura 74.

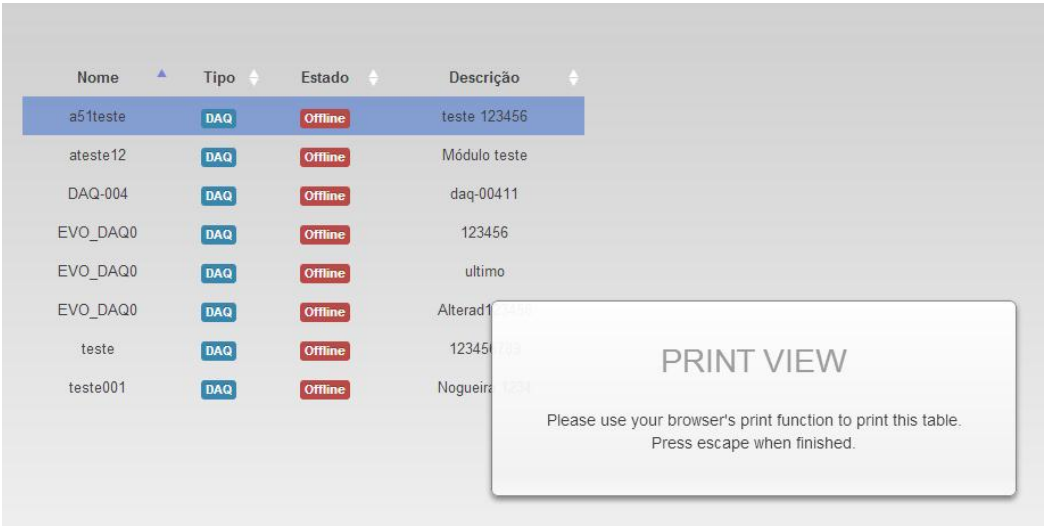


Figura 74 - Ecrã de *print* da lista de dispositivos.

Ao clicar num dos elementos da lista de dispositivos, é carregada a informação sobre o dispositivo escolhido, deixando um *feedback* visual de selecção.

Informação de dispositivos

Nesta secção da página são apresentadas as informações relativas a cada dispositivo da rede, conforme apresentado na Figura 75.

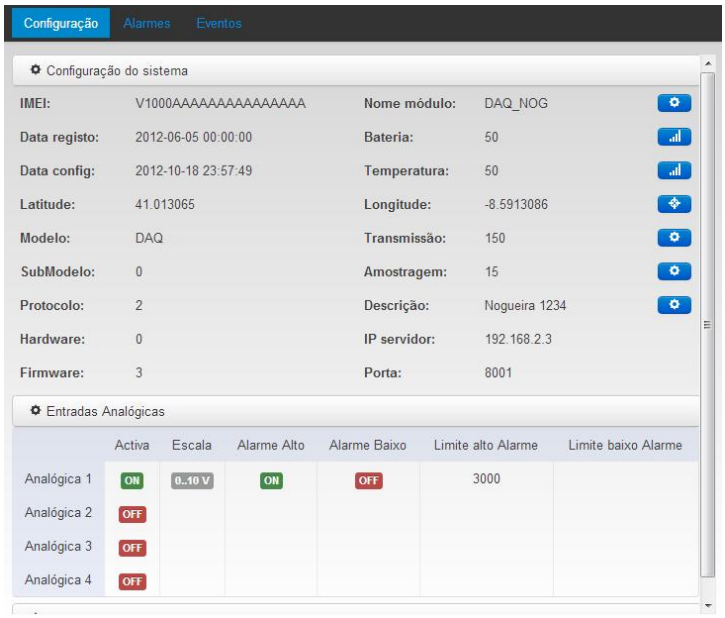


Figura 75 - Campo de dados do dispositivo.

Consoante o tipo do dispositivo, são mostrados os campos respectivos (e.g. no caso do módulo WeSense ENERGY, não são apresentados os *tabs* de *Alarmes* e *Eventos*). No caso do módulo DAQ, a navegação é feita através de três *tabs* principais, apresentados em detalhe nas secções seguintes:

Configuração

Este *tab* apresenta a configuração de sistema, das entradas analógicas e digitais. Os campos de configuração do sistema que podem ser alterados ou verificados com mais detalhe identificam-se pelo botão de edição presente. A Figura 76 apresenta o *popover* lançado ao clicar no botão de edição.

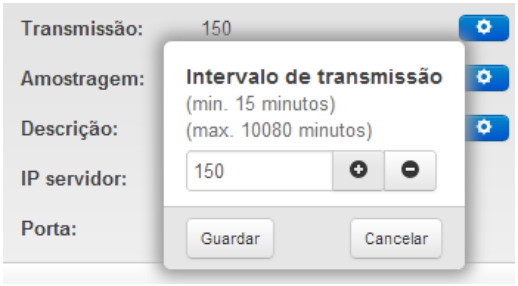


Figura 76 - *Popover* do campo de edição do intervalo de transmissão.

Os campos relativos à configuração do sistema que podem ser reconfigurados no caso do módulo DAQ são os seguintes:

- Nome;
- Intervalo de transmissão;
- Intervalo de Amostragem;
- Descrição;

Os restantes botões apresentados permitem visualizar com mais detalhe as informações apresentadas, como é o caso dos apresentados na Figura 77.

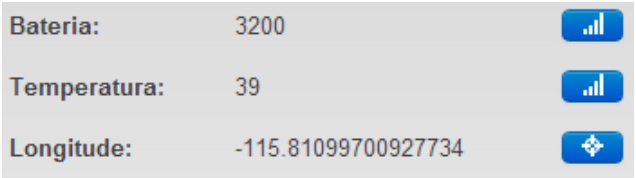


Figura 77 - Campos com opção de vista detalhada.

O botão presente à direita do campo de longitude apresenta, quando pressionado, um mapa interactivo com a localização do dispositivo, caso este esteja instalado numa zona com cobertura GPS. Através de *reverse Geocoding* possibilitado pela API do Google Maps, obtém-se a morada mais próxima das coordenadas do dispositivo conforme indicado na Figura 78.

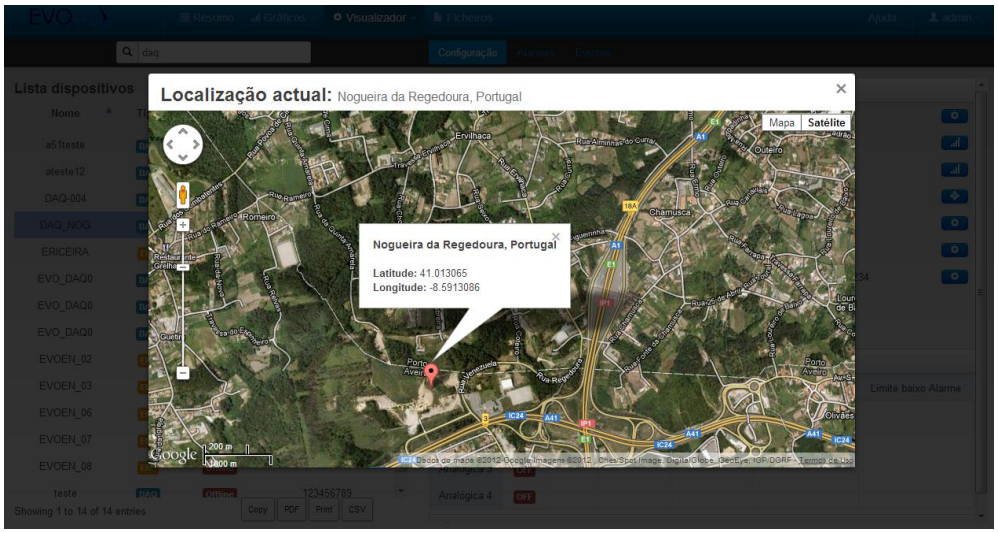


Figura 78 - *Popup* com localização do dispositivo de teste no mapa.

Os botões presentes à direita dos campos *Temperatura* e *Bateria* apresentam um *popup* semelhante ao anterior, onde se pode verificar através de um gráfico a evolução da temperatura e percentagem da bateria medida pelos sensores internos de cada módulo. A Figura 79 apresenta o *popup* gerado antes de estar definido o intervalo de tempo pretendido.

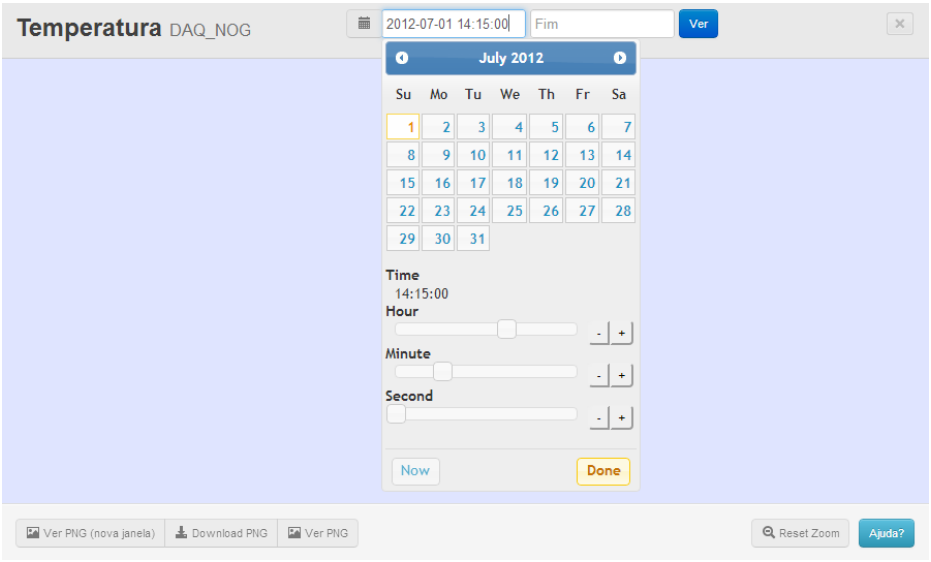


Figura 79 - Aspecto geral do *Popup* ao definir o intervalo de tempo de visualização.

Algumas das funcionalidades e características presentes no *popup* são:

- Definição do intervalo de tempo através de calendário;
- Gráficos interactivos com capacidade de *zoom*, animações de transição, vista detalhada;
- Biblioteca gráfica jqPlot;
- Exportação do gráfico em formato de imagem (PNG) com várias opções de visualização;

O aspecto gráfico do *popup* divide-se em três secções principais tradicionais:

- *Header*, com informações sobre o dispositivo e campos para edição do intervalo de tempo;
- *Body*, área principal do *popup*, onde se apresenta o gráfico interactivo gerado e possível PNG para exportação;
- *Footer*, com os comandos principais de exportação do gráfico e controlo de *zoom*.

Após definido o intervalo de tempo, o gráfico referente à temperatura é apresentado, ficando o *popup* com o aspecto semelhante ao da Figura 80. Os dados apresentados dizem respeito a uma fase de testes de *hardware* e *firmware*, estando-se a implementar o sensor de temperatura do sistema.

