Universidade Atlântica

Licenciatura em Gestão de Sistemas e Computação

Redes de sensores aplicadas à realidade HACCP das empresas de eventos para o controlo de temperaturas

Projeto para o Trabalho Final para a disciplina Laboratório de GSC Aplicado

Elaborado por Hugo Gonçalves Ferreira

Aluno nº 20111471

Orientador: Prof. Doutor Sérgio Nunes

Barcarena, Junho de 2014

Universidade Atlântica

Licenciatura em Gestão de Sistemas e Computação

Redes de Sensores aplicadas à realidade HACCP das empresas de eventos para o controlo de temperaturas

Projeto Final de Licenciatura

Elaborado por Hugo Gonçalves Ferreira

Aluno nº 20111471

Orientador: Prof. Doutor Sérgio Nunes

Barcarena, Junho de 2014

O autor é o único responsável pelas ideias expressas neste relatório

Agradecimentos

Este percurso académico que se encerra agora com a apresentação deste trabalho não teria sido possível sem a ajuda, compreensão e apoio de algumas pessoas que me apoiaram direta ou indiretamente nesta etapa e para as quais eu gostaria de deixar um agradecimento especial.

À minha mulher e família, pelo seu apoio e compreensão, sem os quais não seria de todo possível concluir este projeto;

Ao meu amigo, colega e parceiro nesta etapa Tiago Marques, pelo seu apoio e companheirismo que estiveram presentes desde o primeiro dia;

Ao meu amigo Pedro Silva, pela sua dedicação e empenho com que me acompanhou e ajudou a concluir todas as cadeiras de matemática, sem ele teria sido um desafio de proporções ainda maiores.

À turma de Gestão de Sistemas e Computação, pelo companheirismo e amizade;

À Universidade Atlântica e ao seu corpo docente por me possibilitarem evoluir como ser humano e me transmitirem conhecimentos que me acompanharão ao longo da minha vida;

Por último mas não menos importante, ao meu orientador Prof. Doutor Sérgio Nunes pela sua ajuda e orientação, fundamentais para a conclusão deste trabalho académico.

Resumo

As normas de higiene e segurança da segurança alimentar são cada vez mais, uma realidade com que todas as entidades que operam com alimentos têm de viver. Essa realidade obriga a certos procedimentos de monitorização de pontos de controlo críticos que nem sempre são implementados corretamente por motivos logísticos ou de falta de meios. Este projeto académico pretende apresentar um protótipo para uma solução de baixo custo de monitorização e controlo de temperaturas de uma forma automática para as empresas de organização de eventos.

Palavras-chave: haccp, higiene e segurança alimentar, sensor, redes de sensores em fios, pontos de controlo críticos

Abstract

Safety and hygiene standards of food safety is increasingly a reality with which all entities that operate with food have to live. This reality requires a certain monitoring procedures for critical control points that are not always implemented correctly either because of logistical reasons or lack of resources. This academic project seeks to present a prototype for a low cost solution for automatic monitoring and control of temperatures for event organization area.

Keywords: haccp, food safety, sensor, wireless sensor networks, critical control point

Índice

Índice de Figuras

Capítulo 1
Introdução

O sistema de controlo e segurança alimentar HACCP (*Hazzard Analysis and Critical Control Points*), ou Análise de Perigos e Pontos Críticos de Controlo, é um sistema de gestão alimentar que tem como objetivo prevenir a contaminação cruzada de alimentos na cadeia de produção e processamento alimentar que muitas vezes ocorre no derivado os perigos do seu manuseamento (Vaz, Moreira, & Hogg, 2014).

Um dos pontos críticos de controlo é a monitorização mínima de duas vezes ao dia (Bolton & Maunsell, n.d.), no período da manhã e da tarde, das temperaturas dos frigoríficos e arcas refrigeradoras de forma a conseguir manter um registo das temperaturas de forma a perceber se as mesmas alteraram significativamente podendo prejudicar a conservação dos alimentos.

Na realidade das empresas de eventos, os locais de operação estão muitas vezes afastados da sede da empresa onde se concentram os funcionários nos dias em que não existem eventos e é para essas empresas extremamente dispendioso enviar um funcionário aos locais, duas vezes ao dia para efetuar essa medição de temperaturas. Quanto maior for o número de locais em que a empresa opera, maior será o encargo e a dificuldade de cumprir com os requisitos das normas HACCP.

Para achar uma solução para este problema, que pode levar à tentação das empresas forjarem os valores nos registos simulando visitas aos locais, este trabalho académico pretende procurar uma solução para o problema desenvolvendo um sistema de sensores que podem ser colocados junto dos equipamentos e que comunicarão mais tarde em intervalos regulares, os registos obtidos para um sistema central localizado num ambiente de computação em nuvem onde será possível retirar os relatórios em tempo real, devidamente formatados e preenchidos para serem guardados nos registos de temperaturas do sistema HACCP, caso sejam necessários para uma auditoria das entidades competentes à empresa, fornecendo desta forma uma solução *end-to-end* que possui ainda a possibilidade de alertar em caso de falha de um equipamento.

O sistema pode ainda detetar a falta de comunicação de valores que poderá por exemplo significar uma falha de corrente nos locais onde se encontram os sensores e alertar atempadamente os serviços da empresa.

Um outro ponto em que tentarei manter o foco neste projeto, é o custo acessível dos equipamentos criados/desenvolvidos tentando manter os mesmos a um preço acessível e suportável mesmo por negócio mais pequenos, recorrendo para isso a *hardware* já amplamente testado e utilizado neste tipo de projetos.

O custo de implementação de um sistema de monitorização automático de temperaturas, poderá facilmente chegar a milhares de euros, sendo que os sensores com transmissão por RF/WiFi andam na casa dos 200€ a 500€. Este é um fator que leva a que as empresas se retraiam no investimento em sistemas destes. Adicionalmente, a estes sensores poderemos muitas vezes juntar equipamentos intermédios como *gateways* que aumentam a fatura. A pensar neste constrangimento, proponho-.me neste trabalho a criar um protótipo de sensor de baixo custo, abaixo dos 35€, que tenha a possibilidade de enviar os dados por RF, criando para isso o *hardware* e o *software* necessário à sua operação.

Capítulo 2
HACCP

2.1 Em que consiste o HACCP

Inicialmente desenvolvido pela Pillsbury Corporation com ajuda da NASA para utilização nas viagens espaciais das naves Mercury, Gemini e Apollo, o HACCP (*Hazzard Analysis and Critical Control Point*) é um método que permite analisar e controlar todas as fases da cadeia de processamento alimentar (NASA, 2004).

O princípio base do sistema de gestão alimentar HACCP tem com objetivo primário garantir a segurança dos alimentos identificando os perigos associados ao seu manuseamento e das medidas implementadas que permitem o seu controlo (Vaz, Moreira, & Hogg, 2014).

Este sistema é um sistema de autocontrolo que deverá ser implementado em toda a cadeia alimentar desde o produtor até ao consumidor, fornecendo assim uma ferramenta de análise e prevenção de perigos, que deve ser orientada por evidências científicas dos perigos para a saúde pública (Vaz, Moreira, & Hogg, 2014).

A 1 de Janeiro de 1993, com a liberalização da circulação de mercadorias pela União Europeia, foi efetuada uma perspetiva sobre a implementação e manutenção por parte das indústrias alimentares, um sistema continuado de controlo baseado na metodologia HACCP que culminou na Diretiva 93/43 CEE de 13 de Julho de 1993, relativa à higiene dos produtos alimentares (Vaz, Moreira, & Hogg, 2014).

A implementação de um plano HACCP numa organização, é um processo de várias etapas começando por definir o âmbito de aplicação do plano, formação dos colaboradores, identificação e descrição dos produtos manuseados, identificação do uso desses mesmos produtos, identificação de fluxos dos produtos na organização, identificação de ameaças, identificação dos Pontos de Controlo Críticos (PCC), criação de árvores de decisão nos PCC, identificação dos responsáveis e intervenientes nos processos, controlo e monitorização dos PCC (Vaz, Moreira, & Hogg, 2014).

Segundo Russel Cross (What HACCP Really Means, 2008), o espirito do HACCP complementa o *Total Quality Management* (TQM) e baseia-se numa filosofia “fazer bem a primeira vez, repetir novamente bem e iremos obter um bom produto final”.

2.2 Análise de perigos

A análise de perigos deverá ser efetuada pela equipa HACCP responsável pela implementação do plano e deverá levar a cabo uma análise de perigos que devem ser identificados em relação ao plano HACCP a implementar (Vaz, Moreira, & Hogg, 2014).

Na análise de perigos são equacionadas variáveis como a probabilidade de surgirem perigos e gravidade dos seus efeitos prejudiciais, a avaliação quantitativa e/ou qualitativa da presença de perigos, a análise do índice de sobrevivência ou proliferação de microrganismos envolvidos nos perigos, as condições que podem levar ao aparecimento de toxinas ou agentes físicos nos alimentos e as condições que podem levar a estes fatores (Vaz, Moreira, & Hogg, 2014).

Após serem identificados os perigos, deverão ser posteriormente ser definidas as medidas para a eliminação dos perigos ou redução a níveis aceitáveis de forma a serem produzidos alimentos seguros para a saúde humana (Vaz, Moreira, & Hogg, 2014).

2.3 Pontos de Controlo Críticos

O método para efetuar a medição dos limites críticos dos Pontos de Controlo Críticos é a monitorização ou observações programadas. Através da monitorização deverá ser possível detetar desvios aos limites dos PCC e aplicar as correções necessárias a que sejam ultrapassados estes limites, como temperaturas, análises químicas ou biológicas (Vaz, Moreira, & Hogg, 2014).

Segundo Kelly Bettschen (Monitoring, Verification and Validation, 2012), o processo de monitorização é o ato de:

Conduzir uma sequência planeada de observações ou medições para controlar os parâmetros,

Verificar se os Pontos de Controlo Críticos ou seus pré-requisitos estão sob controlo, e

Produzir um registo exato sobre as atividades de monitorização.

2.4 Medição de temperaturas como meio de controlo dos PCC

As técnicas de monitorização e mediação num plano HACCP, deverão ter a capacidade de detetar situações fora de controlo nos PCC, é desta forma necessário

existirem sistemas que permitam uma deteção atempada de desvios nos limites PCC e alertar de imediato os responsáveis nomeados no plano HACCP para aplicarem as medidas corretivas e evitarem a segregação ou destruição dos produtos afetados, nem sempre tal sendo possível (Vaz, Moreira, & Hogg, 2014).

Capítulo 3
Sensores

3.1 O que é um sensor

Avanços nas tecnologias em áreas como as dos microssistemas eletromecânicos e da eletrónica digital, como melhorias ao nível do *design* e desenvolvimento de sensores de baixo custo e baixo consumo energético, levou a que a produção de sensores de pequeno tamanho e com capacidades de comunicação se tornassem uma realidade. (Akyildiz & Vuran, 2010)

Sensores, quando integrados em estruturas, maquinaria ou mesmo no ambiente, podem de uma forma muito eficiente fornecer um elevado benefício para a sociedade da forma eficaz como recolhem a informação. Benefícios podem ser facilmente obtidos através de menos falhas de nível catastrófico, a conservação de recursos naturais, melhorias ao nível da produtividade fabril, melhorias em casos de respostas de emergência e mesmo ao nível da segurança interna. (Wilson, 2005)

Segundo Winkler (2007), sensores são aparelhos eletrónicos que medem qualidades como a luminosidade ou temperatura e as convertem para uma voltagem. A este processo de passar de uma forma de energia para outra chama-se transdução.

Podemos de uma forma geral classificar os sensores em duas categorias genéricas, sensores digitais e analógicos. Os sensores digitais, funcionam de uma forma binária tendo apenas a possibilidade de dar como output um de dois estados possíveis, ou se encontra ligado (1) por norma transmitindo uma corrente de 5V ou desligado 0V. A maior parte dos sensores digitais funciona utilizando um limiar que define o alcance da medida pela qual vai considerar o estado de *output* (o seu valor de saída) se o limiar for ultrapassado é transmitido uma mudança de estado. (Winkler, 2007)

De forma oposta, os sensores analógicos podem assumir no seu *output*, todo e qualquer valor dentro do alcance resultante das leituras ou medições efetuadas. É muito frequente um sensor analógico ser uma resistência variável que pode ser utilizada para controlar a voltagem. Ao contrário dos sensores digitais que apenas podem alternar entre dois estados, os sensores digitais podem dar como *output* praticamente um alcance de valores infinito. (Winkler, 2007)

3.1.1 Sensores de temperatura e humidade

Existindo no mercado dezenas de soluções de sensores de temperatura, foram selecionados vários sensores que cubram os vários tipos de tecnologia ou implementação para a análise efetuada neste trabalho.

3.1.1.1 DS18B20

O DS18B20 é um termómetro digital de apenas um fio com uma resolução programável que tem a capacidade de transmitir temperaturas em graus Celsius de 9 a 12 bits e possui uma função de alarme com pontos máximos e mínimos programável pelo utilizador. (Maxim Integrated, 2008)

O DS18B20 (Figura 1) utiliza a tecnologia *1-Wire* que permite que as leituras e comunicação com o sensor sejam efetuadas recorrendo apenas a um fio de ligação, sendo os restantes dois contatos a alimentação e terra. (DS18B20 Programmable Resolution 1-Wire Digital Thermometer, 2008)

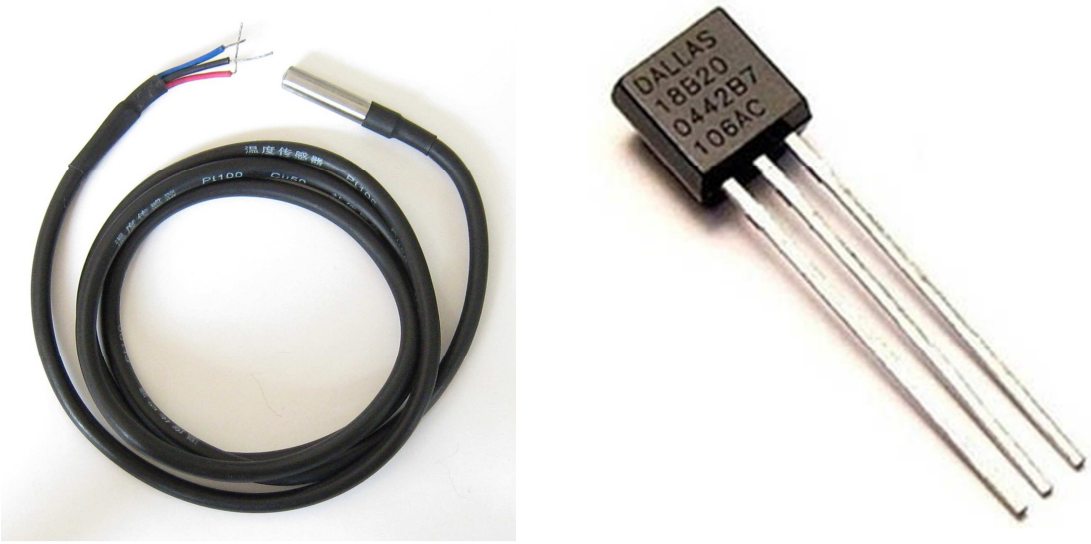


Figura 1 - Sonda/Sensor DS1B20

O sensor DS18B20 tem um alcance na leitura de temperaturas de -55ºC a +125ºC com uma precisão de +-0.5ºC. Adicionalmente cada DS18B20 possui um código de série único que permite que vários sensores DS18B20 funcionem na mesma ligação 1-Wire.