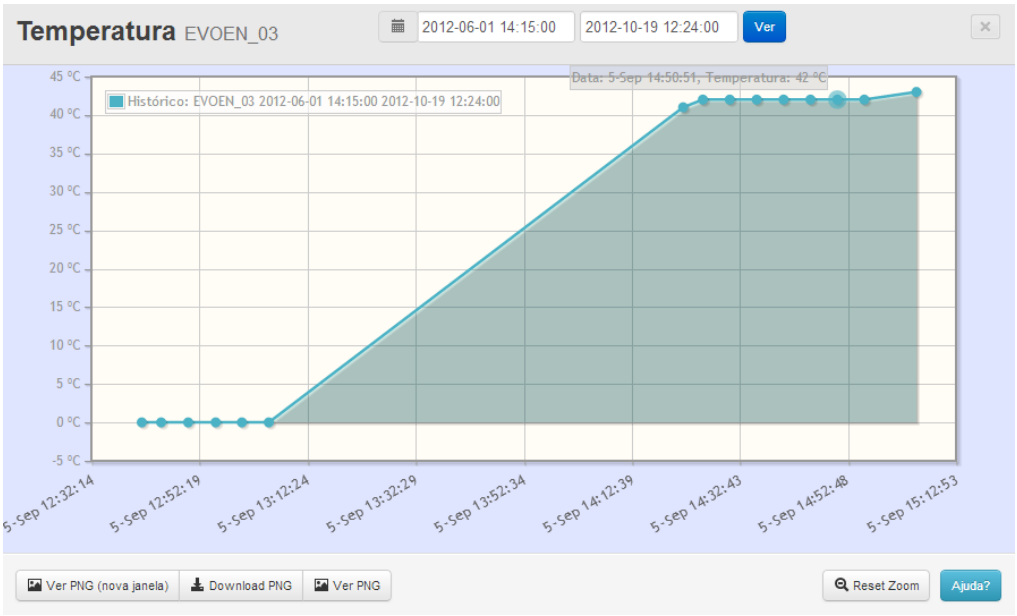


Figura 80 - Gráfico da temperatura registada pelo sensor interno do módulo.

Os controlos de exportação no rodapé do *popup* permitem várias hipóteses de exportação da imagem uma vez que nem todos são compatíveis com a variedade de *browsers* disponíveis actualmente. A Figura 81 apresenta os controlos de exportação referidos.

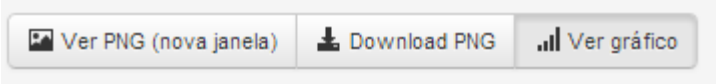


Figura 81 - Controlos de exportação do gráfico.

Conforme apresentado na Figura 81, os botões de exportação são os seguintes:

- Ver PNG – abre em nova janela/*tab* o PNG gerado do gráfico;
- Download PNG – *download* da imagem;
- Ver gráfico – Apresenta a imagem no corpo do *popup*, deslizando o gráfico interactivo para fora da visualização (estado do botão *toogle*, conforme apresentado na Figura 81).

A Figura 82 ilustra o *popup* de temperatura com a imagem do gráfico no corpo do *popup*.

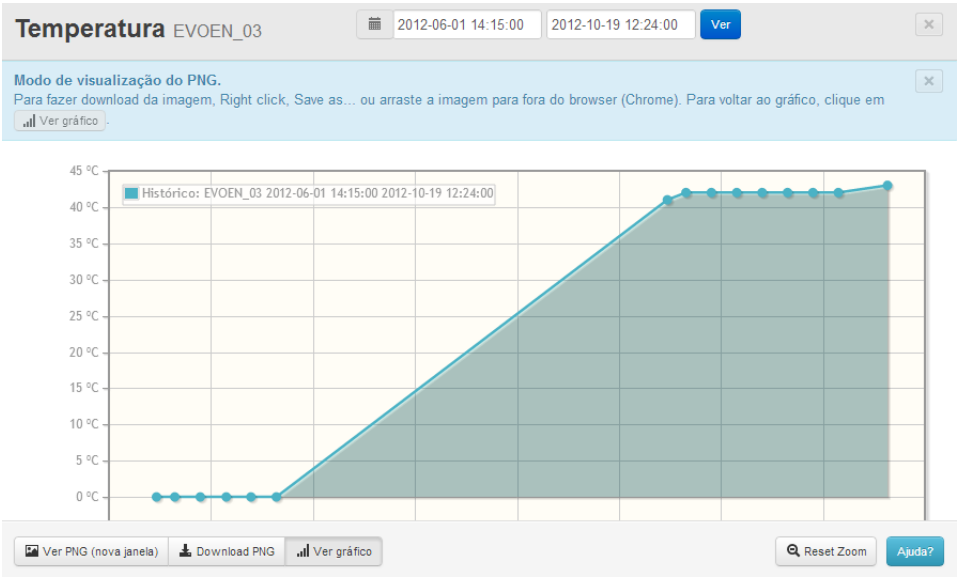


Figura 82 - Imagem do gráfico exportado para PNG no corpo do *popup*.

As tabelas de *Entradas Analógicas* e *Entradas Digitais* fornecem a informação mais relevante sobre cada uma das entradas, conforme apresentado na Figura 83.

Figura 83 - Tabela de configuração das entradas analógicas.

Dependendo do tipo de entrada, podem ser configurados diferentes parâmetros. As entradas analógicas 1 e 2 só podem ser configuradas como entradas em tensão e as entradas analógicas 3 e 4 só podem ser configuradas como entradas em corrente, podendo neste caso seleccionar-se a escala de 0 a 10 mA ou a de 4 a 20 mA. Os restantes parâmetros são comuns a ambas.

Ao clicar numa das filas apresenta-se um *popup* de configuração desse mesmo canal, semelhante ao apresentado na Figura 84.

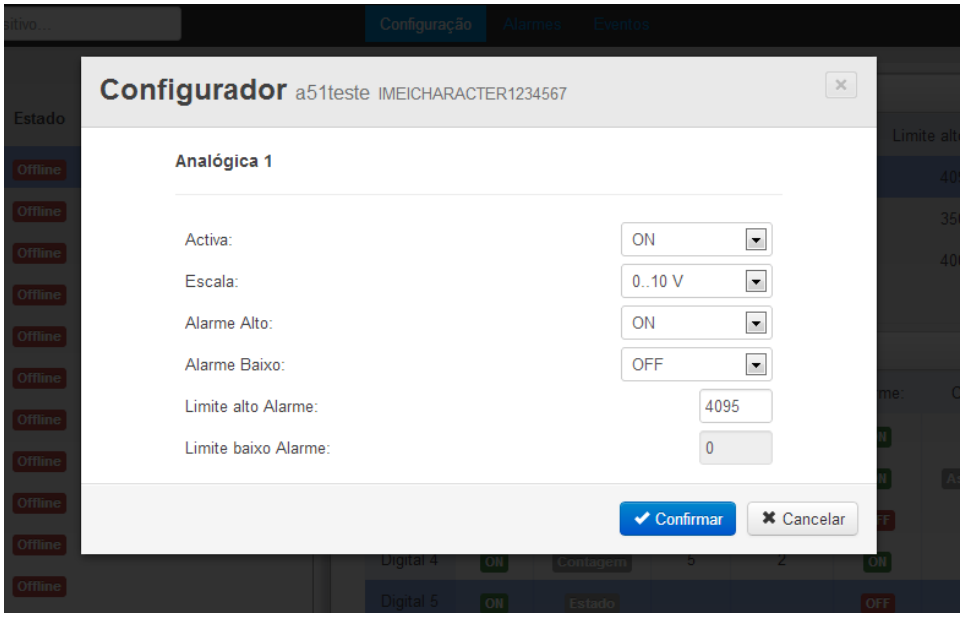


Figura 84 - *Popup* de configuração de entradas analógicas.

Nesta janela pode-se configurar:

- Entrada activada ou desactivada;
- Escala (0 a 10 V, 0 a 20 mA, 4 a 20 mA);
- Alarmes activos (alarme superior e inferior);
- Valor do alarme (superior ou inferior);

Quer do ponto de vista de submissão dos dados na página ou do ponto de vista da recepção dos dados de configuração do lado do servidor, existe validação dos parâmetros introduzidos. A Figura 85 apresenta uma das mensagens de erro geradas ao serem detectados parâmetros errados.

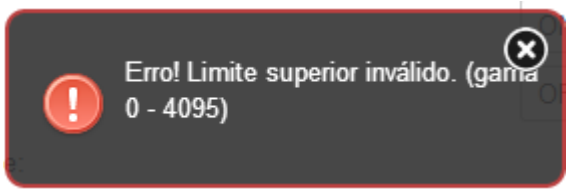


Figura 85 - Mensagem de erro ao submeter dados.

Note-se que o valor introduzido é forçado ao intervalo presente na Figura 85 de 0 a 4095, uma vez que a conversão da tensão é feita por um conversor analógico-digital de 12 *bits* nos módulos WeSense DAQ, que se determinou que oferecia resolução suficiente para as aplicações pretendidas.

A tabela de configuração dos canais digitais apresenta funcionalidade e aspecto semelhante ao da tabela das entradas analógicas, embora agora apareçam parâmetros específicos a estas entradas.

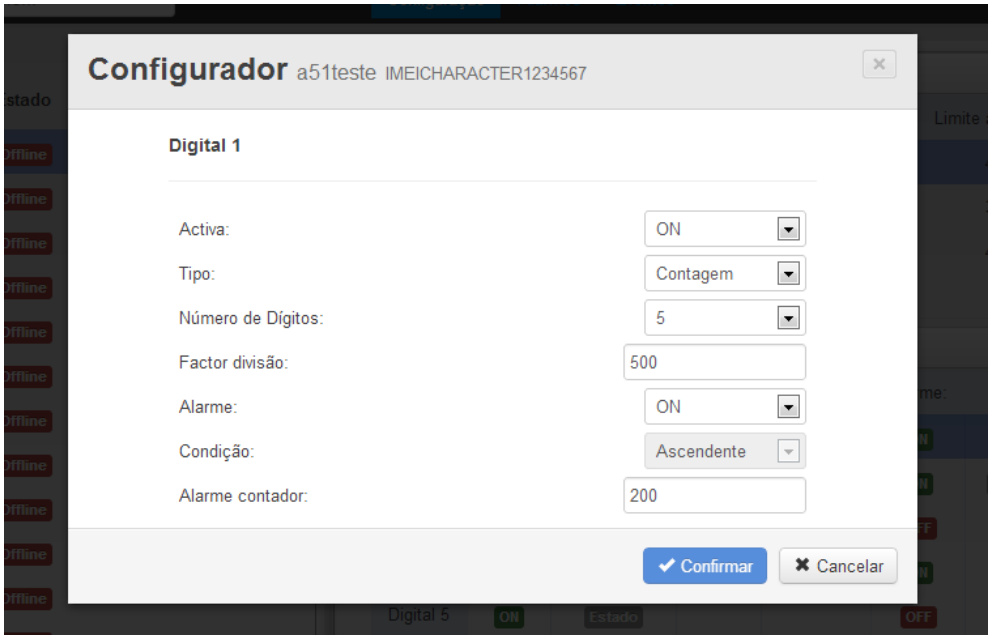


Figura 86 - *Popup* de configuração dos canais digitais.

Nesta janela, pode-se configurar:

- Entrada activada ou desactivada;
- Tipo de entrada: *Contagem*, caso se pretenda colocar a entrada como contadora de impulsos, ou *Estado* caso se pretenda controlar uma determinada variável e gerar um alarme se esta alterar o seu estado (nível lógico *alto* ou *baixo*);
- Número de dígitos: Representa o valor máximo até onde o contador pode incrementar, até fazer *overflow*. No caso de 5 dígitos, o valor máximo será 99999;
- Factor de divisão: relação entre o número de impulsos de entrada necessários para incrementar o contador;
- Alarme: Alarme activado ou desactivado;
- Condição: Condição de alarme no caso de o canal ser configurado do tipo *Estado*. As opções nesse caso são *Ascendente*, *Descendente* ou *Ambos*, referente à alteração do nível lógico da entrada;
- Alarme contador: representa o valor de contagem para o qual é gerado um alarme nesse canal;

Alarmes

No caso do WeSense DAQ, o *tab* de Alarmes é também disponibilizado. Neste *tab* apresenta-se sob a forma de lista todos os alarmes detectados no dispositivo seleccionado. O aspecto geral desta secção apresenta-se na Figura 87.

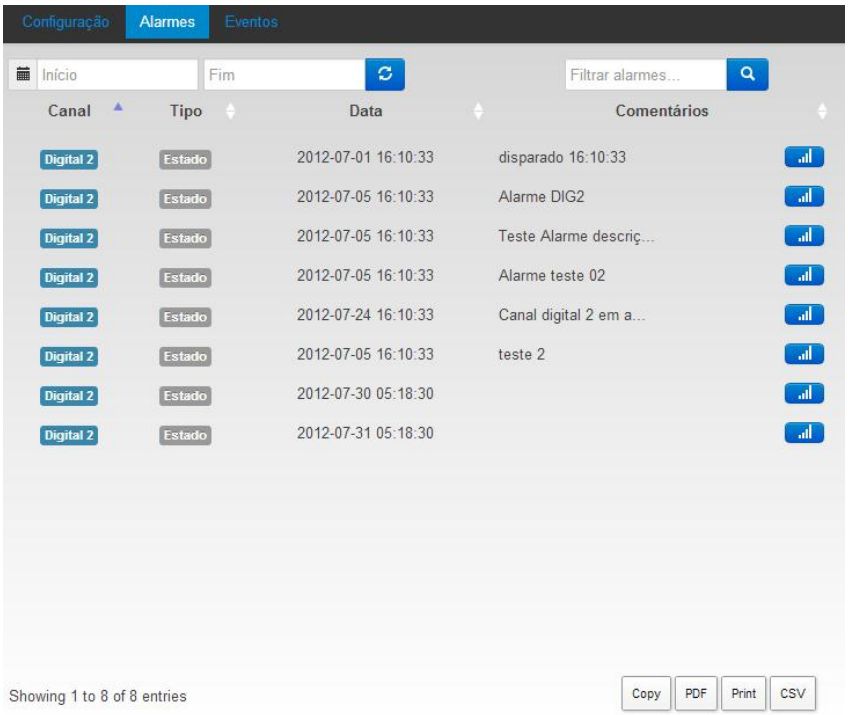


Figura 87 - *Tab* alarmes do dispositivo.

A distribuição dos elementos é uma combinação de alguns já vistos anteriormente. As características principais são as seguintes:

- Campos de definição do intervalo de tempo para o qual se pretende gerar a lista de alarmes (por omissão apresentam-se todos os alarmes);
- Campo de filtragem dos alarmes: permite caso tenham ocorrido alarmes em vários canais, filtrar por exemplo por um canal específico;
- Exportação da lista de alarmes semelhante à exportação da lista de dispositivos;
- Informações principais listadas na tabela e pormenorizadas ao clicar numa das entradas;
- Apresentação gráfica dos valores registados do canal em alarme;

Ao clicar numa das entradas de alarme, é apresentado o *popup* semelhante ao visível na Figura 88.

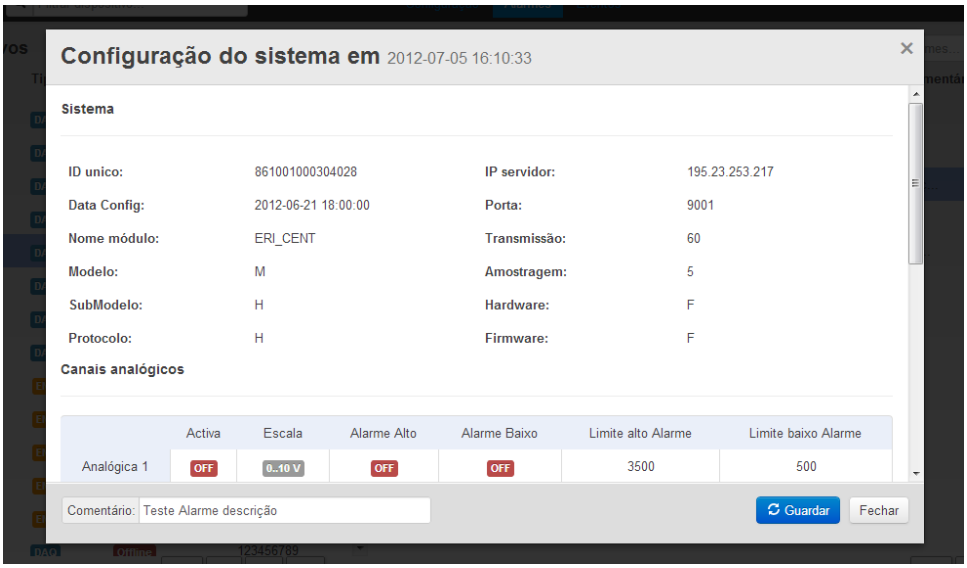


Figura 88 - *Popup* lista de alarmes.

Para além da informação apresentada na tabela anterior, é apresentada toda a configuração do sistema na altura em que ocorreu o disparo do alarme, prevenindo assim alterações da configuração do dispositivo que dificulte a detecção do problema. É também possível guardar um comentário ao alarme ocorrido, no campo presente no rodapé do *Popup*. Ao clicar no botão presente ao lado de cada entrada da *Lista de Alarmes*, é também apresentado um *popup*, desta vez com todos os valores registados no canal onde originou o alarme, conforme indicado na Figura 89.

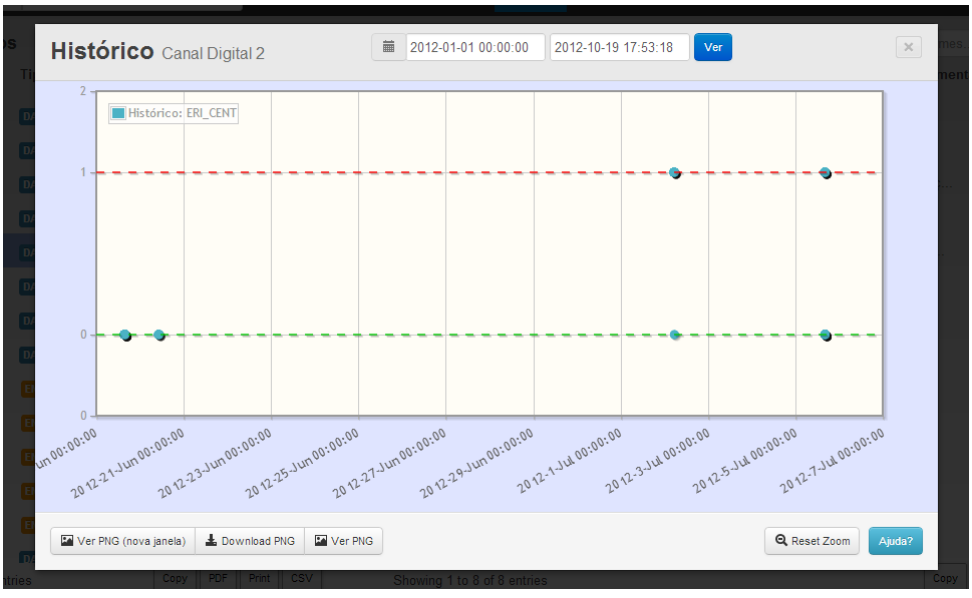


Figura 89 - *Popup* de alarmes detectados.

A estrutura é em tudo semelhante ao *popup* para o registo da temperatura e bateria do dispositivo, razão pela qual não se irão aprofundar as funcionalidades aqui presentes.

Eventos

O *tab* de eventos não foi desenvolvido uma vez que ainda não estão definidos pela empresa quais os eventos a serem enviados e determinados pelo dispositivo. A estrutura desse *tab* será, no entanto, semelhante ao tab de Alarmes.

No caso do módulo WeSense ENERGY, só é apresentado o *tab* de configuração, uma vez que não possui a funcionalidades de recolha de alarmes ou eventos, nem as tabelas com informação relativa à configuração das entradas analógicas ou digitais. O aspecto geral da página de configuração para os módulos de energia é a apresentada na Figura 90.

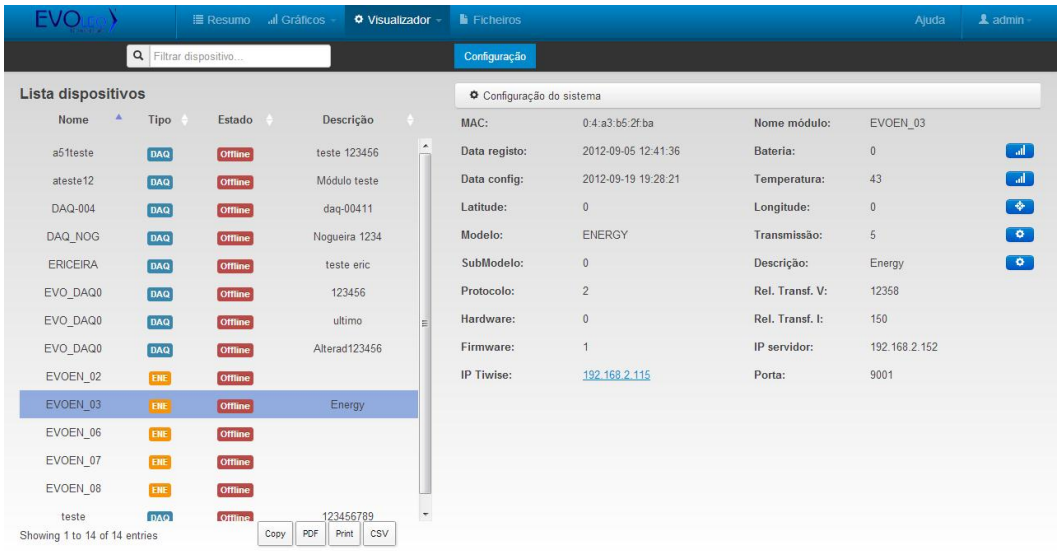


Figura 90 - Página de configuração do módulo WeSense ENERGY.

5.4.3. PÁGINA GRÁFICOS

Os dados recolhidos por cada módulo podem ser visualizados nas duas páginas de gráficos da plataforma. Dois modos foram escolhidos para apresentação da informação: traçado de (*i*) gráficos individuais relativos a cada módulo, e (*ii*) de grupos de variáveis de vários

módulos. Nesta secção apresentam-se ambas as páginas, com as principais características e funcionamento geral do sistema.

Pretendeu-se manter em maior parte das páginas a mesma estrutura visual: barra de navegação no topo, selecção de dispositivo ou grupo à esquerda e conteúdo à direita, conforme se apresenta na Figura 91.

Figura 91 - Estrutura de *design* adoptada.

Deste modo, homogeneiza-se e simplifica-se o aspecto visual das páginas, tornando-se mais evidente nas páginas de gráficos e de visualização/configuração.

A página de gráficos também sofreu sucessivas alterações, até atingir o apresentado nesta secção. Uma das primeiras versões, a de Maio de 2012, foi ainda versão de teste (sem dados reais, inseridos manualmente na base de dados), e cujo aspecto se pode visualizar na Figura 92.

Figura 92 - Protótipo inicial da página de gráficos.

Nesta versão, a informação apresentada dizia respeito a vários dispositivos, seleccionáveis na lista de opções ao lado esquerdo do gráfico e sendo a navegação realizada através dos botões no fundo da página. A biblioteca gráfica utilizada já nesta altura foi o jqPlot pelas características e facilidade de utilização em conjunto com jQuery, assim como o Bootstrap para o aspecto da página.