3.1.1.2 DHT11

O DHT11 (Figura 2) é um sensor digital de temperatura e humidade que possui um sinal de *output* calibrado para maior precisão. O sensor inclui uma resistência sensível a humidade e um aparelho de medição de temperatura NTC, ambos conectados a um processador 8 bits de alta performance.

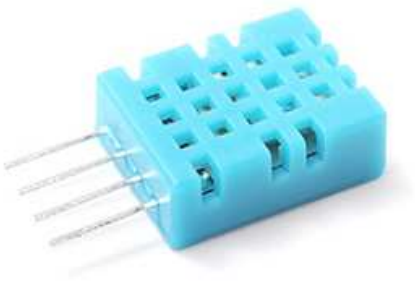


Figura 2 - Sensor DHT11

O DHT11 utiliza um sistema de comunicação série simplificado, onde apenas um dos conetores é utilizado para ler dados e controlar o microprocessador. (Temperature and Humidy Module: DHT11)

3.1.1.3 SHT15

O SHT15 (Figura 3) é um sensor digital de temperatura e humidade, em formato SMD que é fornecido numa pequena PCB soldável para uma fácil utilização. Como caraterísticas principais a Sensirion destaca o seu *output* totalmente calibrado e de alta resolução, um baixo consumo energético e uma estabilidade de leituras a longo prazo. À semelhança das várias soluções apresentadas que permitem leitura simultânea de temperatura e humidade, também o SHT15 utiliza um sistema de comunicação série de um contato para obter as leituras e comunicar com o microprocessador. (SHT1x Humidity and Temperature Sensor IC, 2011)

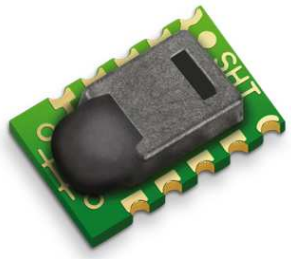


Figura 3 - Sensor SHT15

3.1.1.4 MLX90614

Figura 4 - Sensor MLX90614

O MLX90614 (Figura 4) utiliza a tecnologia de infra vermelhos para obter uma medição de temperatura sem contato. O sensor é composto por dois chips produzidos pela Melexis, o primeiro é um detetor de infra vermelhos termoelétrico e o segundo é o condicionador de sinal ASSP especialmente preparado para processar o sinal de *output* do sensor de infra vermelhos, sendo ambos os componentes assemblados numa embalagem *standard* TO-92. Devido ao amplificador de baixo ruido e de elevada resolução, uma elevada precisão na leitura de temperaturas é alcançada, sendo o sensor calibrado de fábrica e permite uma leitura com uma resolução de 0.01ºC. Os dados são acedidos por dois conetores em serie compatíveis com o protocolo SMBus. O MLX90614 permite ler temperaturas desde -40ºC a 380ºC. (Single and Dual Zone Infra Red Thermometer: MLX90614, 2013)

3.1.1.5 HS1101

O HS1101 (Figura 5) é um sensor de humidade composto por uma única célula capacitiva e possui apenas dois conetores, o primeiro que efetua a alimentação e possibilita a leitura dos valores medidos e o segundo que liga à terra. O HS1101 permite medir a humidade relativa dentro de uma precisão +-5%, permite medir a humidade com um alcance de 1 a 99% operando entre as temperaturas de -40ºC a 100ºC. (HS1101 Relative Humidity Sensor, 2009)

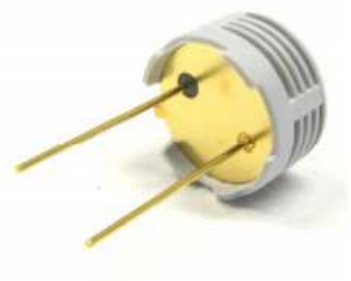


Figura 5 - Sensor HS1101

3.1.2 Sensores de corrente

Os sensores de corrente são desenhados para estabelecer um método económico de monitorizar a corrente elétrica (CR Magnetics). Nos pontos seguintes serão apresentadas algumas soluções e diferentes tipos de implementação para um sensor de corrente.

3.1.2.1 SCT013

O SCT013 (Figura 6) é um sensor de corrente elétrica não invasivo, denominado por *Split-current transformer* na língua inglesa, composto maioritariamente de ferrite, que permite monitorizar e medir a corrente elétrica até 100A. Este aparelho tem a capacidade de registar a corrente elétrica que passa por um cabo sem que para tal tenhamos de efetuar qualquer operação de adaptação no cabo. (YHDC, 2011)

Figura 6 - Sensor SCT-013

3.1.2.2 ACS712

O ACS712 (Figura 7) é um sensor de corrente apresentado num integrado SMD que fornece uma solução de medição de corrente elétrica precisa para utilização em qualquer ambiente industrial. (Allegro MicroSystems LLC, 2013)

Figura 7 - Sensor ACS712

3.1.2.3 FAZ-100

O FAS-100 (Figura 8) é sensor que permite medir a amperagem de corrente, os *watts* de consumo e a voltagem utilizada. Este sensor permite ser ligado a *hub* de leitura ou transmitir as suas leituras através de telemetria para um posto remoto (FrSky, 2012).

Figura 8 - Sensor FAZ-100

3.1.2.4 LF 205-P / SP1

O LF 205-P (Figura 9) é um sensor de corrente que suporta até 200A, que ao contrário do SCT013 não possui uma estrutura em dois blocos que permite adaptar a qualquer cabo sem operações de adaptação adicionais, este sensor é apenas composto por um bloco que pode ser montado diretamente numa PCB. A sua operação funciona através de um circuito fechado em *loop* utilizando o efeito *hall*. Possui uma excelente precisão com tempos de resposta otimizados.

Figura 9 - Sensor LF 205-P

Capítulo 4
Redes de Sensores Sem Fios (RSSF)

4.1 Em que consistem as RSSF

Uma rede de sensores sem fios é composta por dispositivos autónomos e distribuídos por uma área espacial que utiliza sensores para monitorizar as condições físicas ou ambientais. Usualmente, uma RSSF possui um *gateway* que estabelece uma ligação entre a estrutura Sem Fios da rede sensores e a estrutura cablada, utilizada normalmente para enviar os dados para a internet. Existem várias tecnologias utilizadas que são utilizadas conforme os requerimentos aplicacionais, essas tecnologias incluem transmissão rádio a 2.4Ghz onde se baseiam os *standards* IEEE 802.15.4 ou o IEEE 802.11 (Wi-Fi) ou ainda os protocolos de radio proprietários, que operam usualmente na frequência dos 900Mhz (National Instruments, 2012)

Figura 10 - Componentes de uma RSSF (National Instruments, 2012)

É então possível a um sensor interagir com o meio ambiente que o rodeia, processando a informação recolhida localmente ou enviando a mesma por comunicações sem fios para os seus vizinhos (Figura 10).

4.2 Topologias

Os nós de uma rede de sensores sem fios estão normalmente organizados em uma das três topologias mais comuns (Figura 11), topologias em estrela, em árvore ou malha, existindo no entanto outras topologias possíveis (Lewis, 2004).

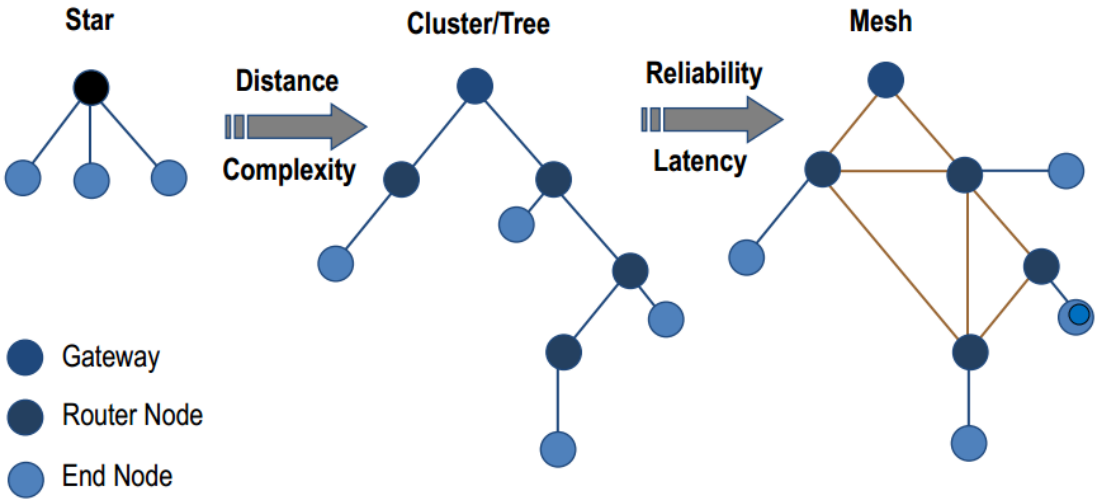


Figura 11 - Topologias de RSSF (Berger, 2009)

4.2.1 Topologia em estrela

Uma rede em estrela (*star network)* é uma rede em que uma única estação base tem a possibilidade de receber ou enviar informação para um número de nós. Esses nós no entanto não possuem a possibilidade de comunicarem diretamente entre si tendo toda a comunicação que passar pela estação base (Wilson, 2005)

4.2.2 Topologia em árvore

Numa topologia em árvore, cada nó conecta diretamente ao nó acima de si na topologia, reencaminhando todo o tráfego para este nó. Esta operação repete-se até atingirmos o *gateway* (National Instruments, 2012).

4.2.3 Topologia em malha

Uma rede em malha (*mesh networks)* é caraterizada pela capacidade de qualquer nó da rede poder comunicar com todos os nós dessa mesma rede que estejam ao alcance da sua capacidade de transmissão radio. Desta forma se um nó quer enviar uma mensagem para um nó que esteja fora do seu alcance radio, terá que utilizar nós intermédios para

reencaminhar a mensagem até chegar ao destino. As redes em malha são normalmente também designadas por *peer-to-peer* (Wilson, 2005).

4.3 Tecnologias

As redes de sensores sem fios, como qualquer tecnologia de rede sem fios, utilizam diferentes frequências de transmissão e modulações de forma a codificar os dados a serem transmitidos.

4.3.1 Modulações

A modulação pode acontecer ao nível da frequência ou da amplitude do sinal. A modulação permite-nos transmitir dados analógicos ou digitais através do canal de suporte (*carrier*). Existem três técnicas base diferentes de modulação, a modulação por amplitude de sinal (AM), por frequência de sinal (FM) e por último a modulação da fase do sinal (PM) (Sing & Rice, 2008). Existem outros tipos de modulação como a modulação multinível disponível nos equipamentos modernos de comunicação e outras que não se encontram no âmbito de este trabalho e não serão analisadas.

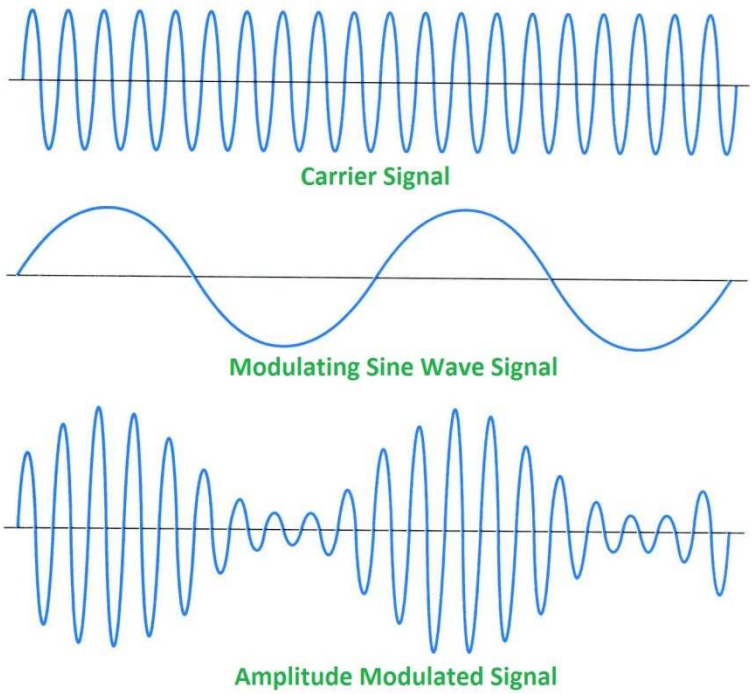


Figura 12 - Modulação da amplitude do sinal (Sing & Rice, 2008)

Através do aumento ou diminuição da amplitude do sinal (Figura 12 - Modulação da amplitude do sinal é possível efetuar uma modulação que nos permite "encaixar" no canal uma sequência de dados que se pretende transmitir (Sing & Rice, 2008).

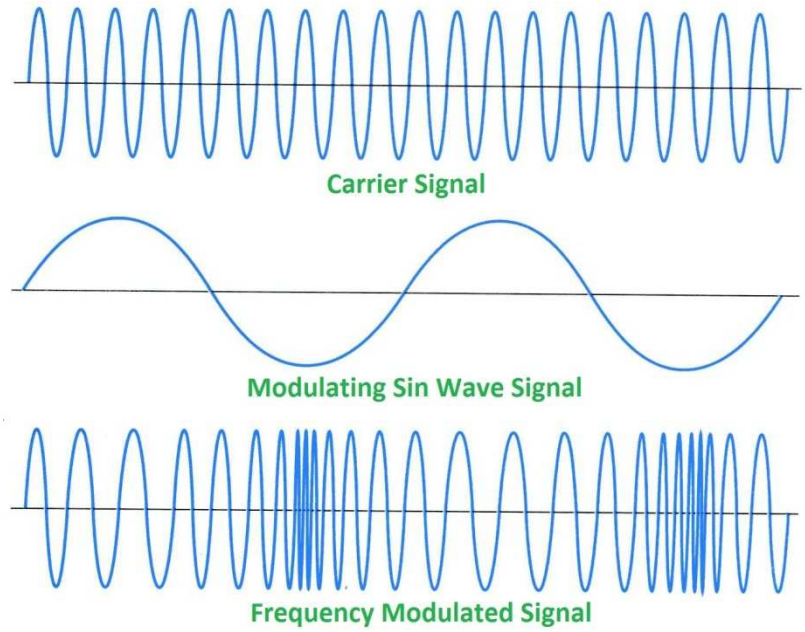


Figura 13 - Modulação da frequência do sinal (Sing & Rice, 2008)

A modulação da frequência do sinal (Figura 13) aumenta ou diminui a frequência do sinal do portador (*carrier*) modulando a onda senoidal possibilitando codificar dados no sinal. Em geral a frequência modulada tende a comportar-se melhor que a modulação de amplitude na presença de ruido (Sing & Rice, 2008).

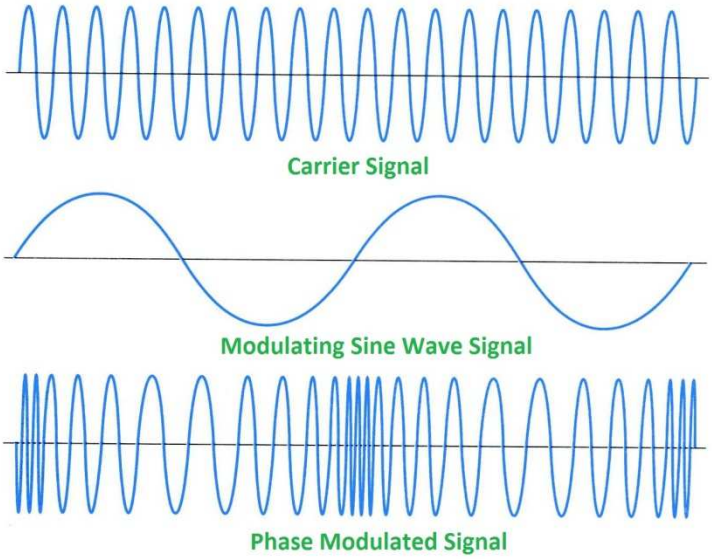


Figura 14 - Modulação da fase do sinal (Sing & Rice, 2008)

A modulação da fase do sinal (Figura 14) é muito semelhante à modulação da frequência onde existe um aumento e diminuição da fase da onda senoidal, para uma melhor compreensão (Sing & Rice) exemplificam com uma analogia onde referem a frequência como a taxa de mudança do angulo da fase, assim como a velocidade é a taxa de mudança para a distância (Sing & Rice, 2008).