- Gráficos individuais de dispositivos

Nesta página são apresentados todos os dados recolhidos por um determinado módulo. No caso do WeSense DAQ são apresentados os gráficos relativos a *Contadores*, *Entradas Analógicas* e *Entradas Digitais*, separados em três *tabs* de forma semelhante aos *tabs* da página de *Configuração*. No caso do WeSense ENERGY, são apresentados os dados recolhidos relativos a energia activa, reactiva e aparente.

Figura 93 - Página de gráficos referentes ao WeSense ENERGY.

As características principais são as seguintes:

- Gráficos interactivos com capacidade de *zoom* e *span*;
- Legenda do gráfico permite esconder ou mostrar as séries pretendidas;
- Exportar valores e gráfico para Excel;
- Exportar imagem do gráfico (PNG);
- No caso do WeSense ENERGY, apresenta lista com os máximos e mínimos relativos à energia activa, reactiva e aparente e o número de amostras efectuado;
- Definição do intervalo de tempo de visualização semelhante ao da Figura 79;

- Biblioteca gráfica jqPlot;
- Navegação através de lista de dispositivos, semelhante à da página de configuração;

A Figura 94 ilustra a página de gráficos para os módulos DAQ, com aspecto e funcionalidades ligeiramente diferentes dos módulos ENERGY.

Figura 94 - Página de gráficos referentes ao WeSense DAQ.

Neste caso, existem três *tabs* que apresentam os dados recolhidos referentes a *Contadores*, *Entradas Analógicas* e *Entradas Digitais*, sendo traçado novo gráfico para cada uma das opções.

- Gráficos de grupos de dispositivos

A página de grupos apresenta várias alterações comparativamente à página de dispositivos, e trata-se de uma evolução desta. A Figura 95 ilustra a navegação nesta página através de um *Tablet*.

Figura 95 - Navegação na página de grupos em *Tablet* (Asus Nexus 7).

O objectivo futuro será o de introduzir algumas das funcionalidades presentes nesta página na de dispositivos, nomeadamente a melhor compatibilidade para navegação gráfica a partir de dispositivos *touch*.

As características principais presentes nesta página são as seguintes:

- Criação de grupos de variáveis para visualização no mesmo gráfico;
- Edição e remoção de grupos de variáveis;
- Navegação totalmente compatível com dispositivos móveis;
- Exportação de valores para Excel;
- Exportação da imagem do gráfico;
- Controlos de navegação e *zoom*;
- Legenda permite mostrar ou esconder as séries pretendidas;
- Biblioteca gráfica Flot em alternativa ao jqPlot;

A principal função desta página é a comparação de variáveis pretendidas no mesmo gráfico. Desta forma, a título exemplificativo, caso se pretenda comparar a evolução das leituras de energia activa entre vários aparelhos, pode-se seleccionar todas as variáveis pretendidas e visualizá-las no mesmo gráfico. A criação dos grupos é feita através de uma janela de *popup* conforme a Figura 96.

Figura 96 - *Popup* criação de grupos.

A estrutura é semelhante aos anteriores *popups*, sendo o corpo divido em três áreas: área de selecção de dispositivo, selecção de variável e a constituição actual do grupo.

Após seleccionado o dispositivo, consoante o seu tipo (DAQ ou ENERGY), são apresentadas as variáveis possíveis de escolha. As variáveis podem então ser arrastadas para os cinco *contentores* do grupo, ou adicionadas através de um simples clique na variável pretendida, até um máximo de cinco variáveis.

Na barra superior é apresentado o campo de filtragem da lista de dispositivos, e em rodapé os campos para definição do nome e descrição do grupo. Quer na adição de variáveis como na criação de grupos são efectuadas validações (número de variáveis, variáveis repetidas, nome de grupo em utilização).

Em termos de navegação da página, a estrutura é semelhante à da página de gráficos anterior, sendo a lista de dispositivos substituída pela lista de grupos. Na área de conteúdo da página, são apresentados os vários controlos de navegação do gráfico, controlos de exportação de ficheiros e o próprio gráfico, conforme apresentado na Figura 97.

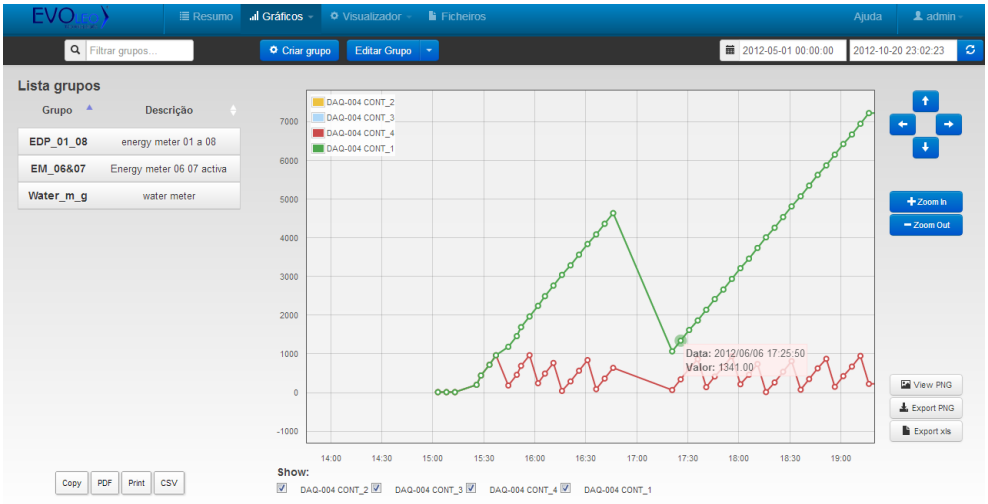


Figura 97 - Gráfico de grupos.

Os controlos direccionais permitem efectuar a movimentação de visualização do gráfico e os botões de *zoom* aumentar ou diminuir o *zoom* aplicado. O controlo pode também ser feito com o clique e arraste do rato e o *zoom* através do *scroll* do rato. O motivo principal dos controlos direccionais é então oferecer possibilidades de navegação em dispositivos *touch*. A Figura 98 apresenta os controlos existentes.

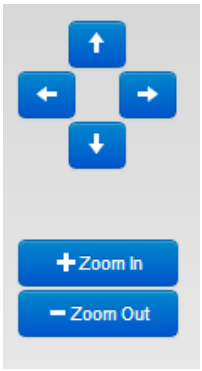


Figura 98 - Controlos direccionais e de *zoom* do gráfico.

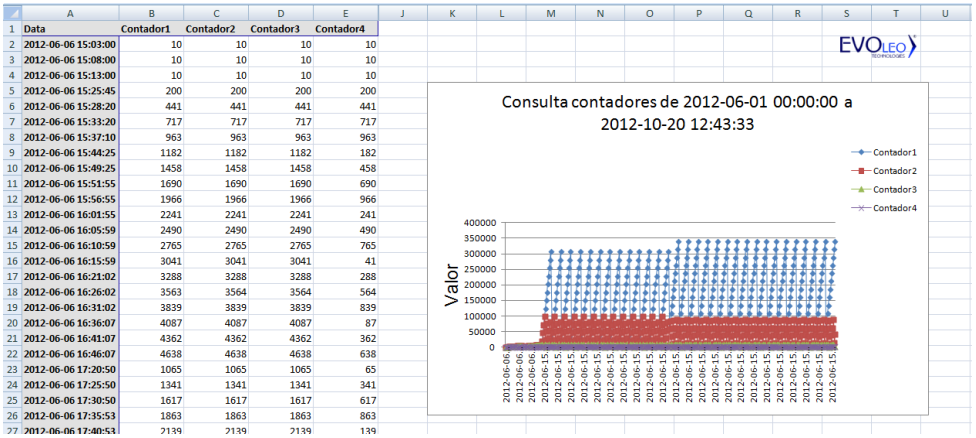
Os controlos de exportação dos dados e imagem dos gráficos são também apresentados na mesma área dos controlos direccionais, bem como a legenda do gráfico. Esta permite a selecção das variáveis que se pretende visualizar através de uma lista de *checkboxes.*

- Relatórios gerados

Os gráficos gerados podem também ser exportados para uma folha de cálculo, com toda a informação dos canais escolhidos. As características principais são as seguintes:

- Consulta detalhada dos valores apresentados no gráfico;
- Formato comum de consulta de dados;
- Apresentação do gráfico gerado dinamicamente através dos dados da folha de cálculo;
- Gerado dinamicamente pelo servidor.

A Figura 99 apresenta uma folha de cálculo exportada a título exemplificativo.



	Data	Contador1	Contador2	Contador3	Contador4
2	2012-06-06 15:03:00	10	10	10	10
3	2012-06-06 15:08:00	10	10	10	10
4	2012-06-06 15:13:00	10	10	10	10
5	2012-06-06 15:25:45	200	200	200	200
6	2012-06-06 15:28:20	441	441	441	441
7	2012-06-06 15:33:20	717	717	717	717
8	2012-06-06 15:37:10	963	963	963	963
9	2012-06-06 15:44:25	1182	1182	1182	182
10	2012-06-06 15:49:25	1458	1458	1458	458
11	2012-06-06 15:51:55	1690	1690	1690	690
12	2012-06-06 15:56:55	1966	1966	1966	966
13	2012-06-06 16:01:55	2241	2241	2241	241
14	2012-06-06 16:05:59	2490	2490	2490	490
15	2012-06-06 16:10:59	2765	2765	2765	765
16	2012-06-06 16:15:59	3041	3041	3041	41
17	2012-06-06 16:21:02	3288	3288	3288	288
18	2012-06-06 16:26:02	3563	3564	3564	564
19	2012-06-06 16:31:02	3839	3839	3839	839
20	2012-06-06 16:36:07	4087	4087	4087	87
21	2012-06-06 16:41:07	4362	4362	4362	362
22	2012-06-06 16:46:07	4638	4638	4638	638
23	2012-06-06 17:20:50	1065	1065	1065	65
24	2012-06-06 17:25:50	1341	1341	1341	341
25	2012-06-06 17:30:50	1617	1617	1617	617
26	2012-06-06 17:35:53	1863	1863	1863	863
27	2012-06-06 17:40:53	2139	2139	2139	139

Figura 99 - Exemplo do relatório exportado para Excel.

5.4.4. PÁGINA DE DEPURAÇÃO

Durante o desenvolvimento e teste da solução *WeSense Network*, surgiu a necessidade de expandir a plataforma de modo a que permitisse apoiar as operações de depuração. Assim, esta página foi utilizada para depuração quer da plataforma em desenvolvimento como do *hardware*/*firmware* desenvolvido pela equipa na Evoleo. Trata-se essencialmente de um registo com todas as comunicações efectuadas pela plataforma, de recepção e envio. A Figura 100 apresenta o aspecto visual da página de depuração.

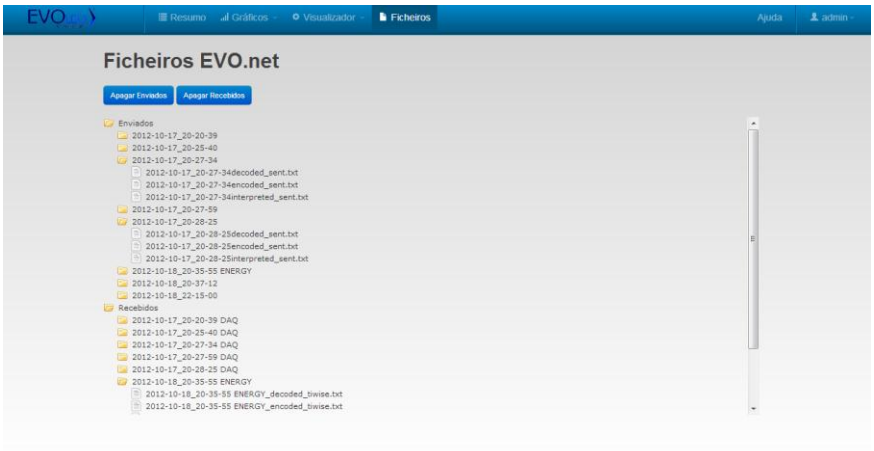


Figura 100 - Página de *debug*.