Na transmissão de dados com os sinais analógicos, temos de recorrer a uma técnica denominada *Shift Keying* ou *Keying*. Existem fárias formas de *keying* mas três principais se destacam e serão analisadas, sendo as restantes na sua maior parte derivações destas.

4.3.1.1 Amplitude Shift Keying

Amplitude Shift Keying (ASK) uma técnica que utiliza a modulação da amplitude para codificar dados num sinal analógico (Figura 15).

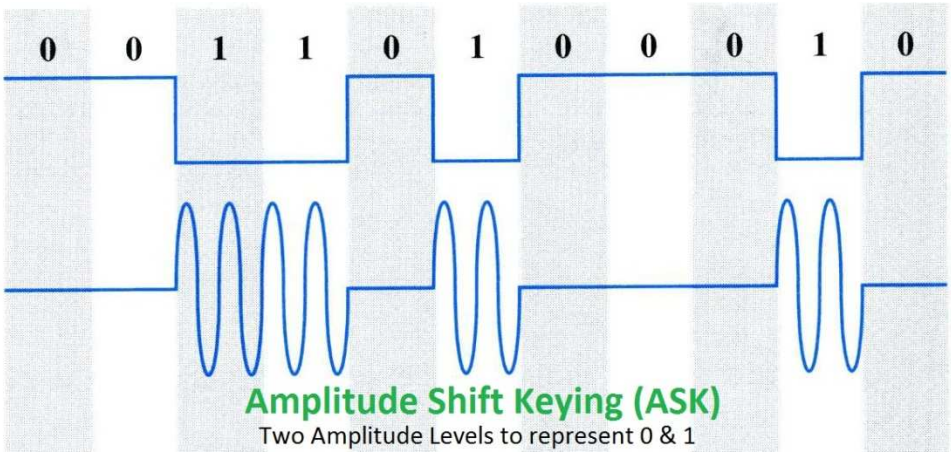


Figura 15 - Amplitude Shift Keying (Sing & Rice, 2008)

Uma das fraquezas ou problemas da ASK é a sensível a impulsos de ruído e não pode ser utilizado para envio de dados a alta velocidade. Poderão existir vários níveis de ASK conforme ilustra a Figura 16 (Sing & Rice, 2008).

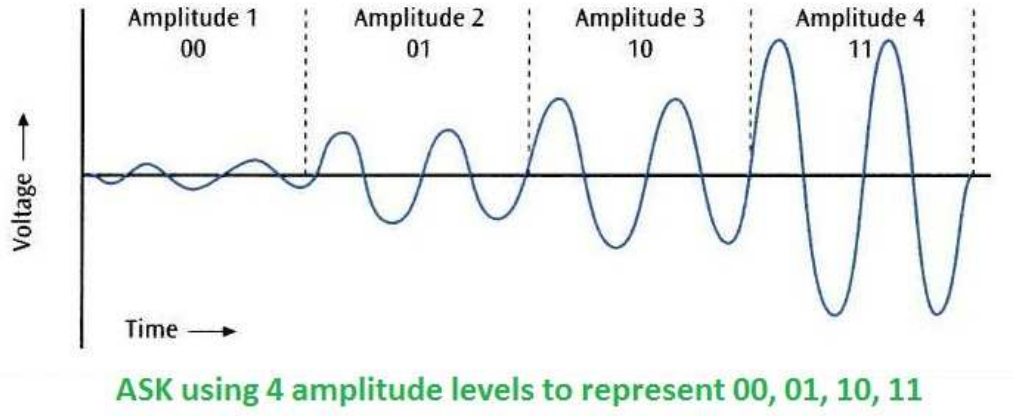


Figura 16 - ASK de 4 níveis (Sing & Rice, 2008)

4.3.1.2 Frequency Shift Keying (FSK)

O método FSK (Figura 17) é uma forma mais simples de modulação de frequência para a transmissão de dados digitais/binários. O FSK tem uma melhor performance que o ASK e não tem problemas com ruido no sinal.

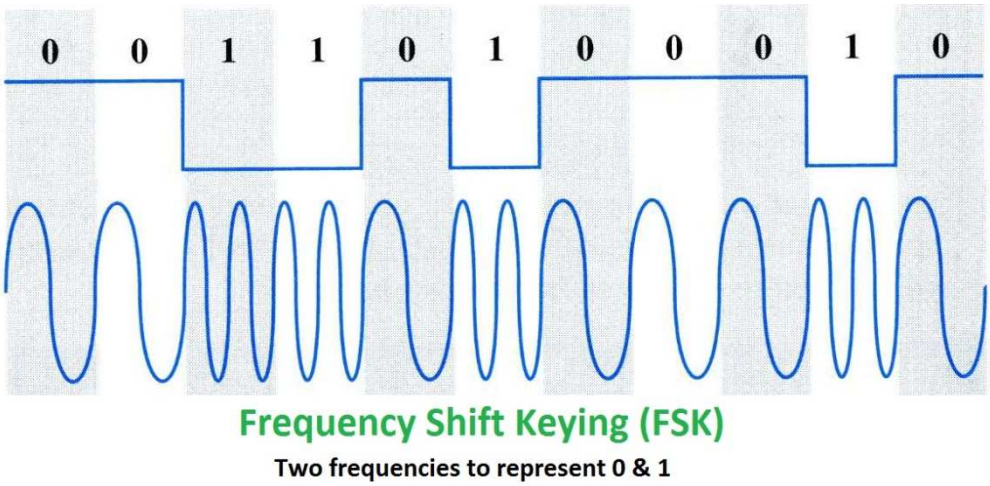


Figura 17 - Frequency Shift Keying

4.3.1.3 On-Off Keying (OOK)

O método *On-Off Keying* (OOK) (Figura 18) é uma forma simplificada do ASK (Maxim Integrated, 2009).

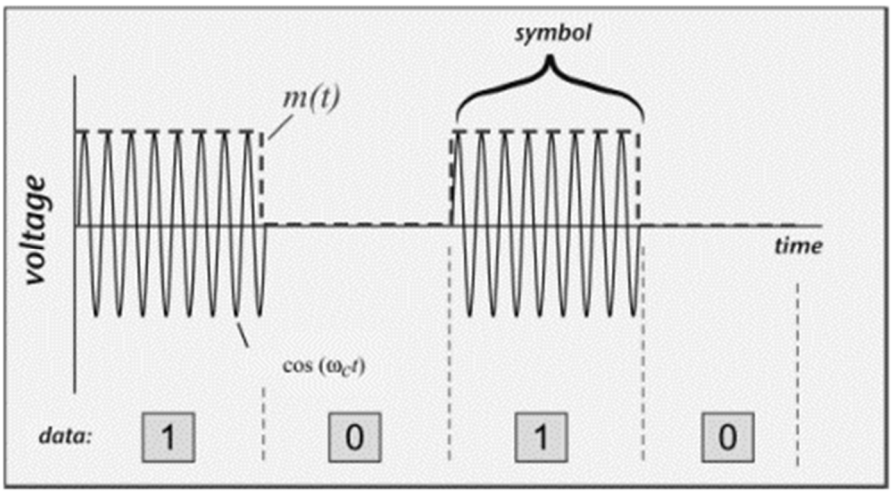


Figura 18 - On-Off Keying (Dobkin, 2007)

Enquanto o ASK transmite uma amplitude larga no sinal do *carrier* quando quer enviar o sinal codificado de ‘1’ e transmite uma amplitude mais pequena para enviar o ‘0’, no OOK quando se pretende transmitir o ‘0’, simplesmente não enviamos sinal de *carrier*.

4.3.2 Frequências e protocolos

As redes de sensores sem fios possuem transmissores e recetores ou transcetores, palavra resultante da junção de transmissor e recetor que significa ambos os aparelhos num único, ou seja, um único chip integrado possui a capacidade de receber e enviar sinal. Estes aparelhos funcionam em várias frequências que podem transmitir dados a uma maior velocidade ou distância de acordo com o método e frequência utilizados. Serão de seguida analisadas algumas destas frequências, tecnologias e aparelhos.

4.3.2.1 RF315/433Mhz

Os módulos que operam na frequência 315/433 Mhz (Figura 19) são usualmente utilizados para comandos de carros, alarmes, portões de garagem, detetores de fumo e outros aparelhos de baixo consumo energético. Sendo uma faz funções em que é mais utilizado como comando para diversas utilidades, é frequente encontrarmos estes sistemas divididos em duas peças de recetor e emissor, mas também existem em *transceivers* com as duas funções num único aparelho. (Maxim Integrated, 2010)

Figura 19 - 315/433Mhz emissor & recetor

4.3.2.2 RFM12B

O HopeRF RFM12B (Figura 20) é um *transceiver* com a possibilidade de receção e emissão de sinal com a capacidade de transmissão de dados com velocidades até 115.2kbps em modo digital e 256kbps em modo analógico. Algumas das particularidades deste moderno *transceiver* estão a capacidade de "acordar" através de temporizador, capacidade de lidar com eventos e capacidade de deteção da qualidade dos dados, entre outros. (Hope Microelectronics Co. Lda, 2006)

Figura 20 - HopeRF RFM12B *transceiver*

4.3.2.3 nRF24L01+ 2.4Ghz

O nRF24L01+ (Figura 21) é um integrado de muito baixo consumo com uma alimentação de 1.9 a 3.6V. Possui integrado um *transceiver* de 2.4Ghz com capacidades de transmissão de dados de 250Kbps, 1Mbps ou 2Mbps que devido à sua capacidade de transmissão de dados e baixo consumo energético já é apropriado para ser utilizado em periféricos de PC, equipamentos desportivos, eletrónica de consumo, entre outros. (Nordic Semicondutor, 2008)

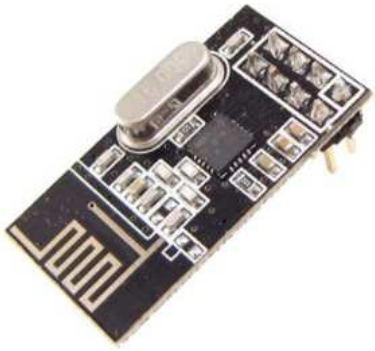


Figura 21- nRF24L01+ *transceiver*

4.3.2.4 ZigBee

ZigBee é uma aliança de várias instituições (Figura 22 e 23) que tem como objetivo promover tecnologias ecológicas, de baixo consumo e abertas à comunidade focadas nas redes de sensores sem fios. Qualquer entidade se pode propor para ser membro, no entanto existem três tipos de membros, os promotores, participantes e os adotantes. Os promotores são os membros que têm mais influência na aliança e que de certa forma dirigem o rumo da mesma. (ZigBee Aliance, 2014)

Figura 22 - Promotores da Aliança ZigBee

Os participantes possuem direito de voto nas decisões da Aliança e estão envolvidos no desenvolvimento das especificações, tendo acesso às mesmas antes de estarem disponíveis para utilização geral. (ZigBee Aliance, 2014) (Figura 23)

Figura 23 - Participantes da Aliança ZigBee

4.3.2.4.1 A tecnologia ZigBee

A tecnologia ZigBee é a única que tenta criar um *standard* de suporte a redes de sensores sem fios a operar na tecnologia 2.4Ghz fornecendo especificações e standards para várias áreas de atuação, quer seja consumo em massa, empresarial ou governamental, sempre tendo em conta o baixo consumo e facilidade de utilização.

Figura 24 – *Standards* da tecnologia ZigBee

Os *standards* (Figura 24) desenvolvidos são destinados a várias área da indústria, tais como gestão comercial de edifícios, eletrónica de consumo, gestão energética, cuidados de saúde, domótica, domótica de retalho e telecomunicações.

Algumas vantagens dos *standards* encontram-se por exemplo da interoperabilidade entre equipamentos de diferentes fabricantes, permite aos mesmos focarem-se no produto em vez da tecnologia utilizada, a tecnologia ZigBee permite então diferenciar os produtos dos seus concorrentes e ir resolvendo problemas de desenvolvimento recorrendo à Aliança, que está sempre a melhorar as especificações e por conseguinte também os produtos das empresas que adotaram a tecnologia. (ZigBee Aliance, 2014)

4.3.2.4.2 Especificações

Na base de todos os *standards* criados pela Aliança encontram-se três especificações com objetivos diferentes.

ZigBee

É o *core*, a base de todas as especificações e define redes em malha inteligentes, energeticamente eficientes e com muito baixo consumo. São redes inovadoras, auto configuráveis e auto reparáveis, com baixo custo e muito baixo consumo energético possuindo até nós livres de bateria, aproveitando a energia disponível no meio ambiente.

ZigBee IP

ZigBee IP é primeiro *standard* aberto utilizando o protocolo IPv6 para a implementação de uma solução de rede em malha com ligação direta à internet, assenta na especificação ZigBee e foi construído tendo em mente o suporte do standard em desenvolvimento ZigBee *Smart Energy* que é direcionado especificamente para a monitorização do consumo de água e energia em ambientes domésticos.

ZigBee RF4CE

Esta especificação é uma simplificação da ZigBee e permite comunicação simples e bidirecional entre dois dispositivos para ambientes que não necessitam de uma rede em malha como o que é oferecido pela especificação ZigBee. Requere muito menos memória para operar o que permite implementações com custos mais reduzidos. (ZigBee Aliance, 2014)

4.3.2.4.3 Standards

A base de qualquer *standard* ZigBee está a especificação da poderosa norma IEEE 802.15.4 para utilização física e sem licenciamento de bandas em todo o mundo entre as frequências 2.4GHz (global), 915Mhz (Américas) e 868Mhz (Europa). Tem a capacidade e transmissão de dados 250Kbs utilizando 2.4GHz (16 canais), 40Kbs utilizando os 915Mhz (10 canais) e 20Kbs utilizando os 868Mhz (1 canal).

As distâncias de transmissão situam-se entre os 10 e 1600 metros dependendo da topologia do terreno e as condições atmosféricas e apresentam um consumo energético notavelmente baixo.

Em termos de *standards* a Aliança ZigBee desenvolveu os standards orientados para a monitorização, gestão e controlo de sensores em RSSF de forma a criar um

interface comum para os equipamentos dos diversos fabricantes, deixando o foco destes no desenvolvimento e melhoria do produto em vez de controlo e desenvolvimento de uma tecnologia proprietária de interconexão. (ZigBee Aliance, 2014)

Na Figura 25 temos um resumo dos vários *standards* criados conforme a área industrial ou doméstica utilizável. Não irei entrar em detalhes sobre cada *standard* pois não são objeto de estudo deste trabalho académico.