Os dados recebidos ou enviados são apresentados em três ficheiros:

- Sem qualquer processamento (*raw*);
- Com todos os valores descodificados no seu valor hexadecimal;
- Com o resultado da interpretação completa segundo o protocolo definido;

Esta página revelou-se particularmente útil na fase de depuração inicial de problemas na plataforma e *hardware/firmware*.

5.5. APLICAÇÃO ANDROID

A aplicação Android desenvolvida teve como principal objectivo o "*proof of concept*" da ideia definida na Figura 64 i.e. aceder aos dados da plataforma não necessariamente através do *site* desenvolvido, mas através de qualquer aplicação que tenha a possibilidade

de aceder à Internet. A aplicação desenvolvida lista todos os dispositivos da rede e permite a leitura e configuração de alguns campos comuns aos dois tipos de dispositivos.

A Figura 101 representa o ecrã de entrada na aplicação. Aqui define-se:

- O endereço IP do servidor da plataforma;
- Nome de utilizador;
- Palavra-passe;

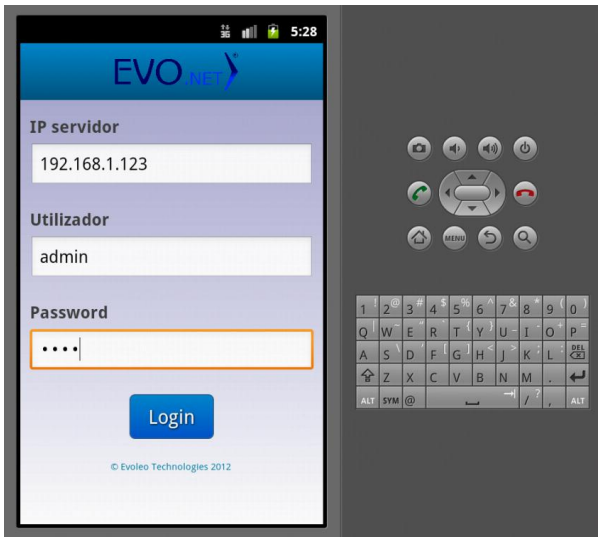


Figura 101 - Aplicação Android.

Após o *login* com sucesso no servidor, são apresentados os dispositivos detectados pela plataforma, conforme apresentado na Figura 102. Cada entrada apresenta assim alguma da informação mais relevante sobre cada dispositivo para facilitar a identificação.

Figura 102 - Menu de lista de dispositivos.

A comunicação entre aplicação e plataforma realiza-se da mesma forma que a página com a plataforma: pedidos GET e resposta codificada em JSON. Dado a popularidade deste tipo de codificação, existem já várias classes de *parse* JSON disponíveis pela Google, alteradas consoante a necessidade para a realização deste protótipo.

Ao tocar em qualquer uma das entradas, são apresentados alguns campos de configuração que podem ser alterados para reconfiguração do dispositivo. A Figura 103 apresenta o menu de reconfiguração de cada dispositivo.

Figura 103 - Menu de configuração do dispositivo.

O objectivo da aplicação será de futuro:

- Reestruturação da página de detalhes de cada dispositivo;
- Listagem de todos os campos de configuração;
- Apresentação de gráficos interactivos, semelhantes aos do portal;
- Página de detalhes de dispositivo separada em três *tabs* (Sistema, Entradas analógicas e digitais);
- *Design* da aplicação semelhante ao do portal;

5.6. RESUMO DO CAPÍTULO DE DESCRIÇÃO DA SOLUÇÃO

Neste capítulo foram apresentados os vários módulos desenvolvidos no âmbito do trabalho. O projecto global é de uma dimensão relativamente elevada, tendo sido desenvolvido durante os oito meses de estágio na empresa. No total, são cerca de 10 módulos independentes para a plataforma, que podem ser representados pela Tabela 5.

Tabela 5 Módulos desenvolvidos

	Módulo	Número de linhas	Número de caracteres
Processamento	Núcleo de processamento	6201	235750
	Frontend Núcleo	637	16474
	Aplicação de teste	350	9369
Páginas	Página Visualização/Config	5425	203423
	Página Gráficos grupos	1843	61307
	Página Gráficos individuais	1880	63327
	Página Depuração	230	6786
	Página Resumos	384	12249
	Página Login	263	8435
Servidor	Scripts Servidor (23 scripts)	2338	78300
Total		19551	695420

No total, foram desenvolvidas cerca de 20 000 linhas de código, a que correspondem cerca de 700 000 caracteres. Note-se que nesta tabela não está incluída a aplicação Android desenvolvida, nem as extensas folhas de estilo criadas para definir o *design* dos elementos das páginas.

6. VALIDAÇÃO

Um aspecto importante sobre qualquer trabalho desenvolvido passa necessariamente pela respectiva validação. Alguns sistemas de teste devem ser também desenvolvidos paralelamente ao trabalho principal de forma a avaliar o comportamento do sistema. Desta forma, torna-se possível efectuar a depuração de uma solução completa e em desenvolvimento. Nesta situação, quando surgir um erro, o problema pode ser relacionado com o *software* desenvolvido, com o *firmware* desenvolvido para cada módulo ou ainda com o *hardware* de cada módulo WeSense. Foi assim importante criar sistemas que permitissem ajudar a localizar a origem do erro. Nesta secção, apresentam-se os resultados obtidos, os procedimentos efectuados para validar o sistema, bem como as ferramentas desenvolvidas para tal efeito.

6.1. TESTES INICIAIS SEM *HARDWARE*

Para validação da plataforma, os testes iniciais foram desenvolvidos através de *software* desenvolvido especificamente para esse efeito. O elemento mais crítico para garantir o correcto funcionamento do sistema seria o núcleo de processamento, uma vez que é o elemento responsável por receber e descodificar os dados e armazená-los correctamente na

base de dados. Assim, desenvolveu-se uma aplicação de teste (*Aplicação Cliente*), esquematizada na Figura 104.

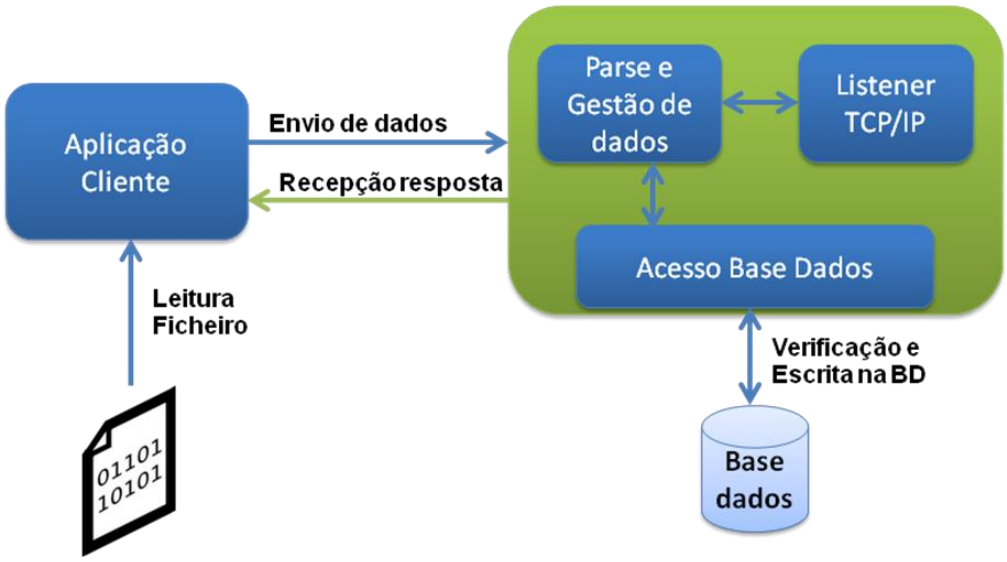


Figura 104 - Representação dos testes iniciais realizados.

Trata-se essencialmente de um bloco responsável por estabelecer ligação com o servidor e enviar os dados, de acordo com o protocolo específico de cada módulo. As configurações e dados a enviar eram obtidos através de um ficheiro de texto. Desta forma, foi possível testar e proceder à depuração do sistema sem dependência dos módulos de *hardware* em desenvolvimento, acelerando assim o processo de desenvolvimento da própria plataforma.

O que se pretendeu verificar através deste método consiste genericamente:

- Na descodificação correcta dos dados recebidos, antes de serem armazenados na base de dados (*print* para o ecrã da consola);
- No armazenamento correcto dos dados na base de dados (comparar valores inseridos no ficheiro de texto com valores descodificados pelo núcleo e com valores guardados na BD);
- Na rejeição de pacotes desconhecidos;
- Na avaliação da mensagem de resposta do núcleo de processamento;

6.2. INSTALAÇÃO-PILOTO: MAIA

Sensivelmente em Julho de 2012, foram produzidos os primeiros protótipos dos módulos WeSense, nomeadamente o WeSense DAQ, apresentado na Figura 105. A partir desta fase, foi possível começar os testes de validação do sistema com o *hardware* desenvolvido.

Figura 105 - Pormenor do WeSense DAQ.

Para tal, em conjunto com a equipa de desenvolvimento da Evoleo, foram instalados dois módulos WeSense DAQ ligados a um gerador de sinais para simular impulsos nas entradas digitais de contagem, e cinco módulos WeSense ENERGY, ligados a alguns equipamentos eléctricos de forma a medir os consumos efectuados. Foi assim criado um procedimento de teste para proceder à verificação dos requisitos enunciados no capítulo 4. Este procedimento foi realizado em conjunto com a equipa da Evoleo, para verificar se as configurações enviadas pelo portal para os dispositivos estavam de acordo com o esperado. Os primeiros módulos a serem testados foram os WeSense ENERGY, uma vez que apresentam menos campos de configuração e a verificação seria mais rápida. Como esperado, surgiram alguns erros nas fases iniciais, quer do lado da plataforma desenvolvida como do *firmware* dos dispositivos, que foram detectados e corrigidos. A fase de validação com o *hardware* desenvolvido iniciou-se em finais de Julho, tendo-se prolongado até Setembro. A lista de requisitos apresentada na subsecção seguinte é o resultado da fase de testes de validação realizados, e resultados obtidos.

6.3. LISTA DE REQUISITOS VALIDADA

Nesta secção apresenta-se novamente a tabela de requisitos, definida no início do estágio, de forma a verificar os objectivos que foram atingidos.

Tabela 6 Tabela de validação dos requisitos.

REQ ID	Descrição do Requisito	Aceitação
FUN 1	Tem de receber pacotes de informação provenientes do WESENSE DAQ e WESENSE Energy.	Validado
FUN 2	Tem de ser possível configurar o WESENSE-DAQ e WESENSE-Energy através da página *web*.	Validado
FUN 3	Os dados de configuração do WESENSE-DAQ e WESENSE-Energy têm de ser enviados por TCP/IP.	Validado
FUN 4	Tem de ser feito o parsing dos pacotes TCP/IP.	Validado
FUN 5	Tem de guardar os dados recebidos na Base de dados.	Validado
FUN 6	Tem de permitir Multi-threading / Multi-processing para suportar múltiplos pedidos.	Não implementado
FUN 7	Deve ser implementado em servidor APACHE.	Validado
FUN 8	Deve-se utilizar MySQL para acesso à base de dados.	Validado
FUN 9	A página tem de ser desenvolvida em HTML, *Javascript*, CSS e PHP.	Validado
FUN 10	O *web* server tem de funcionar na plataforma Windows.	Validado
FUN 11	O módulo de recepção de pacotes do WESENSE tem de ser desenvolvido em C/C++.	Validado
FUN 12	Interface gráfico no browser tem de estar disponível para qualquer utilizador	Validado
FUN 13	Interface de reconfiguração do WESENSE tem de estar disponível só para administrador do sistema.	Não implementado
FUN 14	A página *web* tem de permitir a visualização de gráficos de todos os tipos de dados provenientes dos equipamentos WESENSE.	Validado
FUN 15	O eixo de tempo do gráfico tem de ser parametrizável.	Validado

FUN 16	Tem de ser possível actualizar os gráficos em tempo real, sem necessidade de fazer refresh da página.	Não implementado
FUN 17	Pedidos por parte do cliente têm de ser processados pelo script PHP do lado do servidor.	Validado
FUN 18	Deve ser utilizada a *framework* Dojo/jQuery para o desenvolvimento de interface gráfica.	Validado (Dojo não utilizado)
FUN 19	O cabeçalho da página deve conter identificação do projecto e empresa.	Validado
FUN 20	Deve existir um menu horizontal com cinco páginas: Resumo, Gráficos, Vista detalhada, Reconfiguração e Exportação de dados.	Validado com alterações
FUN 21	Tem de ser possível selecionar quais os sensores WESENSE sobre os quais se pretende mostrar informação.	Validado
FUN 22	Tem de ser possível parametrizar os gráficos.	Validado
FUN 23	Tem de ser possível realizar o autofit do gráfico no eixo vertical.	Validado
FUN 24	Tem de ser possível alterar a grandeza observada(V, mA, Wh).	Validado
FUN 25	Deve ser possível a visualização individual do nível da bateria do módulo WESENSE (Gauges semelhantes aos do LabView).	Validado
FUN 26	Deve ser possível a visualização individual da temperatura interior do módulo WESENSE (Gauges semelhantes aos do LabView).	Validado
FUN 27	Visualização individual de todas as características de cada WESENSE.	Validado
FUN 28	Deve-se utilizar as coordenadas GPS fornecidas pelo módulo para o posicionar num mapa na página.	Validado
FUN 29	Tem de ser apresentada a última hora de configuração realizada.	Validado
FUN 30	Tem de ser possível reconfigurar o intervalo de transmissão do WESENSE.	Validado
FUN 31	O período de amostragem dos contadores tem de ser reconfigurável.	Validado

FUN 32	Tem de ser possível exportar dados referentes a um ou vários sensores sob a forma de ficheiros de texto (CSV e XML).	Validado
FUN 33	Deve ser possível visualizar vários traçados de vários sensores no mesmo gráfico.	Validado
FUN 34	Tem de existir uma página onde se apresentam factores mais relevantes como os resumos de alarmes e medições mais recentes (home).	Parcial (medições abandonado)
FUN 35	Tem de existir uma página onde se apresenta o gráfico de todos os WESENSE nessa rede e tabelas de manipulação do gráfico.	Não implementado
FUN 36	Tem de existir uma página onde é possível ver informação detalhada sobre cada WESENSE em particular (temperatura, tensão da bateria, posição geográfica)	Validado
FUN 37	Tem de existir uma página de reconfiguração do WESENSE com todos os parâmetros reconfiguráveis disponíveis.	Parcial (só alguns parâmetros)
FUN 38	Tem de exisitir uma página de exportação de informação sob o formato CSV e XML.	Não implementado (incluido nas outras páginas)
FUN 39	Deve ser possível configurar os módulos através de uma aplicação para *smartphone*, dedicada.	Validado

Relativamente aos requisitos que não foram cumpridos:

- O *multi-threading* ou *multi-processing* foi inicialmente considerado, mais tarde colocado de parte uma vez que o intervalo de transmissão entre as mensagens seria no mínimo de 5 minutos, e mais tarde voltado a ser considerado como relevante. Contudo, não foi possível implementar o *multi-thread* por questões limitação de tempo.
- O interface de reconfiguração de cada módulo WeSense está disponível, actualmente, para qualquer utilizador que tenha acesso à plataforma. Não foi detectada, para já, necessidade de criar vários níveis de utilizadores com diferentes níveis de acesso e restrições. Contudo, será algo a implementar em trabalho futuro.
- A actualização dos gráficos em tempo real foi também rejeitada uma vez que o intervalo de transmissão de dados não o justifica (5 minutos no mínimo até novos dados de cada sensor).

- Inicialmente planeava-se utilizar a biblioteca Dojo em conjunto com jQuery, para desenvolver a interface *web*. No entanto, no desenrolar do projecto, mostrou-se suficiente utilizar jQuery com os *plugins* necessários.
- A estrutura inicial para o *site* foi modificada, passando a página de visualização a englobar a visualização de dados e permitir a reconfiguração de cada módulo. A página de exportação de dados foi também eliminada, passando a função de exportação a ser uma opção nas restantes páginas.
- O previsto para a página de resumos foi mais tarde alterado. Não se mostrou útil aparecer informação sobre os resultados das últimas medições efectuadas, pelo que essa função foi abandonada. Optou-se antes por apresentar informação mais genérica sobre a instalação como alarmes, novos dispositivos instalados, eventos, etc..
- A página com gráficos de todos os módulos na instalação mostrou-se ser pouco perceptível com a quantidade de informação que iria estar presente. Desta forma, optou-se por separar os dados em duas páginas, gráficos individuais de cada dispositivo ou gráficos de grupos de variáveis (no máximo cinco variáveis).
- Relativamente à configuração de cada dispositivo, optou-se por não permitir a reconfiguração de todos os campos através do portal, como é o exemplo do IP do servidor. Caso se mudasse o IP do servidor, o módulo deixaria de transmitir para o servidor actual e seria impossível voltar a reconfigurá-lo remotamente.

6.4. VALIDAÇÃO DA PLATAFORMA EM DIVERSOS NAVEGADORES

Outro teste efectuado consistiu na validação da plataforma em vários navegadores. Desde o início do desenvolvimento da plataforma que houve uma preocupação em manter a compatibilidade com o maior número de dispositivos e *browsers* possíveis. Foi assim desenvolvido o sistema, sempre tendo em conta a utilização do portal em *tablets*, pelas diferenças de controlo que apresentam face a *desktop*. Um desses aspectos é a interactividade com gráficos, nomeadamente através do efeito de *scroll* do rato e através do "*drag & drop*".

As resoluções de ecrã deste tipo de dispositivos são também inferiores à maioria dos existentes para *desktop*, pelo que o aspecto da página se deve tentar manter.

A interface *web* da plataforma é totalmente compatível com as versões mais recentes dos *browsers* mais comuns. Todas as funções foram testadas nos seguintes *browsers*:

- Internet Explorer 9 (32 e 64 bit);
- Google Chrome;
- Mozilla Firefox;

As funções de exportação para imagem só são suportadas na totalidade no Google Chrome.

Um dos motivos que levou a adopção da biblioteca Flot nos gráficos de grupos, ao invés da biblioteca jqPlot nos gráficos de cada dispositivo, deveu-se a estes testes realizados. Mostrou-se inviável tornar o gráfico interactivo utilizando o jqPlot, uma vez que a possibilidade de *Zoom* era realizada através de *click&drag*, só possível através do rato. Desta forma, criaram-se os controlos na própria página, conforme visto anteriormente e na Figura 106.

Figura 106 - Página de grupos no *Tablet* Nexus 7.

A plataforma foi também testada no *tablet* Nexus 7 e em *smartphone*s Android, sendo a totalidade das funções suportadas. Adicionalmente, o mapa de dispositivos suporta multi-touch e o *pinch-to-zoom* tornando assim a navegação pelo mapa mais simples. A Figura 107 ilustra a navegação através do *pinch-to-zoom* no portal.

Figura 107 - Controlo do mapa de visualização do dispositivo.

A visualização da informação e reconfiguração nos *tablets* mostrou-se particularmente interessante mesmo durante as fases de desenvolvimento e testes de calibração na empresa, pelo tamanho reduzido e funcionalidade dos mesmos.

6.5. PARTICIPAÇÃO NA CONFERÊNCIA EES 2012

Com base neste trabalho, foi publicado um artigo (ver Anexo A) que foi submetido pela empresa à 1ª Conferência sobre Energia, Ambiente e Sustentabilidade, realizada no ISEP nos dias 26 e 27 de Setembro de 2012. O trabalho foi aceite pela comissão da conferência e apresentado. A Figura 108 mostra a apresentação deste trabalho na conferência.

Figura 108 - Apresentação da plataforma na conferência EES.

Os principais objectivos desta conferência foram a apresentação de novas propostas para redução de consumos e desperdícios energéticos e apresentação de fontes alternativas de energia que podem ser viáveis. A apresentação decorreu no âmbito da sustentabilidade e redução de consumos energéticos proporcionada pela solução desenvolvida.

As principais questões colocadas pelos participantes foram referentes à segurança da plataforma e dos dados, algo que de facto não foi tido como requisito principal do trabalho. *HTTPS* seria uma opção a ter em conta, permitindo a encriptação dos dados transferidos entre cliente e servidor. Actualmente, a segurança é fornecida através da limitação do acesso à página através da página de *login*, o que foi de início previsto como suficiente, uma vez que o objectivo seria instalar a plataforma numa rede local sem acesso à Internet.

7. CONCLUSÕES E TRABALHO FUTURO

Ao longo deste texto foram sendo apresentadas conclusões que permitiram sustentar as opções de desenvolvimento efectuadas ao longo do projecto. Assim, nesta última secção é realizada uma síntese das principais conclusões, consequências e relevância do trabalho realizado e perspectivados futuros desenvolvimentos.

A fase inicial de estudo do estado da arte mostrou-se relevante para a tomada de decisão de como abordar o problema. Uma vez que o objectivo inicial não estava ainda definido, foi importante fornecer todas as hipóteses de desenvolvimento que se poderiam tomar, para assim se optar pela que fosse mais conveniente.

Foi desenvolvida uma plataforma com um nível de complexidade relativamente elevado. Foram desenvolvidos 10 módulos gerais para o servidor, que totalizam cerca de 20 000 linhas de código a que correspondem quase 700 000 caracteres.

A plataforma desenvolvida mostrou-se versátil, permitindo o acesso e visualização de dados a partir de diversos dispositivos, como mostrou a aplicação Android desenvolvida, não ficando limitada à interface gráfica *web*, onde recaiu grande parte do trabalho.

A opção de realizar a interface gráfica via *web browser* mostrou-se também vantajosa: a maior parte dos *browsers* testados suporta a interface, sem necessidade de instalar *software* adicional no dispositivo cliente. A não utilização de *Flash* foi também uma vantagem, uma vez que este tem vindo a ser abandonado pela maioria dos *browsers* actuais, principalmente de *smartphones*.