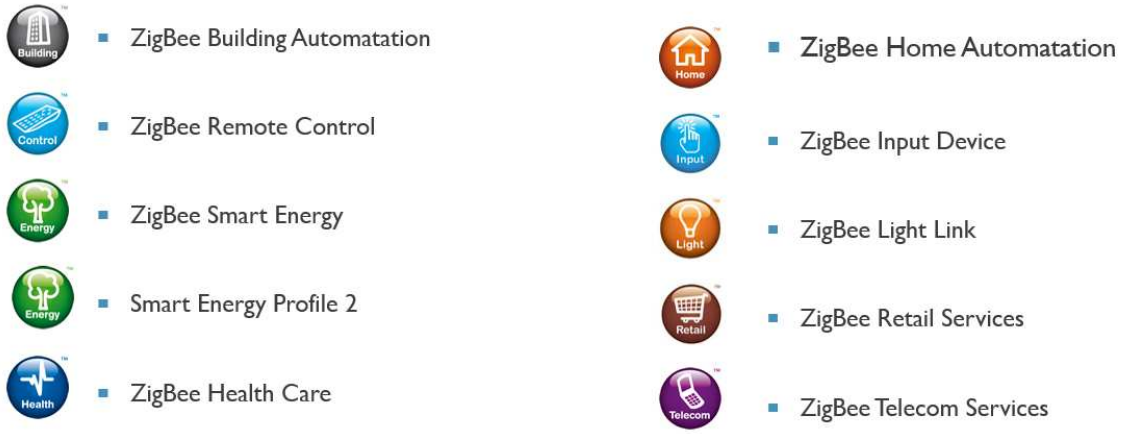


Figura 25 - ZigBee *Standards*

Capítulo 5 - Cloud Computing

A *Cloud Computing* é abordada neste trabalho devido à escolha deste tipo de estrutura para o alojamento do portal de controlo criado em função deste projeto. Até ao advento da computação na nuvem, um sistema alojado num *datacenter*, um servidor, por norma possua recursos dedicados e exclusivos desse servidor. Este conceito acarreta vários problemas:

O *hardware* para um servidor é caro, se queremos pensar na evolução de um serviço teremos que efetuar um investimento inicial mais elevado para dar recursos suficientes para eventuais picos de utilização ou para prever o crescimento do serviço.

Em caso de falha do *hardware*, o serviço fica em baixo ou se possuímos redundância temos no mínimo de duplicar o investimento inicial.

Estes sistemas requerem técnicos especializados que os mantenham.

Estes sistemas não possibilitam um aumento de utilização apenas numa altura no tempo para repor a utilização normal posteriormente.

Para solucionar alguns destes desafios que surgiram e com aparecimento e vulgarização da virtualização de sistemas, surgiu o conceito de *Cloud Computing*.

5.1 O que define um sistema de *Cloud Computing*

A definição de *Cloud Computing* baseia-se em cinco princípios básicos:

- *Multitenancy* ou na sua tradução mais direta, partilha de recursos
- Escalabilidade
- Elasticidade
- Pagamento por recursos utilizados
- Autoaprovisionamento de recursos

(Mather, Kumaraswamy, & Latif, 2009)
(NIST, 2013)
(Cloud Security Alliance, 2013)

5.1.1 Partilha de recursos

Considerando os fatores da nota introdutória deste ponto, pensou-se em criar uma estrutura em que os recursos pudessem ser partilhados por todos os clientes e que garantisse redundância para o funcionamento contínuo das aplicações e serviços. Esta partilha de recursos por vários clientes iria rentabilizar o investimento inicial na estrutura e permitir que todos possam partilhar a estrutura de acordo com alguns parâmetros pré-definidos. São reservados recursos específicos para cada cliente, mas é possível atribuir recursos não utilizados (memória, processador) a outros clientes de uma forma dinâmica, conseguindo desta forma uma utilização mais eficiente do *hardware*, permitindo mais clientes por servidor físico.

5.1.2 Escalabilidade massiva

Um problema de qualquer gestor e do seu administrador de sistemas, é a escalabilidade de um negócio baseado em serviços TI. Se ponderamos logo à partida o suporte de um grande processamento de dados, o investimento será demasiado elevado, por outro lado um investimento fraco em *hardware* irá a curto prazo limitar o crescimento do negócio, não sendo escalável. Uma solução baseada na nuvem acompanha as necessidades do cliente não tendo este que se preocupar com os recursos. A alocação de recursos é da responsabilidade do *provider* e é a este que cabe a responsabilidade de garantir a disponibilidade de recursos contratada e de adicionar novos recursos para aumentar a escala de um negócio de uma forma massiva, alocando novos recursos como processamento, banda ou armazenamento.

5.1.3 Elasticidade

Um outro problema que surge nos negócios e alojamentos tradicionais, é que com a perca de clientes a despesa de manutenção da estrutura mantem-se, ao passo que um crescimento súbito de negócio obriga de imediato de um investimento para suportar este crescimento mas podemos não ter garantias que no futuro o negócio continue de forma a conseguir rentabilizar o investimento feito este aumento rápido ou contração das necessidades do serviço/cliente leva a que os serviços na nuvem tenham uma vantagem pois são elásticos. Ao dizer que são elásticos, estamos a dizer que os mesmos podem crescer rapidamente fornecendo novos recursos praticamente no momento ou poderemos

diminuir os recursos utilizados caso já não necessitemos deles, libertando desta forma despesa (Figura 26).

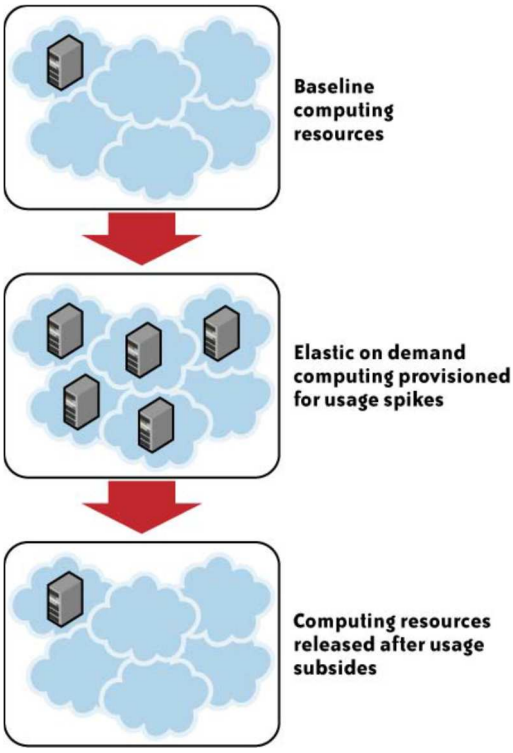


Figura 26 - Elasticidade em Cloud Computing

(Mather, Kumaraswamy, & Latif, 2009)

A elasticidade é uma das características mais interessantes do *Cloud Computing*. Imaginemos uma empresa que através do seu *website* efetua uma campanha em que os clientes tem de preencher um formulário ou efetuar qualquer operação no website da empresa. Esta campanha tem uma aceitação enorme por parte do público e está a sobrecarregar o *website* com os acessos dos clientes tornando todo o serviço lento. Numa solução *Cloud Computing* basta o responsável IT que gere o *website*, aceder ao painel de gestão da solução de *Cloud Computing* e aumentar os recursos do seu alojamento. Com efeito imediato, os recursos ficam disponíveis e o *website* da empresa pode suportar mais uns milhares de *hits* por segundo.

5.1.4 Pagamento por recursos utilizados

Também conhecido por *pay as you go*, é um conceito que assenta especialmente na propriedade da elasticidade do sistema que permite aumentar e reduzir recursos de uma forma muito rápida, permite que os utilizadores paguem apenas os recursos que estão a consumir no momento, não havendo necessidade de investimento avultados para suportar

eventuais picos de utilização. Esta solução permite que se comece com um custo mais baixo para menos recursos e ir aumentando a capacidade e os custos conforme seja necessário.

5.1.5 Autoaprovisionamento de recursos

O autoaprovisionamento de recursos permite ao utilizador sem intervenção de terceiros automaticamente adicionar recursos ao seu sistema, podendo desta forma gerir os recursos necessários para o seu serviço sem a intervenção do *provider.*

5.2 Principais ameaças em Cloud Computing

A *Cloud Security Alliance* realizou em Fevereiro de 2013 um inquérito a especialistas em tecnologias *cloud* e compilou um relatório denominado "The Notorious Nine – Cloud Computing Top Threats in 2013". (Cloud Security Alliance, 2013)

Este relatório debruça-se sobre as principais preocupações na indústria no que diz respeito aos serviços *cloud* e às ameaças que os mesmos estão expostos.

A lista de ameaças por ordem de importância:

1. Divulgação de dados (*Data Breaches*)
2. Perca de dados (*Data Loss*)
3. Sequestro de contas (*Account Hijacking*)
4. APIs Inseguras (*Insecure API's*)
5. Negação de Serviço (*Denial of Service*)
6. Empregados Mal-intencionados (*Malicious Insider*)
7. Abuso dos Serviços Cloud (*Abuse of Cloud Services*)
8. Diligência Insuficiente (insuficiente Due Diligence)
9. Problemas Relacionados com Tecnologia Partilhada (*Shared Technology Issues*)

5.2.1 Data Breaches

Data Breaches ou divulgação de dados, está no top das preocupações relacionadas com Cloud Computing. Segundo este relatório, este é o pior cenário possível para um CIO (Chief Information Officer).

Esta não é propriamente uma ameaça nova, a divulgação de dados por quebras na segurança já acontecia muito antes do advento da *Cloud Computing*, no entanto a globalidade dos serviços *Cloud* e a forma como aglomeram clientes de diferentes áreas e localizações geográficas, tornam este serviços mais apetecíveis a potenciais atacantes e a sua exposição muito maior devido à escala global onde operam.

Segundo este documento, em Novembro de 2012 a Universidade da Carolina do Norte, a Universidade de Wisconsin e a RSA, divulgaram um documento em que descrevem como numa máquina virtual conseguiam utilizar a informação de ajustes temporal de canal para extrair informação criptográfica privada (Zhang, Reiter, Juels, & Ristenpart, 2012).

O documento refere ainda que muitas vezes nem é preciso ir tão longe, uma vez que se a base de dados de utilizadores de um serviço *Cloud* não estiver corretamente desenhada, a falha de segurança na aplicação de um utilizador pode expor os dados de todos os utilizadores desse serviço.

Implicações: É aqui que nos encontramos num ponto em que as soluções para a mitigação do problema, poderão aumentar o próprio problema. Por exemplo, se encriptarmos todos os dados para não permitir o acesso ao mesmo, perdendo a chave privada de encriptação inutiliza todos os dados, por outro lado, se efetuamos *backups* para uma outra localização dos dados podemos estar a aumentar o risco de acesso aos mesmos.

5.2.2 Data Loss

A perca de dados é um cenário que causa grande preocupação tanto a utilizadores como a provedores. Os dados armazenados na *Cloud* podem perder-se por razões técnicas, como por exemplo um provedor acidentalmente apagar os dados de um utilizador ou ainda segundo este documento, por exemplo por uma catástrofe como um

incêndio ou um terramoto que pode destruir os dados de forma permanente, se não tiverem sido tomadas medidas eficazes para o *backup* desses dados.

Mas segundo este relatório, não todas as preocupações sobre este ponto recaem sobre os provedores, muitos utilizadores podem encriptar os dados antes de enviar para o serviço *Cloud* perdendo depois a chave dos mesmos e por conseguinte todos os dados.

Implicações: Segundo a legislação Europeia para a proteção de dados, a destruição ou corrupção de dados pessoais são consideradas como *Data Breaches* e requerem notificações apropriadas do acontecido.

5.2.3 Account or Service Trafic Hijacking

O sequestro de conta acontece quando um utilizador mal-intencionado ganha controlo da conta através de um ataque como por exemplo *phishing* ou outras técnicas utilizadas para obter os seus dados de acesso como por exemplo troianos.

Nesta situação não importa quão boa é a segurança do serviço *Cloud*, porque o atacante utilizando as suas credenciais tem acesso a todos os dados do utilizador.

Segundo este relatório, o reuso de palavras-chave ainda aumenta os danos deste tipo de ataques. Os serviços *Cloud* do utilizador passam a funcionar como plataforma para novos ataques.

Implicações: Com as credenciais de acesso dos utilizadores, o atacante tem acesso a dados críticos dos serviços *Cloud* do utilizador, conseguindo desta forma comprometer a confidencialidade, integridade e disponibilidade dos serviços ou dados.

Como forma de mitigação deste tipo de ataque, recomenda o relatório que as organizações desenvolvam politicas que proíbam a partilha de credenciais entre utilizadores e que implementem formas de segurança baseadas em autenticação de dois fatores.

5.2.4 Insecure Interfaces and APIs

Pela sua própria natureza os serviços de *Cloud Computing* expõem uma série de interfaces e APIs a um acesso público que fica exposto aos mais diversos tipos de ataques.

Existem muitos serviços de terceiros que assentam no acesso a essas mesmas APIs, sem as quais os seus serviços poderão ficar inoperacionais, por exemplo *sites* que utilizem Google *Maps*, Google *fonts*, calendários, GMail, entre outros.

Um outro exemplo é por exemplo *sites* que utilizam os serviços de armazenamento de dados da Amazon ou outro, e que utilizam as APIs para aceder a esses serviços.

Ainda segundo este relatório, os serviços de provisionamento, gestão, orquestração e monitorização são todos efetuados recorrendo a esses interfaces.

Apesar do risco de dependência destas APIs para o negócio, esta ameaça desceu de nível de importância, demonstrando que a maturidade das APIs e interfaces criou uma confiança nos próprios serviços.

Implicações: Apesar do esforço levado a cabo pelos provedores de serviço em manter as suas APIs e interfaces o mais seguros possível, os utilizadores tem de ter a consciência do risco inerente à utilização destes serviços no que diz respeito à confidencialidade, integridade, disponibilidade e responsabilização.

5.2.5 Denial of Service

Os ataques de negação de serviço são ataques que tem como objetivo impedir que os utilizadores consigam aceder aos seus dados e/ou aplicações.

Este tipo de ataque força a vitima a consumir quantidades imensuráveis de recursos até chegar ao ponto em que não consegue dar resposta aos pedidos.

Esta é uma preocupação que não existia no relatório de 2010 e que passou diretamente para o quinto lugar das principais preocupações/ameaças aos serviços *Cloud* demonstrando a importância que este assunto tem vindo a ganhar.

Segundo o relatório, a interrupção de serviço por exaustão de recursos não e a única forma de ataques DoS, existem ataques que exploram falas assimétricas ao nível aplicacional e que tiram vantagem de falhas e vulnerabilidades em servidores web, bases

de dados e outros recursos *Cloud*, permitindo aos atacantes colocar uma aplicação em baixo utilizando um ataque de muito pequena dimensão.

Implicações: Um ataque DoS é comparado por este relatório como ser apanhado num engarrafamento em hora de ponta, não existe nada que possa fazer para chegar ao sues destino, a não ser sentar-se e aguardar.

Enquanto consumidor um ataque DoS pode ser frustrante, mas enquanto cliente de serviços *Cloud* pode levar a que reavalie se realmente mover os seus dados apenas para reduzir custos de infraestrutura vale a pena, face aos prejuízos que pode sofrer por um ataque destes.

5.2.6 Malicious Insiders

Empregados mal-intencionados são uma preocupação que está relacionada com empregados ou ex-empregados dos provedores que tem ou tiveram acesso a todo o tipo de dados sensíveis e que os possam utilizar para fins que lesem o interesse do provedor ou dos seus clientes.

Tem sido amplamente discutido se estes empregados podem ser considerados como uma ameaça, mas na realidade, um empregado com acesso a determinado tipo de informação sensível, pode ser contratado pela concorrência desse cliente ou mesmo do provedor, com o fim de utilizar o conhecimento adquirido nas suas funções.

A CERN criou inclusive uma definição de um *malicious insider*:

"Uma ameaça do tipo '*malicious insider'*, é uma ameaça a uma organização que provém de um empregado atual ou ex-empregado, empreiteiro, ou outro parceiro de negócio que tem ou teve acesso autorizado à rede de uma organização, sistema ou dados e que intencionalmente excedeu ou mal utilizou esse acesso de uma maneira que afeta negativamente a confidencialidade, integridade ou disponibilidade da informação da organização ou dos seus sistemas de informação." (Cloud Security Alliance, 2013)

Implicações: Um *malicious insider* como um administrador de sistema, num sistema *Cloud* mal desenhado pode ter acesso a informação sensível.

5.2.7 Abuse of Cloud Services

Abuso dos serviços *cloud* é uma ameaça relacionada com o problema do poder computacional desses serviços ser utilizado para fins ilícitos, como "*crackar"* uma chave de encriptação ou executar um ataque DoS, distribuir *malware* ou *software* pirata.

Implicações: Mais do que uma ameaça aos dados, é uma ameaça à forma como os provedores tem de se proteger para que os seus serviços não sejam utilizados para fins ilícitos e definir ao certo o que são fins ilícitos e como se controlam.

5.2.8 Insuficient Due Diligence

A *Cloud Computing* despoletou uma série de serviços prestados neste modelo por empresas que prometem redução de custos, eficiências operacionais e maior segurança.

Certamente que em muitas empresas com recursos estes objetivos são bem reais e são atingidos, mas no meio do frenesim de adotar estas tecnologias, muitas empresas saltam para um modelo de prestação de serviços *Cloud* sem terem a verdadeira noção dos recursos necessários para fornecerem este tipo de serviço.

Sem uma noção real das necessidades de um ambiente *Cloud*, suas aplicações ou serviços, as responsabilidades operacionais como resposta a incidentes, encriptação e monitorização de segurança, estão a entrar num mundo em que desconhecem os riscos, riscos que podem até nem compreender.

Implicações: Uma organização que adota serviços *Cloud* sem a diligência necessária para avaliar a mudança, pode deparar-se com um número de situações que não estaria à espera.

Desde cumprimento de SLAs contratuais, transparência e lidar com expetativas frustradas dos clientes, terá de fornecer uma série de garantias e serviços adicionais, que por vezes nem tem a noção de existência dos mesmos.

Se o desenho técnico da estrutura não estiver bem implementado, esta fica vulnerável aos ataques externos, dai o conhecimento necessário ser elevado para poder passar uma estrutura para o modelo *Cloud*.

5.2.9 Shared Technology Vulnerabilities

Os serviços baseados na *Cloud Computing* são escaláveis e elásticos pois baseiam-se na partilha da infraestrutura, plataforma e aplicações entre os vários consumidores apesar da infraestrutura de base (ex. Cache de CPU, GPUs, etc) não ter sido criada para oferecer uma isolação perfeita de recursos.

Falhas na configuração ou mesmo falhas de segurança no código das aplicações podem levar a acesso indevido a dados.

Implicações: Se uma parte da tecnologia Cloud como *hypervisor* for comprometida, mais do que apenas o cliente que levou a que esse *software* tivesse comprometido fica exposto.

Numa estrutura SaaS, compromete todo o ambiente de produção. Esta ameaça é muito perigosa pois pode comprometer toda a estrutura Cloud.

Apesar de este ser a última das ameaças, poderá ser uma das mais importantes. Em Dezembro de 2013, um grupo de técnicos entre os quais um dos criadores do algoritmo RSA, publicou um *paper* no qual descrevem o processo de como através de um simples telemóvel a uma distância de 30cm ou utilizando um microfone a uma distância de 4m, conseguem através de sons emitidos pelo processador, identificar e recolher uma chave de 4096bits em menos de uma hora. (Genkin, Shamir, & Tromer, 2013)

Uma das utilizações imediatas, poderá ser equipa um servidor com um microfone e colocar o mesmo num *datacenter* em que em poucas horas conseguimos obter as chaves privadas de todos os computadores em redor desse mesmo servidor, inutilizando desta forma o sistema de encriptação dos mesmos.

Capítulo 6 - Virtualização

Qualquer sistema baseado em *Cloud Computing* assenta em estruturas virtualizadas, em termos latos virtualização é a capacidade de criar uma versão virtual, não física, de algo.

Uma das características mais conhecidas da virtualização, é a virtualização de *hardware* onde um sistema físico pode virtualizar as suas características e correr simultaneamente vários sistemas virtuais.

Apesar de não ser o único tipo de virtualização disponível, existindo a virtualização de d*esktops*, de aplicações, de *storage* e outras, a virtualização de *hardware* tornou-se popular devido a aplicações desenvolvidas por empresas como a VMWare, Xen (Citrix) e a Microsoft.

A virtualização de *hardware* é possível porque existem programas que emulam os componentes físicos de um computador através de máquinas virtuais (*Virtual Machines* ou VMs), permitindo que num único computador físico possamos correr vários sistemas operativos iguais ou distintos, sem uma perca significativa de performance.

(VMWare, n.d.)

(Bugnion, Devine, Rosenblum, Sugerman, & Wang, 2012)

6.1 Porquê virtualizar?

Um dos problemas que nos dias de hoje nos deparamos no alojamento de servidores e gestão de *datacenters*, é o custo do consumo elétrico, quer dos servidores quer dos sistemas de refrigeração dos *datacenters* e o consumo elétrico de ar condicionados e dos próprios servidores.

Este é realmente um problema se pensarmos que todos os servidores atualmente em funcionamento em *datacenters* consumem energia que é responsável por 340 milhões de toneladas de CO^2 um valor superior aos 173 milhões de toneladas que um país como a Holanda ou a Argentina produzem.

(List of countries by carbon dioxide emissions, 2014)

(Lucente, 2010)

(McKinsey & Company, 2010)

Em 2009, cerca de 0.5% de toda a energia mundial consumida, era por *datacenters* (McKinsey & Company, 2010). Hoje em dia, o custo elétrico num *datacenter* para um cliente consegue suplantar o custo de banda e espaço alocado.

Este fator a juntar ao facto de um servidor nunca estar a ser utilizado a 100% da sua capacidade, ou seja, devido à possibilidade de ser necessário em picos de processamento haver disponibilidade, os servidores terem em média uma disponibilidade 70% superior às necessidades, tornam que o investimento inicial seja mais elevado e que o desperdício de recursos face aos requisitos reais seja enorme. (Mattos, 2008)

A virtualização permite de uma forma simples resolver a grande maioria das questões, desde mobilidade dos servidores virtuais, alta disponibilidade, *backups* eficientes de informação, *snapshots* antes de operações relevantes entre outras vantagens que se tornam rapidamente evidentes numa estrutura de TI.

A virtualização permite-nos de uma forma eficiente gerir todos os recursos do nosso parque de servidores de uma forma inteligente, com controlo e relatórios de performance integrados em painéis centrais que nos dão uma visão completa e global de toda a estrutura em funcionamento. (Mattos, 2008)

Um balanceamento eficiente dos recursos de um servidor físico pelas suas máquinas virtuais, permite otimizar o tempo de processamento do CPU para escalar perto dos 80%, reduzindo assim o rácio custo/disponibilidade necessária. A vantagem da alta-disponibilidade, permite-nos ainda mover máquinas virtuais entre servidores físicos para que possamos sempre balancear as máquinas virtuais pelos servidores com menos carga, distribuindo a carga por uma rede de servidores.

6.1.1 O fator económico

A virtualização permite por uma ínfima parte do valor do custo do *hardware* físico, criar vários servidores virtuais para as mais diversas funções num único servidor físico. Os modernos sistemas operativos de virtualização de máquinas virtuais, permitem uma alocação dinâmica de recursos para que, uma aplicação necessite de mais recursos num determinado espaço de tempo, esta posso usar esses mesmo recursos e libertando-os de seguida para utilização pelas restantes máquinas virtuais do servidor físico.

Este fato permite-nos reduzir consideravelmente o custo despendido na instalação de uma solução que assente em vários servidores. Não necessitamos de ter servidores físicos com tanta disponibilidade livre para contingências, porque essa situação é controlada automaticamente numa rede de servidores físicos com balanceamento das máquinas virtuais.

6.1.2 Alta disponibilidade e fail-over

Ainda que tenhamos o mesmo número de servidores físicos à nossa disposição, a virtualização continua a ser uma vantagem pois podemos facilmente colocar esses servidores num *cluster* em que existe uma distribuição/utilização dinâmica dos recursos por todas as maquinas do cluster, para que quando uma máquina física esteja em esforço, o sistema de gestão de servidores virtuais possa automaticamente mover essa máquina virtual, em tempo real e sem interrupção no funcionamento, para execução num outro servidor físico com menos carga e desta forma garantir que a performance dos servidores virtuais está sempre assegurada.

Uma outra situação a ter em conta é a questão da possível falha de *hardware*. Os modernos sistemas de *hypervisor* permitem *load-balancing* de servidores para que se um sistema físico apresenta falhas no seu processamento ou se falha por completo, os outros nodes físicos do *cluster* assumem de imediato a execução da máquina virtual numa questão de segundos.

Esta vantagem é muito importante para sistemas de alta disponibilidade como bancos ou instituições onde a disponibilidade dos serviços tem de ser assegurada perto dos 100% de disponibilidade e política de redundância tem de ser eficiente e célere na implementação de alternativas. (Mattos, 2008)

Capítulo 7 - Tecnologias Aplicacionais Web

7.1 Em que consistem as tecnologias aplicacionais web

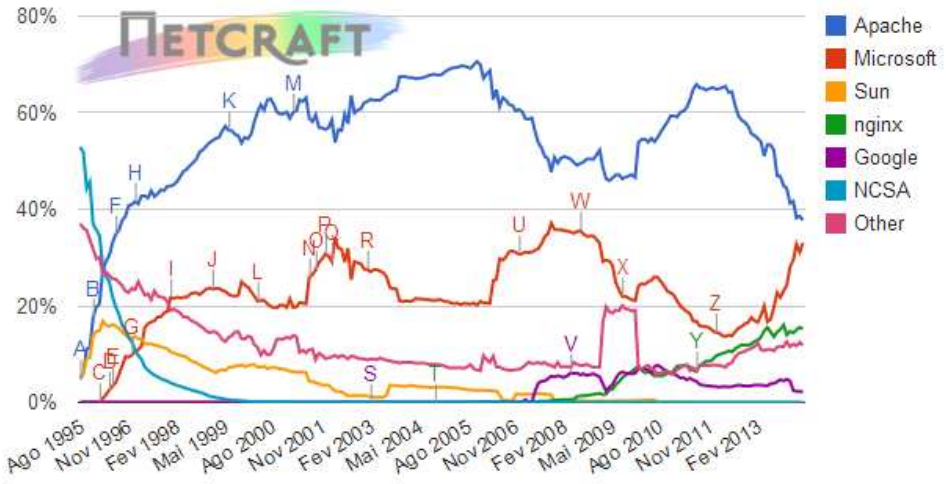
As tecnologias aplicacionais web são constituídas por um conjunto de aplicações que operam num servidor e permitem a execução de determinados serviços, como os descritos abaixo e utilizados neste projeto.

7.1.1 Apache Web Server

Em 1995 o NCSA httpd era o *sofware* servidor de páginas HTTP mais utilizado na *internet*, no entanto o fato de Rob McCool ter saído da NSCA, fez com que o projeto ficasse num impasse e o seu desenvolvimento tivesse parado. (Apache Foudation, n.d.)

Um grupo de *webmaster* juntou-se e começou a desenvolver o software baseado na altura no NCSA httpd 1.3 e sobe o novo nome Apache sendo a sua primeira *release* publica a versão 0.6.2.

Conforme a Figura 27, podemos verificar que desde 1995 o Apache tem sido o *software* mais utilizado em toda a internet como servidor de páginas http (Netcraft, 2014).



Developer	March 2014	Percent	April 2014	Percent	Change
Apache	354,956,660	38.60%	361,853,003	37.74%	-0.87
Microsoft	286,014,566	31.10%	316,843,695	33.04%	1.94
nginx	143,095,181	15.56%	146,204,067	15.25%	-0.31
Google	20,960,422	2.28%	20,983,310	2.19%	-0.09

Figura 27 - Quota de mercado de servidores http (Março 2014)

A utilização em larga escala e o suporte da comunidade, aliado aos anos de funcionamento do produto em ambientes de produção, tornam o produto ideal para a implementação deste projeto.

7.1.2 PHP

PHP: Hypertext Preprocessor teve as suas origens em 1994 quando Rasmus Lerdof criou o PHP/FI, um conjunto de ficheiros binários que funcionava como CGI (*Common Gateway Interface*) e tinha como objetivo o acompanhamento das visitas ao seu *website*. Este conjunto de ferramentas foi mais tarde chamado de PHP *Tools*.

Em Junho de 1995 Rasmus tornou o código de domínio público o que permitiu que vários utilizadores pudessem corrigir bugs e aperfeiçoar o código.

A primeira versão de produção que se assemelha ao PHP atual surgiu em 1997 com a versão 3.0, após o código ter sido largamente reescrito e melhorado. A popularidade do PHP surge de um dos seus maiores pontos fortes, a facilidade de utilização e flexibilidade. (The PHP Group, n.d.)

Segundo a W3Techs, empresa que efetua recolha de dados sobre tecnologias na internet, ao consultar a Figura 28 e 29 constata-mos que a linguagem PHP é utilizada por cerca de 80% dos *websites* da internet. (W3Techs, 2014)

	2013 1 May	2013 1 Jun	2013 1 Jul	2013 1 Aug	2013 1 Sep	2013 1 Oct	2013 1 Nov	2013 1 Dec	2014 1 Jan	2014 1 Feb	2014 1 Mar	2014 1 Apr	2014 1 May	2014 3 May
PHP	78.9%	80.1%	80.5%	80.6%	80.9%	81.1%	81.3%	81.4%	81.6%	81.7%	81.8%	81.9%	81.9%	81.9%
ASP.NET	20.0%	19.9%	19.6%	19.5%	19.1%	18.8%	18.6%	18.4%	18.2%	18.0%	17.9%	17.8%	17.6%	17.6%
Java	4.1%	3.0%	2.9%	2.8%	2.8%	2.7%	2.7%	2.7%	2.7%	2.7%	2.7%	2.7%	2.7%	2.7%
ColdFusion	1.1%	0.9%	0.9%	0.9%	0.9%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%
Perl	0.7%	0.7%	0.7%	0.7%	0.7%	0.7%	0.7%	0.6%	0.6%	0.6%	0.6%	0.6%	0.6%	0.6%
Ruby	0.5%	0.4%	0.4%	0.4%	0.5%	0.5%	0.5%	0.5%	0.4%	0.5%	0.5%	0.5%	0.5%	0.5%
Python	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%
JavaScript	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%

Figura 28 - Progressão das tendências de utilização desde Maio 2013

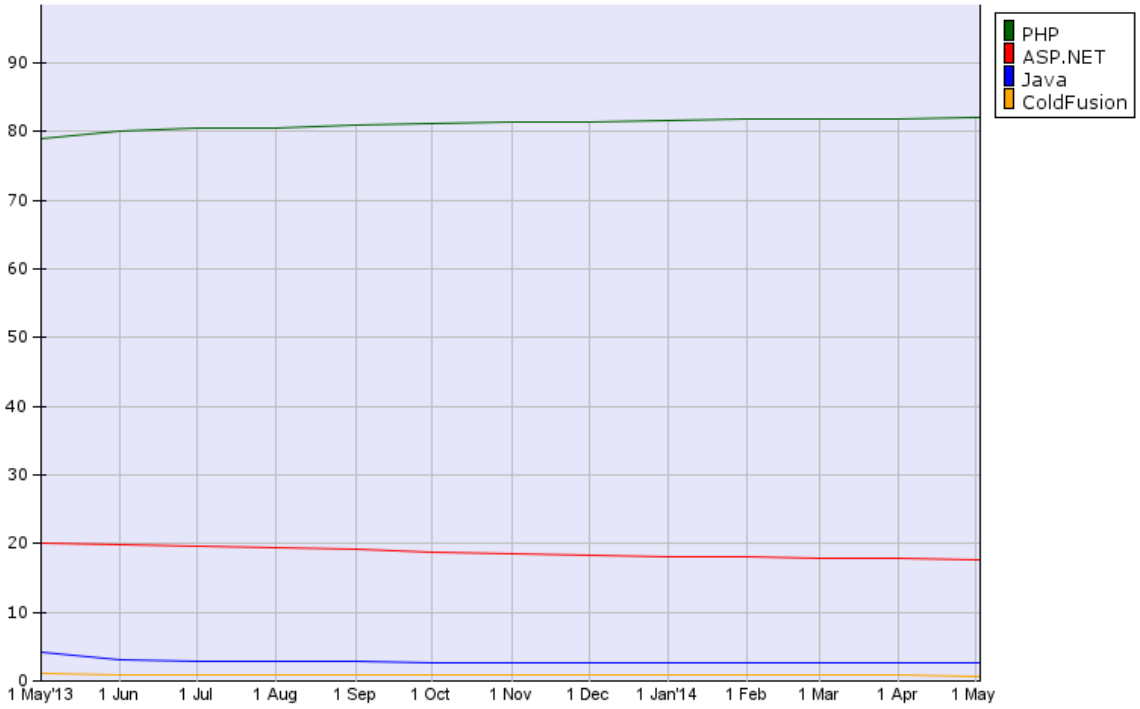


Figura 29 - Linguagens com mais de 1% de utilização

7.1.3 MySQL

O MySQL é a base de dados *open source* mais popular do mundo (DB Engines, 2014) e ao nível do alojamento de websites das mais utilizadas, fazendo parte integrante da maioria dos painéis de controlo disponíveis no mercado.