A plataforma apresenta também a possibilidade de suportar novos dispositivos, baseados no mesmo protocolo, como se pôde constatar pelos dois módulos desenvolvidos (o objectivo inicial era o desenvolvimento da interface para o WeSense DAQ, e mais tarde teve de acomodar o WeSense ENERGY também). A ideia da solução WeSense passa mesmo por aqui: ser uma plataforma que suporte os dispositivos criados futuramente na Evoleo Technologies. Um desses exemplos seria o módulo de aquisição de vibrações, WeSense Vibrations, tendo já sido alvo de um estágio curricular na empresa a placa de condicionamento de sinal. O conceito pretende não só abordar a aquisição de dados mas também o controlo da instalação. Nessa altura tornar-se-ia relevante um módulo actuador controlado remotamente. O conceito da solução futura pode ser representado pelo diagrama apresentado na Figura 109:

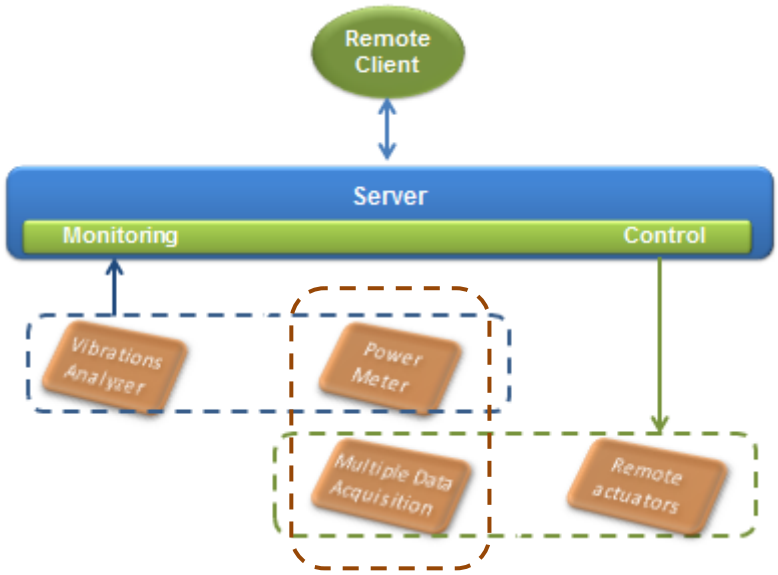


Figura 109 - Diagrama da solução futura WeSenseNet.

As vantagens apresentadas por esta solução são essencialmente as das redes WSN. A utilização desta solução será útil principalmente a nível industrial dado que permite monitorizar e controlar consumos das várias máquinas, analisar as vibrações de vários motores, que poderão dar indícios de avarias, ou encurtamento da vida útil da máquina. A

instalação numa ETAR é também uma opção já que através dos módulos WeSense DAQ poder-se-á monitorizar os níveis de água nos tanques, ou o estado de válvulas na instalação. Uma vez que o módulo possui entradas analógicas, dependendo do tipo de sensor utilizado, poder-se-á medir o pH da água, ou a concentração de cloro, por exemplo. O WeSense DAQ mostra-se como o módulo mais versátil com as duas entradas em tensão e em corrente, sendo assim compatível com uma grande quantidade de sensores.

Contudo, o trabalho está ainda em desenvolvimento e planeiam-se já alterações à plataforma.

O SGBD utilizado deverá ser substituído pelo PostgreSQL, uma vez que apresenta uma licença livre mesmo para uso comercial ao contrário do MySQL.

Apesar de inicialmente definido nos requisitos a compatibilidade da plataforma ser destinada a Windows, o objectivo futuro será tornar a plataforma compatível com Linux. Desta forma, a sugestão passa por desenvolver o núcleo de processamento em PHP, mantendo compatibilidade com Windows e Linux.

A aplicação Android, desenvolvida principalmente como "*proof of concept*" pretende-se também que permita a visualização dos dados de forma interactiva, recorrendo a gráficos, mantendo algumas semelhanças com a estrutura do portal.

Em termos de segurança, pretende-se como já foi dito tornar a ligação segura através de protocolos conhecidos como o HTTPS, encriptando a ligação cliente-servidor.

Tendo em conta os requisitos definidos inicialmente, conclui-se que a plataforma cumpre cerca de 88% destes. Contudo, deve-se salientar que no decorrer do trabalho, muito frequentemente foram sendo definidos novos requisitos a cumprir pela plataforma, como novas funcionalidades ou suporte a novos dispositivos, o que tornaria a lista de requisitos total muito extensa, motivo pelo qual se optou por apresentar a tabela de requisitos inicial aprovada pela Evoleo.

De um ponto de vista pessoal, o desenvolvimento deste trabalho permitiu acima de tudo a participação num dos principais projectos da empresa, desenvolvido durante vários meses em conjunto com toda a equipa de *I&D* da Evoleo Technologies. Tendo em conta a constante interacção entre os elementos do grupo de trabalho, todos os dias foram surgindo

novos desafios, novas ideias e novas propostas a implementar na plataforma, tendo assim sido um trabalho dinâmico no decorrer de todo o estágio.

Referências Documentais

[1] DIAS C., - *As energias renováveis no trinómio Desenvolvimento, Energia e Ambiente*, Caderno das energias renováveis, separata da revista *O electricista*.

[2] OMER A.M., 2008. - *Energy, environment and sustainable development. Renew Sust Energy* Rev;12 (9):2265–300.

[3] EUROSTAT, http://epp.eurostat.ec.europa.eu/portal/page/portal/eurostat/home/ visitado em Setembro 2012

[4] BARROS, Cristina – *Plano de Promoção de eficiência no Consumo de Energia Eléctrica (PPEC).* ERSE, 13 de Março 2012.

[5] MASOSO, O.T., GROBLER, L.J. - *The dark side of occupants' behaviour on building energy use, Energy and Buildings*, Volume 42, Issue 2, Fevereiro 2010, Páginas 173–177.

[6] LINDELÖF D., MOREL N., - *A field investigation of the intermediate light switching by users, Energy and Buildings*, 38 (2006), pp. 790–801.

[7] MAHDAVI A., MOHAMMADI A., KABIR E., LAMBEVA L., - *Occupants' operation of lighting and shading systems in office buildings*, Journal of Building Performance Simulation, 1 (1) (2008), pp. 57–65.

[8] RNAE (Associação Nacional das Agencias de Energia e Ambiente), 28 Março 2012.

[9] *Plano de Promoção da Eficiência no Consumo da Energia Eléctrica Para 2011-2012*, Entidade Reguladora dos Serviços Energéticos (ERSE), Novembro 2010.

[10] MARTINS R. - *Data Interface – Data Handling and Processing Unit*, Evoleo Technologies Paper, 2010.

[11] ESA TELECOM, http://telecom.esa.int/telecom/www/category/index.cfm?fcategoryid=205&completed_projs=1, visitado em 4 Outubro 2012

[12] AMARAL M., BRÁS T., *Portuguese Space Catalog 2011*, Fundação para a Ciência e a Tecnologia, I.P. Junho 2011.

[13] EVOLEO Technologies, www.evoleotech.com/space.htm

[14] EVOLEO Technologies, www.evoleotech.com/competences.htm

[15] PINTO I., *Análise Estratégica e Desenvolvimento de uma Estratégia de Expansão para a Evoleo Technologies,* Tese de Mestrado em Marketing, Outubro 2012.

[16] MARTINS R., *Sistema de Monitorização das vibrações dos "bogies" do comboio rápido Alfa Pendular entre Porto e Lisboa,* National Instruments Case Study.

[17] LEWIS F.L. - *Wireless Sensor Networks,* Smart Environments: Technologies, Protocols, and Applications ed. D.J. Cook and S.K. Das, John Wiley, New York, 2004.

[18] ZUNGERU A.M., ANG L., SENG K.P. - *Classical and swarm intelligence based routing protocols for wireless sensor networks: A survey and comparison*, Journal of Network and Computer Applications, Setembro 2012.

[19] NATIONAL INSTRUMENTS – *Wireless Sensor Network Topologies and Mesh Networking*, 26 Março 2010.

[20] NATIONAL INSTRUMENTS – *What is a Wireless Sensor Network?* 23 Julho 2009.

[21] ESTRIN D., GIROD L., POTTIE G., SRIVASTAVA M. – *Instrumenting the world with Wireless Sensor Networks,* ieeexplore.ieee.org, 2001.

[22] SOHRABY, K., MINOLI, D., ZNATI, T. - *Wireless sensor networks: technology, protocols, and applications*, John Wiley and Sons, 2007 ISBN 978-0-471-74300-2, pp. 203–209.

[23] DARGIE, W. e POELLABAUER, C., - *Fundamentals of wireless sensor networks: theory and practice*, John Wiley and Sons, 2010 ISBN 978-0-470-99765-9, pp. 168–183, 191–192.

[24] HART, J. K. e MARTINEZ, K. - *Environmental Sensor Networks:A revolution in the earth system science?* Earth-Science Reviews, 78. pp. 177-191. 2006.

[25] WERNER-ALLEN G., LORINCZ K., WELSH M., MARCILLO O., JOHNSON J., RUIZ M., LEES J., - *Deploying a Wireless Sensor Network on an Active Volcano*, IEEE Internet Computing, vol. 10, no. 2, pp. 18-25, 2006.

[26] SURIE, D., LAGUIONIE, O., PEDERSON, T. - *Wireless Sensor Networking of Everyday Objects in a Smart Home Environment*. International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Sidnei, Australia, pp. 189-194, 2008.

[27] SHAMIM N. PAKZAD; GREGORY L. FENVES; SUKUN K.; e DAVID E. CULLER – *Design and Implementation of Scalable Wireless Sensor Network for Structural Monitoring* Journal of Infrastructure Systems © ASCE, Março 2008.

[28] LOW K. S., - *Wireless Sensor Networks for Industrial Environments,* ieeexplore.ieee.org, 30 Novembro 2005.

[29] KENNEY J.D., POOLE D.R., WILLDEN G.C., ABBOTT B.A., MORRIS A.P., MCGINNIS R.N., e FERRILL D.A., *Precise Positioning with Wireless Sensor Nodes: Monitoring Natural Hazards in All Terrains*, IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX, USA, Oct. 2009.

[30] HARVARD SENSOR NETWORKS LAB, http://fiji.eecs.harvard.edu/Volcano visitado em 4 Abril 2012.

[31] YAJIE M., RICHARDS M., GHANEM, M., GUO, Y., HASSARD J. - *Air Pollution Monitoring and Mining Based on Sensor Grid in London*. 2008.

[32] LIBELLIUM WASPMOTE, http://www.libelium.com/wireless_sensor_networks_to_detec_forest_fires/ visitado em 4 Abril 2012.

[33] PEREIRA A., POUPA C., - *Linguagens Web*, Edições Sílabo, 4ª Edição, 2011.

[34] CROWDER D.A. – *Criar Websites para Totós.* Wiley Publishing Inc. Porto Editora, Setembro de 2008.

[35] W3TECHS - http://w3techs.com/technologies/overview/client_side_language/all visitado em 10 Abril 2012.

[36] GARRET J.J. – *AJAX: A New Approach to Web Applications*, 18 Fevereiro 2005.

[37] WELLING L., THOMSON L. – *PHP and MySQL development*, Indianapolis, Ind: ams, 2ª Edição, 2003.

[38] WIKIPEDIA http://pt.wikipedia.org/wiki/sql visitado em 16 Setembro 2012.

[39] RONALDO – *Funções básicas de um Sistema de Gestão de Bases de Dados.*

[40] OPEN HANDSET ALLIANCE - *Industry Leaders Announce Open Platform for Mobile Devices.* http://www.openhandsetalliance.com/press_110507.html visitado em Junho 2012.

[41] MOBITHINKING.COM - Global mobile statistics 2012 http://www.itu.int/net/pressoffice/press_releases/2012/70.aspx. visitado em Junho 2012.

[42] ANDROID DEVELOPERS - *What is Android?* http://developer.android.com/guide/basics/what-is-android.html Visitado em 16 Junho 2012. visitado em Junho 2012.

[43] ANDROID DEVELOPERS - http://developer.android.com/images/system-architecture.jpg. visitado em Junho 2012.

[44] SKILL GURU - http://www.skill-guru.com/blog/2011/01/13/android-activity-life-cycle/. visitado em Junho 2012.