O MySQL é uma base de dados relacional com uma vasta integração em várias linguagens de programação e um suporte muito ativo por parte da comunidade.

Capítulo 8
Hardware

À parte dos sensores, utilizamos neste projeto microcontroladores ou computadores que fazem parte de um leque de dispositivos de *hardware* de grande popularidade e que analiso um pouco nos pontos seguintes.

8.1 Atmel ATmega 328P

O ATmega é um processador desenvolvido pela ATMEL e utilizado em larga escala para projetos de automação. Entre as suas caraterísticas (atmel, 2012) estão o baixo consumo energético para uma performance elevada, uma *flash* programável de 32Kbytes, com uma EEPROM de 1KByte e 2Kbytes de RAM. Uma das características interessantes do microcontrolador é que suporta um mecanismo de *Read-While-Write Self-Programming* Que permite facilmente escrever no microcontrolador enquanto este se encontra em funcionamento.

8.2 Arduíno

O Arduíno é um microcontrolador desenhado para interagir com uma variedade de sensores para registar os *inputs* produzidos e processar os dados conforme os seus valores para periféricos externos como leds, motores ou altifalantes (Leung, n.d.).

O primeiro Arduíno surgiu no Interactive Design Institute em Ivrea, Itália a partir da tese de um aluno Colombiano de seu nome Hernando Barragan com o título “Arduino – La rivoluzione dell’open hardware”.

Uma equipa de cinco técnicos trabalhos nesta tese e quando a nova plataforma ficou concluída, trabalharam para a tornar mais compacta, económica e disponível para a comunidade *open source*.

O Arduíno, enquanto projeto, aproveitou muitas das potencialidades oferecidas pelos microcontroladores ATmega da Atmel.

8.3 Raspberry PI

O Raspberry Pi é um pequeno computador de dimensões semelhantes a um cartão de crédito, que foi criado a partir de uma ideia comum de Eben Upton, Rob Mullins, Jack Lang e Alan Mycroft da Universidade de Cambridge, que se encontravam preocupados com a diminuição ano após ano do número de alunos que se aplicavam nos cursos de ciências da computação e ainda no número de alunos que obtinham níveis de excelência ao finalizarem esses cursos (Raspberry Pi, n.d.).

A mudança no cenário de como as crianças utilizam os computadores, com a introdução de ambientes gráficos, consolas e outros, fez com que novos programadores começassem a ser escaços num mercado que necessita destes para funcionar.

Em 2008 com os processadores desenhados para aparelhos móveis, foi possível a criação de um PC que utilize um destes processadores e tenha uma boa prestação em multimédia que possibilite ter um pequeno PC com que possamos interagir pelos seus *slots* de expansão GPIO e ensinar eletrónica e programação aos alunos entusiasmando-os novamente com a descoberta de novas tecnologias.

Capítulo 9
Desenvolvimento do Modelo Conceptual

Este projeto foi desenvolvido em várias vertentes, o relatório que se encontra a ler, o *design*, criação e programação do hardware do nó sensor, o design e programação do *gateway* e por fim a programação do portal de controlo.

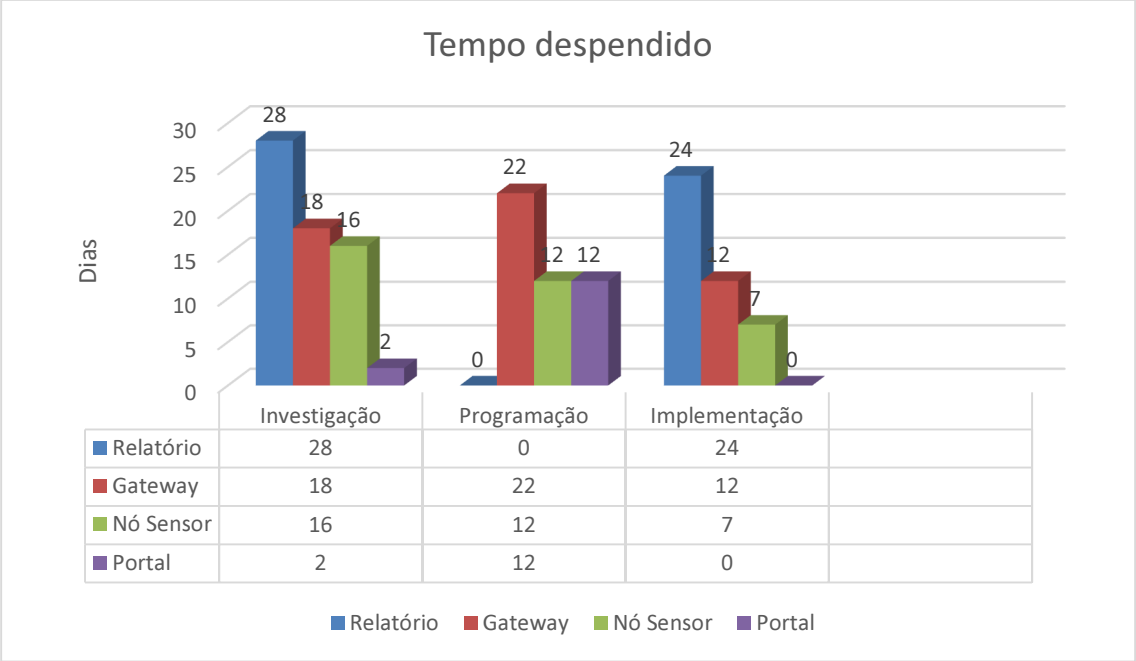


Figura 30 – Tempo despendido em dias

O Gráfico 1 e respetiva tabela ilustram o tempo de desenvolvimento deste projeto, sendo que em investigação inclui-se a revisão bibliográfica, investigação do estado da arte em termos de soluções e investigação para a resolução de problemas durante a implementação dos protótipos utilizados no projeto; programação, refere-se à programação feita em *processing* para programação quer do nó sensor quer do *gateway*, instalação, parametrização e programação do Raspberry Pi como alternativa de implementação do *gateway* e por ultimo a programação do portal; implementação refere-se não só a implementação de *hardware*, soldaduras, preparação de cabos e testes de soluções, assim como a redação deste relatório.

9.1 Estrutura de rede

Para o desenvolvimento desta solução, após uma análise das várias soluções acima revistas, optei por uma estrutura de rede em estrela (Figura 31), podendo no entanto um local físico possuir várias estruturas em estrela a comunicar com o servidor central.

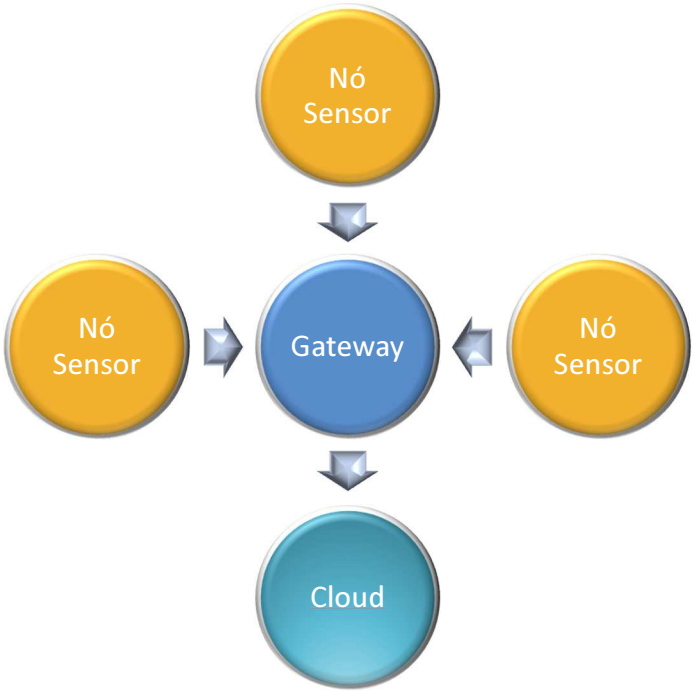


Figura 31 – Modelo concetual da RSSF

Uma vez que é possível ter vários *gateways* em funcionamento paralelo, é possível cobrir uma área mais extensa utilizando apenas alguns pontos de rede, mesmo que existam obstáculos visíveis à passagem do sinal de rádio, mas por sua vez a estrutura em estrela também nos fornece a flexibilidade de monitorizar equipamentos em locais geograficamente perto, necessitando para isso de ir adicionando nós que conectam ao *gateway*.

Na Figura 32 podemos visualizar a implementação técnica dos vários componentes que compõe a solução, desde os sensores num extremo aos servidores, a representar a estrutura em *Cloud*, que suportam a aplicação.

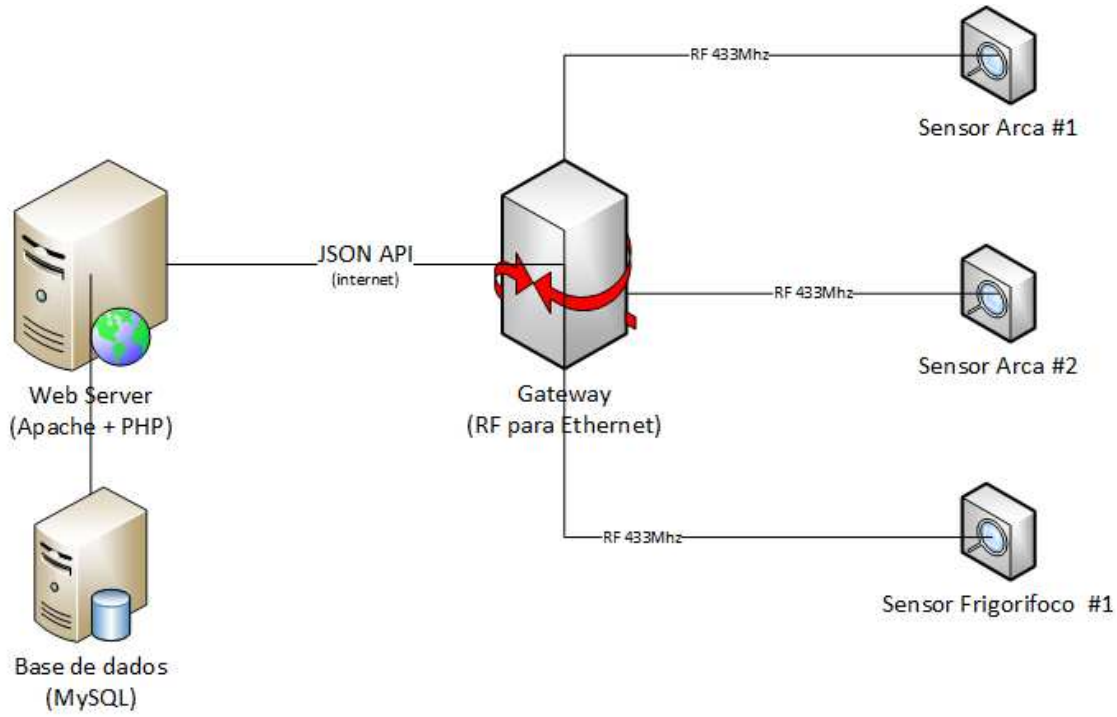


Figura 32 – Esquema técnico da tecnologia utilizada

9.2 Equipamento

9.2.1 Nós sensores

O nó sensor é um equipamento desenhado à medida e implementado exclusivamente para este projeto, como protótipo e prova de conceito.

9.2.1.1 Hardware

Baseia-se num processador ATmega 328P-PU da ATMEL e utiliza um *bootloader* Arduíno para facilitar a programação de código e implementação, recorrendo tecnologias de fácil utilização e com eficácia comprovada.

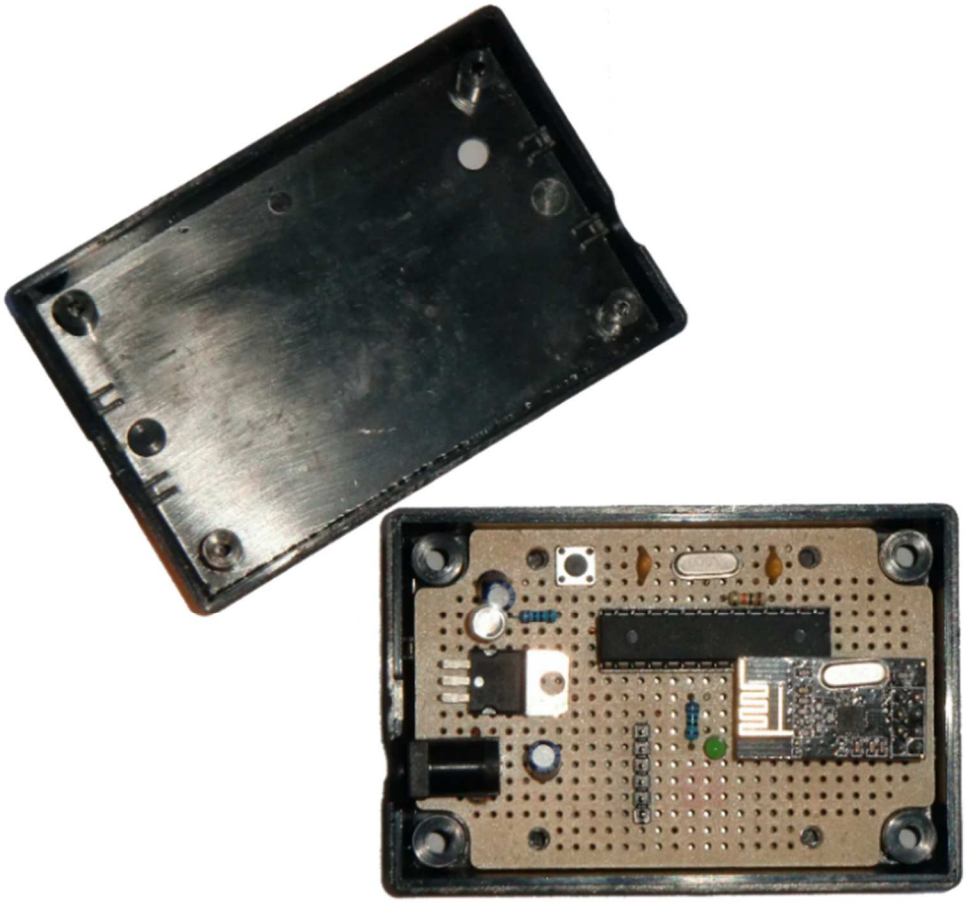


Figura 33 - Nó Sensor (interior)

Figura 34 - Nó Sensor (exterior)

Os componentes foram assemblados numa caixa de pequenas dimensões para ser facilmente acomodada ao pé dos equipamentos a monitorizar (Figura 33 e 34).

Figura 35 - Sensores de temperatura e voltagem

Conforme podemos verificar na Figura 35, o projeto tem como intenção suportar dois sensores que, através de fichas *jack*, que permitem facilmente ligar os sensores ao hardware, utilizando fichas *jack* de 3.5mm para o sensor de corrente e de 2.5mm para o sensor de temperatura.

A distinção nas fichas dos conetores pretende prevenir eventuais problemas que possam vir a ocorrer com a má ligação de sensores.

Inicialmente o nó sensor foi equipado com um módulo RF24L01 da Nordic que permite comunicar a 2Ghz entre o nó sensor e o *gateway*. Uma versão posterior que resultou da alteração do equipamento RF no *gateway* levou à substituição do equipamento de rádio do RF24L01 para um transmissor a 433Mhz FST.

Inicialmente contei colocar dois sensores em funcionamento em cada nó, mas na implementação deparei com problemas graves ao nível da implementação do sensor de corrente que mostraram ser de difícil resolução e para os quais eu não consegui em tempo útil resolver para poder incluir neste projeto.

Estes problemas passam pela não identificação correta da corrente e respetiva calibragem, onde seria necessário alguns meses de pesquisa e desenvolvimento para conseguir obter leituras totalmente corretas, prazo que não foi possível disponibilizar tendo em conta o tempo que outros problemas (ao nível do *gateway*) consumiram no prazo disponível para a realização deste projeto.

9.2.1.2 Código Fonte

A linguagem de programação utilizada para programar o chip ATmega foi o *Processing*, linguagem que é interpretada pelo editor de código do Arduíno a que recorri para auxiliar no desenvolvimento do projeto, abaixo encontra-se o código fonte que permite recolher os dados e enviar os mesmos em formato JSON para o *gateway*. Abaixo encontra-se o código fonte para o nó sensor:

```
#include <VirtualWire.h>
#include <OneWire.h>
#include "EmonLib.h"

EnergyMonitor emon1;

// Cliente ID 1
String clientStr = "1";

// Vamos fazer a leitura dos dados no pino digital 4
int DS18S20_Pin = 4;

// Inicializamos o protocolo Onewire
OneWire ds(DS18S20_Pin);

void setup()
{
  // inicialização do terminal série
  Serial.begin(9600);

  // inicialização do módulo RF
  vw_set_ptt_inverted(true);
```

```
  // vamos fixar a velocidade a 2000bps
  vw_setup(2000);

  // configuramos o pino de dados no digital 3
  vw_set_tx_pin(3);

  // calibração do sensor de corrente
  emon1.current(6, 60);
}

void loop()
{
  // Calcula o Irms do sensor de corrente
  double Irms = emon1.calcIrms(1480);

  // corrente aparente
  Serial.print(Irms*230.0);
  Serial.print(" ");
  Serial.println(Irms);

  // Buffer que guarda a temperatura
  char tempStr[5];

  // obtemos a temperatura
  float temperature = getTemp();

  // convertemos a temperarura para string
  dtostrf(temperature,1,2, tempStr);

  // variaveis necessárias para criar a string json
  String jsonStr;
  char json[100];

  // construção da string json
  jsonStr = "";
  jsonStr.concat("{\"sensor\":\""+clientStr+"\",\"temperature\":\"");
  for (int i=0; i<5; i++)
    jsonStr.concat(tempStr[i]);

  jsonStr.concat("\",\"current\":\"");
  jsonStr.concat("\"}");

  // passamos o valor final para um array
 jsonStr.toCharArray( json, jsonStr.length()+1 );

  // apontador para a mensagem a enviar
  const char *msg = json;

  // enviamos a mensagem
 vw_send((uint8_t *)json, strlen(msg));
```

```
  // aguardamos o final da transmissão de dados
  vw_wait_tx();

  // enviamos para o terminal serie para debug
  Serial.println( msg );

  // fazemos uma pequena pausa para evitar problemas de recepção
  delay(200);
}

/**
 * Retorna a temperatura do sensor DS18S20 em graus centigrados
 */
float getTemp(){
  byte data[12];
  byte addr[8];

  if ( !ds.search(addr)) {
      ds.reset_search();
      return -1000;
  }

  if ( OneWire::crc8( addr, 7) != addr[7]) {
      Serial.println("CRC is not valid!");
      return -3000;
  }

  if ( addr[0] != 0x10 && addr[0] != 0x28) {
      Serial.print("Device is not recognized");
      return -2000;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44,1);

  byte present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);

  for (int i = 0; i < 9; i++) {
    data[i] = ds.read();
  }

  ds.reset_search();

  byte MSB = data[1];
  byte LSB = data[0];

  float tempRead = ((MSB << 8) | LSB);
  float TemperatureSum = tempRead / 16;
```

```
  return TemperatureSum;
}
```

9.3 Gateway

9.3.1 Hardware

O *gateway* foi pensado inicialmente e concebido para operar num Arduíno Uno com um *shield* de rede W5100. Este equipamento foi pensado para ter a capacidade de receber os dados dos nós sensores através de rádio e transmitir essa informação ao servidor central através da placa W5100 que possibilita a ligação a uma rede *ethernet* por RJ45 conforme se encontra ilustrado na Figura 36.

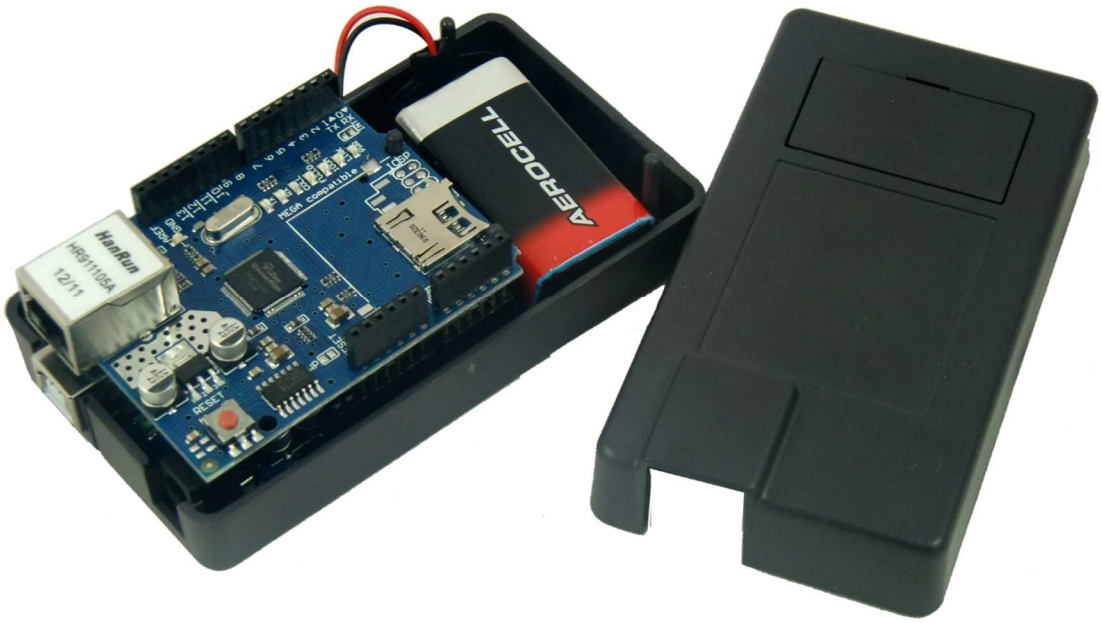


Figura 36 - Gateway

O *gateway* poderia ser alimentado por uma pilha de 9V ou ligado diretamente à rede elétrica através de um transformador (Figura 32).

Para o *gateway* até como se pretende neste momento efetuar apenas uma prova de conceito, optei como já referi por utilizar um Arduíno Uno e um *ethernet shield* W5100 para que possamos obter o desenvolvimento do *gateway* sem um gasto de tempo adicional em implementação de hardware especifico, até porque a implementação de um sistema

que suporte comunicação por rede *ethernet*, requer um tempo de pesquisa que não é compatível com o âmbito do projeto que desejo implementar, desta forma e após a seleção de hardware, passei à fase de programação do equipamento e implementação da solução.

Foi no momento da implementação que me deparei com um sério problema relacionado com a utilização do SPI (*Serial Peripheral Interface*), um interface responsável pelas comunicações entre o processador e periféricos que utilizam comunicação com este, como é o caso do *shield* W5100 e do módulo RF24L01, amplamente documentado na *internet* (Howard, 2013) como acabei por vir a descobrir. A utilização destes dois componentes utilizando este interface demonstrou ser inviável pois o módulo RF24L01 na presença do *shield* W5100 fica inoperacional.

Após uma vasta pesquisa sobe o assunto que consumiu alguns dias de trabalho, deparei-me com um artigo (Shanes.net, 2014) que indicava como poderíamos operar com uma versão do SPI por *software* (SoftSPI) que possivelmente permitiria a operação em simultâneo do módulo RF24L01 e do *shield* W5100, uma vez que ambos utilizam o SPI em simultâneo. Esta solução apesar de devidamente documentada, foi testada durante três dias em várias alternativas sem nunca conseguir uma operação com sucesso.

Face ao exposto foi necessário encontrar uma solução alternativa para prova de conceito que passou por um equipamento mais dispendioso mas que assenta na tecnologia ARM e já possui 512MB de memória, possuindo no entanto um interface GPIO que

permite ligar através de SPI o módulo RF24L01, refiro-me neste caso ao Raspberry Pi (Figura 37).

Figura 37 – Raspberry Pi

O Raspberry Pi é um PC que corre um sistema operativo baseado em Linux (Debian) que foi a versão utilizada nesta implementação.

Com esta nova solução, apesar de necessitar de dois dias para colocar os equipamento operacional (RF24L01), que inicialmente não era detetado pois não estavam a ser carregados os módulos/drivers corretos para a sua operação, consegui após esta fase sem problemas de maior começar a utilizar o módulo RF24L01 conseguindo inclusive operações básicas de deteção de *hardware*, mesmo em linguagens totalmente viradas para a web como é o caso do PHP, que consegui utilizar como linguagem de programação para este dispositivo, linguagem que domínio por completo e sempre ajudou a poupar algum tempo na curva de aprendizagem de uma nova linguagem para operar o *hardware*.

No entanto, e após este sucesso de curta duração, depressa me apercebi que sempre que o módulo RF24L01 era inicializado em modo de comunicação, a placa de *ethernet* do Raspberry Pi deixava imediatamente de operar, inviabilizando desta forma a utilização deste equipamento para o projeto à semelhança do que aconteceu anteriormente com o

Arduíno e o *ethernet shield* W5100, sendo este um problema a juntar ao custo mais elevado do *hardware* no projeto, que tem como objetivo apresentar uma solução acessível economicamente, este hardware foi excluído como opção.

O próximo passo partiu da lógica do processo, se com um Arduino é possível controlar o *ethernet shield* W5100 e com esse mesmo Arduíno também é possível controlar o módulo RF24L01, então podemos colocar dois Arduíno em paralelo, estabelecer comunicação entre eles em série e colocar cada aparelho a controlar um dispositivo, sendo que a informação circula por ambos pelas ligação direta em modo série estabelecida entre os dois.

Para a implementação desta solução, criei um plano para efetuar *design* e desenvolvimento de uma placa de *hardware* que possui dos processadores ATmega 328P-PU, onde o primeiro será responsável pelas comunicações rádio utilizando o módulo RF24L01 e o segundo processador será responsável pelas comunicações *ethernet* utilizando um módulo de rede ENC28J60, um módulo com menos funcionalidades que o *ethernet shield* W5100 mas que é capaz de resolver o problema das comunicações *ethernet* a um custo muito reduzido.

A comunicação entre ambos os sistemas passa então por ligar os dois processadores em modo série, como demonstrado na Figura 38:

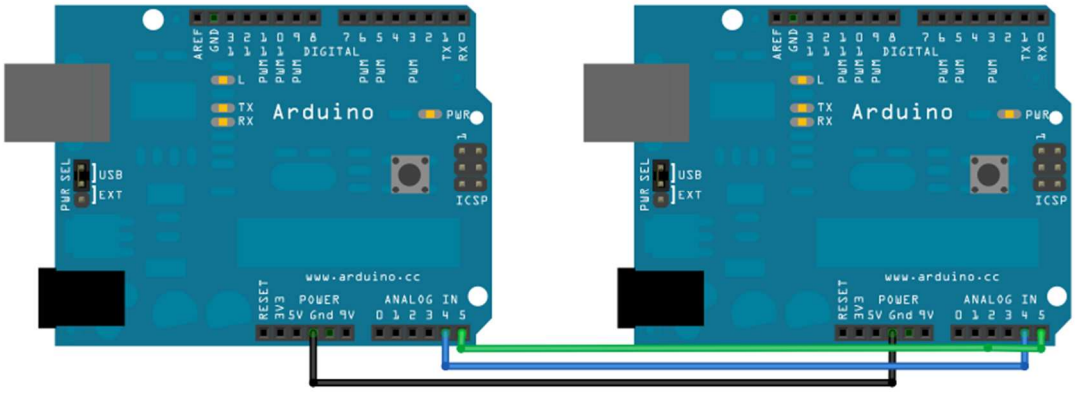


Figura 38 - Arduino em comunicação série

A solução final passa pelo desenvolvimento de um *hardware* à medida, sendo que a solução apresentada na imagem anterior se trata apenas de uma prova de conceito. Para

o desenvolvimento rápido do projeto foi criado um *shield* (Figura 39, 40 e 41) que encaixa diretamente sobre o *ethernet shield* e por cima do Arduíno de forma a conseguirmos o efeito que pretendemos mais rapidamente, mas não necessitarmos de dois Arduínos. Este *shield* não é mais do que um Arduíno minimizado com a capacidade de receber os dados por RF e passar os mesmos por ligação série ao Arduíno que opera o *ethernet shield* e a ligação à internet.