[45] SENSbee SAYME.es - http://sayme.es/?lang=en, visitado em Março 2012.

[46] WORKLIGHT – http://worklight.com, visitado em Março 2012.

[47] COSM – http://cosm.com, visitado em Março 2012.

[48] LIBELLIUM WASPMOTE - http://www.libelium.com/, visitado em Março 2012.

[49] QT - http://en.wikipedia.org/wiki/Qt_(framework), visitado em Março 2012.

[50] AQUAMON - http://www.cermetek.com/Catalog/Coming-Soon/product.cfm?AquaMon - visitado em Março 2012.

[51] SENSORCLOUD - http://www.sensorcloud.com visitado em Março 2012.

Anexo A. *Abstract* submetido para a conferência EES

Energy, Environment and sustainability

1St International Conference

September 26th – 27th 2012

An infrastructure to support resources management – The Human-machine interface

Leite A.[1], Sousa A.[2], Felgueiras M.[3], Martins R.[4]

1 pedro.leite@evoleotech.com

2 antonio.sousa@evoleotech.com

3 mcf@isep.ipp.pt

4 rodolfo.martins@evoleotech.com

Annotation

Electric energy consumption has been rising as a direct consequence of global development and the increasing number of electric devices. A significant percentage of electrical energy has been produced through non-renewable energy resources, yet, the energy dependency, associated with the price escalation and the reduction of greenhouse gas emissions, fosters new solutions based on the global performance to be developed. The performance, however, depends on infrastructure characteristics as well as on the user behaviour. The energetic performance in buildings is very important once they consume more than half of the total energy produced. In order to achieve better performance it's important to not only consider the infrastructure performance but also monitor the user behaviour. This last issue poses several limitations once it largely depends on the kind of user. The present work focuses on the human-machine interface developed to remotely store, monitor and control energy consumption values as well as monitor the behaviour of building occupants, obtained by a series of installed devices throughout the monitored infrastructure, as part of the whole project developed at Evoleo Technologies.

Keywords: *Energy management, buildings, user behaviours, sustainability.*

1. INTRODUCTION

All Human development carries the increase of energy consumption on all its forms, which has created a negative impact in the environmental quality. Thus, the energy consumption has both positive and negative effects in the quality of life. In this context, energy consumption in buildings take a very important role since globally they are responsible for the consumption of 40% of the total amount of energy produced [1]. These numbers are particularly important to the European Union (EU) since the external energy dependency surpasses 50% of all the consumed energy. For some countries this rate is even more impressive, assuming vales over than 80% [2]. The energy efficiency is very important and associated with economics factors (high fuel prices, markets instability, etc.), environmental factors (green house gas effect, NOx production, natural

resources dilapidation etc.), and social factors. In order to achieve a balanced development, the sustainable development proposal has been introduced. The EU established a set of challenging goals until 2020 but only a small part of its budget was invested in education program schools and information campaigns [3]. As consequence, we are in a good way in terms of products efficiency but we still have a large amount of energy waste that depends strictly on the end energy user, i.e. the user efficiency. Several studies show that the bottleneck of the efficiency depends largely on the users. In some cases more energy is used during non-working hours (56%) than during working hours (44%) [4]. Lindelöf and Morel [5] have studied light switching patterns of occupants. During working hours, they noticed that due to the location of the light switch at the door (out of arm's reach from working desk), lights are left on all day even when they are not needed. Mahdavi et al. [6] tracked 48 offices from three buildings with digital cameras and other data recorders. They noticed that on average, occupants spend more than 50% of the time away from their work station nevertheless lights and office equipment remain operational all day long. To pursuit the challenging target for 2020 it's important to conduct a detailed study of the human behaviour in the buildings, particularly for the public ones where the energy consumption is higher.

This work presents part of an infrastructure that enables studying behaviours of building's users.

2. The Wireless Sensor Network (WSN) infrastructure

One of the present emerging concepts are called the wireless sensor networks. With today's technology, small modules can be developed with many connectivity possibilities, processing power and autonomy without being overly expensive. This provides the means to deploy several devices throughout the desired area to collect a variety of data and then pass it through the network to the central server.

The fundamental blocks of the designed sensor infrastructure are called "WeSense" modules, designed at Evoleo Technologies. These nodes collect data from the field of operation and periodically send it to the central server, where it is stored and made available to the user. The collected data can then be presented to the user, giving a record of energy consumption in a given timeframe, since the installation of the WSN took place. Given all the collected data is stored on the server, studies can be made to determine typical usage, possible faulty equipment, quality of the electric energy, etc, therefore, determining where energy is being wasted and enable the user to take actions to diminish it. If the central server is connected to the internet, remote monitorization of energy consumption and reconfiguration of the devices properties is possible.

3. Results

A variety of accessibility options was one of the main concerns from the concept stages of the project, and as such, the choice was made to develop the interface as a "*web*-app": essentially providing energetic consumption monitorization and control in the infrastructure from any device with a browser and internet access, but without needing to install any *software* on the users side.

The main advantage of the proposed solution resides on being able to easily monitor the energy consumption of the installation from any part of the world through any device that has internet access, and being able to remotely control the system. The chosen tools used to develop the *software* (html, *JavaScript*, PHP) gives it the possibility to run seamlessly among all *smartphone*s, tablets, or PC platforms, provided there is an installed browser and working internet connection. The developed prototype system enables the user to securely enter his central server and visualize the devices configuration and collected data through interactive charts or data tables, visualize or set alarms for determined variables and reconfigure each device remotely.

Since all the data is stored on the server, additional plugins can be developed to extend functionality to the already existing platform.

The server's main tasks can be divided in two essential functions:

- accept incoming connections originated by the WeSense modules, receive the data stream, decode it, and afterwards safely store the information on a database;
- accept client requests to view the collected data online;

The presented work offers functionality at very little expense: the developed server *software* can be installed on any machine running Linux or Windows OSes, and is planned to be distributed free of charge to clients acquiring the WeSense modules.

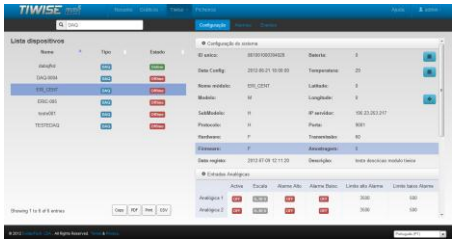


Fig 1 - Developed Interface

4. Conclusions

One of the major goals of this project is providing connectivity, and therefore the possibility of generating feedback, to existing systems, where there was none initially. This becomes particularly interesting in the field of energy efficiency, as it enables users to take control of their expenses with actual feedback from each device i.e. inspecting a chart from the consumption made by a specific device during the last week, will help end users determine if there is something wrong with that device and take the appropriate action.

5. References

[1] Omer A.M., 2008. Energy, environment and sustainable development. Renew Sust Energy Rev;12 (9):2265–300.

[2] Eurostat, 2009

[3] Entidade Reguladora dos Serviços Energéticos; http://www.erse.pt/pt/Paginas/home.aspx

[4] O.T. Masoso, L.J. Grobler, The dark side of occupants' behaviour on building energy use, Energy and Buildings, Volume 42, Issue 2, February 2010, Pages 173–177.

[5] D. Lindelöf, N. Morel, A field investigation of the intermediate light switching by users, Energy and Buildings, 38 (2006), pp. 790–801

[6] A. Mahdavi, A. Mohammadi, E. Kabir, L. Lambeva, Occupants' operation of lighting and shading systems in office buildings, Journal of Building Performance Simulation, 1 (1) (2008), pp. 57–65

Anexo B. Fotos conferência

Figura 110 - Apresentação da solução realizada na conferência EES 2012.

Figura 111 - Conferência EES 2012.

Figura 112 - *Stand* Evoleo Technologies na Conferência.

Anexo C. *Poster* Conferência

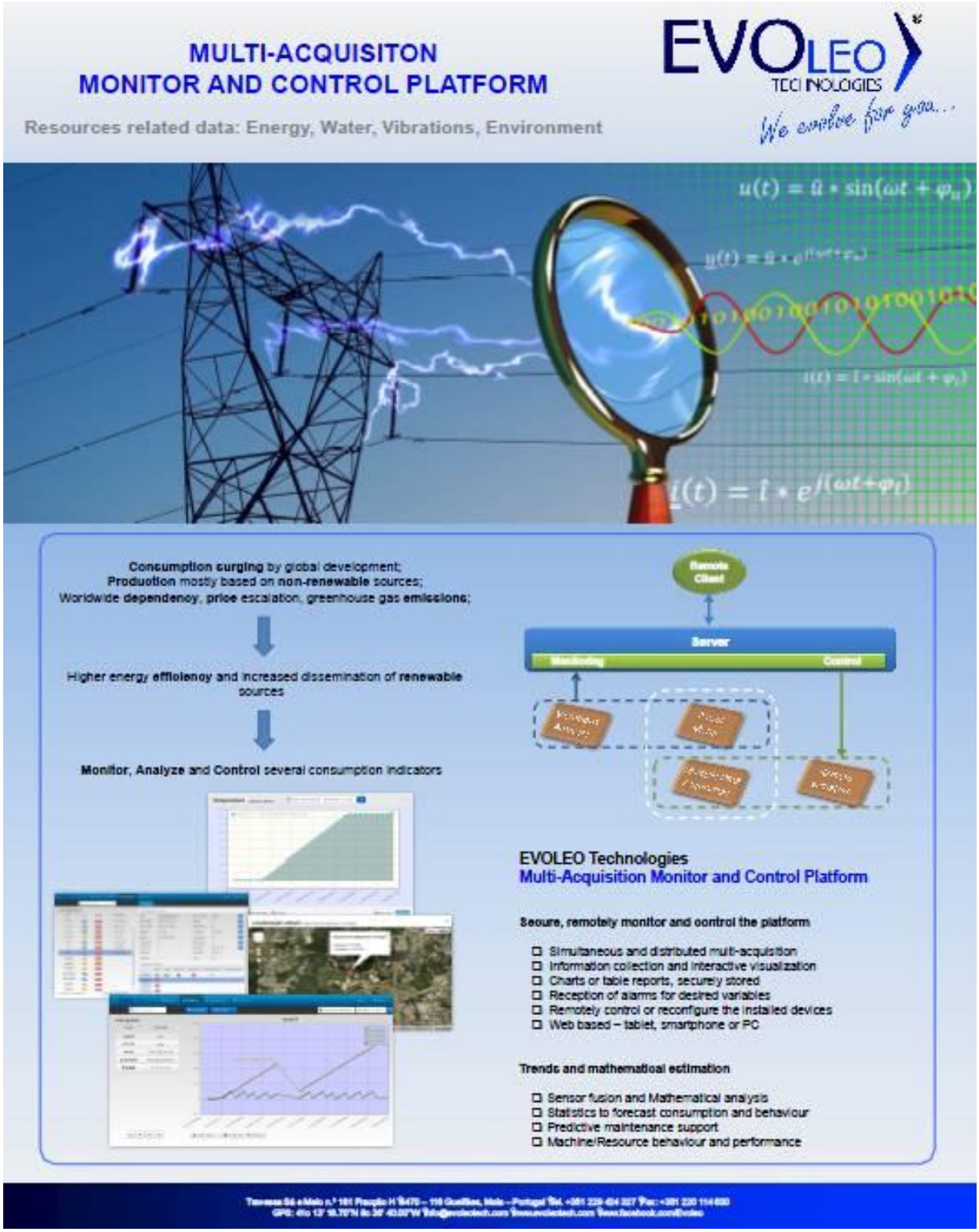


Figura 113 - Poster da participação na conferência EES 2012.