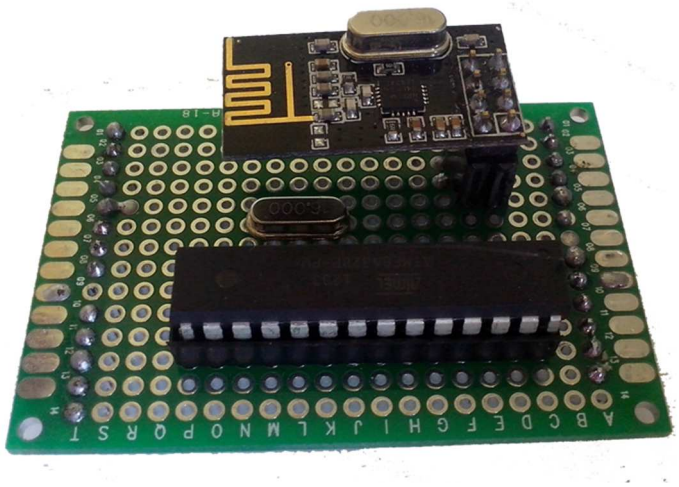


Figura 39 - Shield Arduino + RF24L01

Figura 41 - Shield Arduino + RF24L01 (frente)

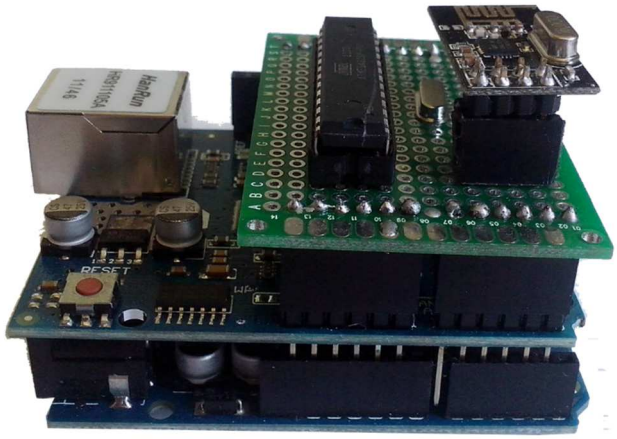


Figura 40 - Shield Arduino + RF24L01 (lado)

Este *shield*, como já foi explicado, é apenas uma prova de conceito sendo que a solução final irá passar pela criação de hardware próprio com dois processadores ATMEGA 328P-PU e um processador de *ethernet* ENC28J60-I/SP DIP dispensando desta forma o *ethernet shield* utilizado nesta prova de conceito.

A solução do segundo Arduíno em paralelo, apesar de funcionar corretamente, não me pareceu a solução mais apropriada e que vá de encontro ao objetivo inicial deste projeto que passa por oferecer uma solução de muito baixo custo, dimensões reduzidas e de fácil operação. A juntar a este problema do custo adicional e de ser uma solução pouco elegante, os módulos RF24L01 mostraram ser de pouca fiabilidade em termos de *hardware* e durante os testes ficaram inoperacionais pelo menos 6 unidades, o que também mostrou ser um problema em termos de fiabilidade do *hardware*, ainda que, enquanto operacionais funcionem com bastante eficiência.

Desta forma, decidi afastar-me mentalmente do *design* e original e dos problemas que o mesmo estava a criar, quer por via de utilização do SPI, quer mesmo pela fiabilidade do *hardware*, e decidi alterar o equipamento de rede para transmissores e recetores RF 433Mhz que necessitam apenas de 3 ligações, corrente, terra e dados, não necessitando de operar dados a uma grande velocidade, a mudança para este tipo de solução não mostrou ser prejudicial em termos de performance e consegui resolver um problema grande que já me havia feito perder largos períodos de tempo em busca de soluções viáveis.

Implicações de mudança de tecnologia RF

Como já expliquei o fato de o transmissor e recetor FST 433 Mhz só necessitar de alimentação e passar a informação utilizando a tecnologia de passagem de dados apenas por um fio, permitiu-me resolver por completo os problemas que estava a ter até ao momento na resolução da questão SPI ou mesmo ter a sincronização de dois ATmega 328P-PU de forma a conseguir gerir a transmissão RF e de dados por *ethernet*. Esta solução mostrou ser bastante estável e funcionar sem problemas com vários nós sensores, sendo que esta alteração trouxe esta vantagem.

A mudança no entanto limitou o projeto em termos de expansão futura, fato que tive que aceitar até conseguir arranjar uma solução “elegante” para o *design* do *gateway*,

tentando que a mesma não tenha que passar por dois microcontroladores em paralelo e a comunicar por série. A limitação que surgir com esta troca de opção de equipamento RF está relacionada com o fato de o novo equipamento ser unidirecional, ao passo que o RF24L01 conseguia comunicação bidirecional, que nos permitia por exemplo consultar os nó sensores em vez de ficar apenas à espera de informação, permitia ao *gateway* ter uma função proactiva a invés de ser apenas passiva à espera de dados. Esta filosofia proactiva, permite por exemplo alterar o nó sensor para integrar um relé e desta forma ligar e desligar remotamente os equipamentos que monitorizamos.

Esta possibilidade de expansão deixou de ser possível com os novos transmissores e recetores de 433Mhz, mas no entanto, tendo em conta um âmbito do projeto, foi uma perca que a meu ver é aceitável pois o que se pretende neste projeto é monitorizar os equipamentos, sendo que por exemplo o controlo remoto seja uma boa característica para melhoramentos em versões futuras.

9.3.2 Código fonte

Em termos de *gateway*, o código foi então reescrito para suportar esta nova tecnologia, sendo nesta altura código que se encontra abaixo, é o código que permite a operação do *gateway*:

```
#include <VirtualWire.h>
#include <Ethernet.h>
#include <SPI.h>

// mac address do gateway
byte mac[] = { 0xDE, 0xAD, 0x02, 0xCC, 0xFE, 0xAA };

// servidor da API
char server[] = "tfc.hugo.webe.pt";

// IP em caso de falha do DHCP
IPAddress ip(192,168,1,177);

// gateway em caso de falha do DFHCP
IPAddress gateway(192,168,1,1);

// Objeto do cliente web
EthernetClient client;

void setup()
```

```
{
  // inicialização do terminal série
  Serial.begin(9600);

  // inicialização do módulo RF
  vw_set_ptt_inverted(true);

  // vamos fixar a velocidade a 2000bps
  vw_setup(2000);

  // configuramos o pino de dados no digital 3
  vw_set_rx_pin(3);

  // inicializamos o modulo para receção de dados
  vw_rx_start();

  // inicializamos o gateway em modo DHCP, em caso de falha em modo
estático
  if (Ethernet.begin(mac) == 0) {
    Serial.println("SERIAL: Failed to configure Ethernet using DHCP");
    Ethernet.begin(mac, ip, gateway);
  } else {
    Serial.println("Ok");
  }

  delay(1000);
}

void loop()
{
  // definimos o tamanho da mensagem/buffer
  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;

  // verificamos se recebemos dados
  if (vw_get_message(buf, &buflen))
  {
    int i;

    // se tivermos recebido dados e enão estivermos conectados
    // conectamos à API
    if (!client.connected()) {
      Serial.println("connecting...");
      if (client.connect(server, 80)) {
         Serial.println("Connected: Getting URL");
         client.print("POST /readings/add");
         client.println(" HTTP/1.1");
         client.println("Host: tfc.hugo.webe.pt");
         client.println("User-Agent: Arduino");
         client.println("Accept: application/json");
         client.println("Content-Type: application/json");
```

```
          client.println("Connection: close");
          client.println();
        } else {
          // avisamos para o terminal em caso de falha
          Serial.println("SERIAL: connection failed");
        }
      }

      if ( client.connected() )
      {
        client.print("POST /readings/add");
        client.println(" HTTP/1.1");
        client.println("Host: tfc.hugo.webe.pt");
        client.println("User-Agent: SensorGateway");
        client.println("Accept: application/json");
        client.println("Content-Type: application/json");
        client.println("Connection: close");
        client.println();

        // mostramos a mensagem recebida no terminal e enviamos
        // por HTTP para o portal
        for (i = 0; i < buflen; i++)
        {
          Serial.write(buf[i]);
          client.print( buf[i] );
        }
        Serial.println(":");
        client.println();
      }

      // exibimos a resposta HTTP no terminal para debug
      if (client.available()) {
        char c = client.read();
        Serial.print(c);
      }

      // Se o servidor tiver desligado paramos o cliente
      if (!client.connected()) {
        Serial.println();
        Serial.println("Disconnecting.");
        client.stop();
      }
    }
}
```

9.4 Servidor/Portal de controlo

O portal de controlo é desenvolvido recorrendo à tecnologia PHP e à *Zend Framework 2*, utilizando como motor de base de dados o MySQL. Foram utilizados para

o portal as ultima tecnologias de desenvolvimento, sendo que foi elaborado utilizando já Bootstrap 3 e jQuery para um visual mais dinâmico.

Na entrada temos um gráfico que nos faculta imediatamente uma visão da geral do comportamento de todos os sensores da plataforma (Figura 42).

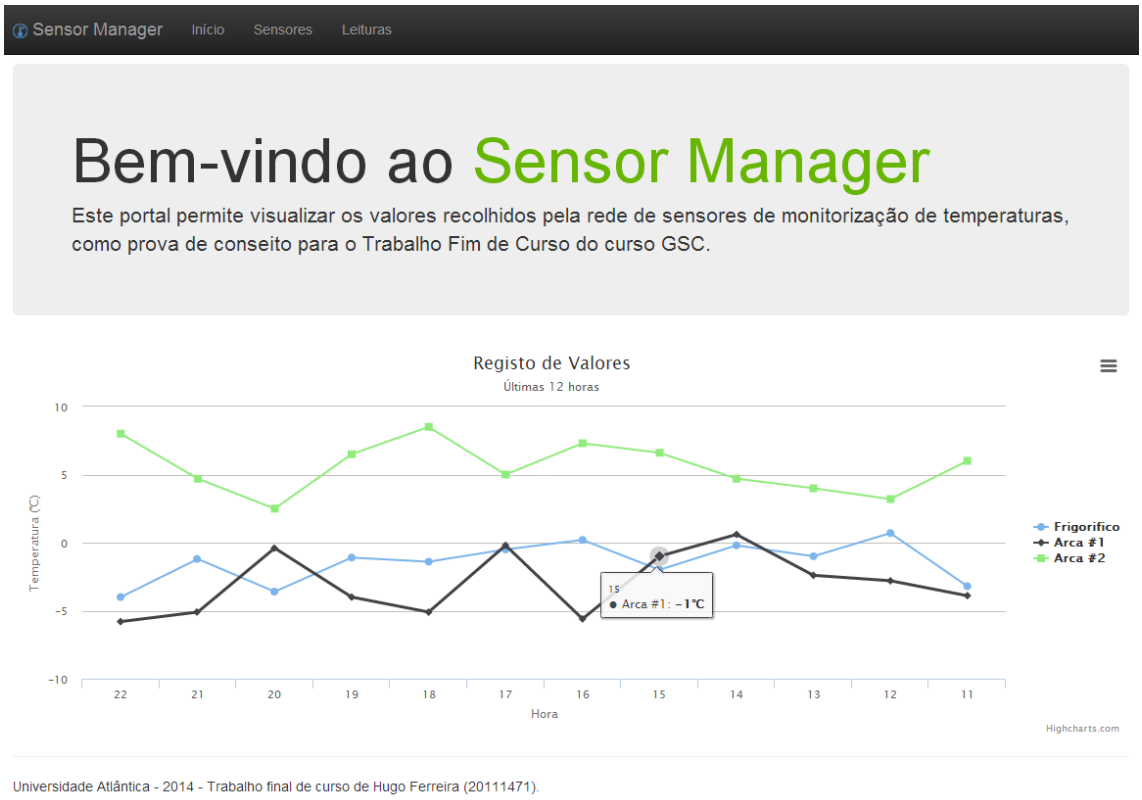


Figura 42 – Entrada do portal de controlo (Sensor Manager)

O gráfico é dinâmico e permite-nos analisar as leituras em cada ponto de forma a possibilitar uma análise mais cuidada dos dados.

O portal possibilita ainda a visualização das últimas leituras realizadas por sensor numa página própria que nos permite navegar por todas as leituras.

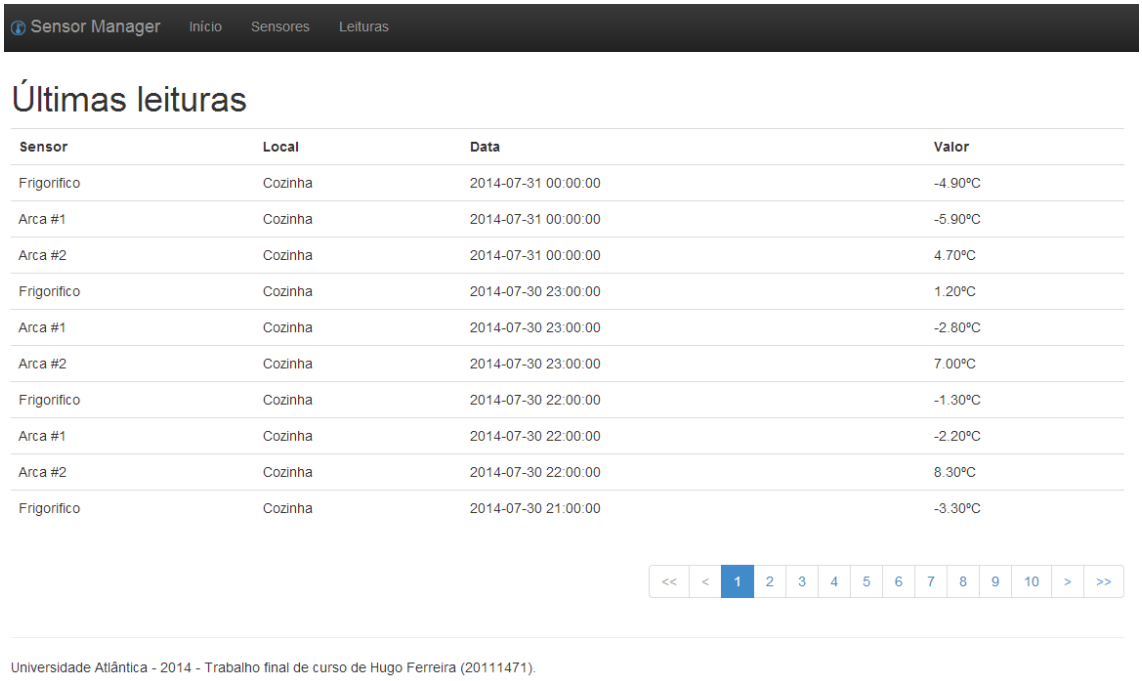


Figura 43 – Página Últimas Leituras

Apesar do sistema estar neste momento a trabalhar apenas com temperaturas, já está preparado para processar valores de sensores de corrente (voltagem), sensores de gás e humidade, adequando o valor às unidades lidas (Figura 43).

É possível gerir dinamicamente os sensores, adicionando ou removendo sensores conforme a nossa conveniência (Figura 44).

Figura 44 – Listagem de Sensores

9.5 API JSON

O portal está equipado com uma API que permite receber a informação através de cadeias de código JSON (*JavaScript Object Notation*) que permite passar cadeias de dados e objetos de uma forma simples e em formato de texto.

O formato de entrada de dados para o sistema é o seguinte:

```
{“sensor”:”<id sensor”, “temperature”:”<graus>”, “current”:”<valor>”}
```

No campo *sensor* é nos fornecido a identificação do sensor no sistema, no campo *temperature* é nos dados em graus centígrados o valor da temperatura e por fim em *current* obtemos o valor em *volts* da corrente elétrica consumida.

Para suportar outro tipo de sensores basta adicionar os campos em questão e adicionar ao portal o suporte para cada um dos tipos de valor obtidos.

Capítulo 10
Linhas futuras e conclusão

Este projeto permitiu-me aplicar todo o conhecimento adquirido pelos meus anos de frequência acadêmica permitindo-me apresentar uma solução de baixo custo para a resolução de um problema real no ramo dos eventos e muitas vezes na restauração onde os relatórios de temperaturas são adulterados.

No geral o sistema é bastante estável e confiável, sendo que as melhorias aplicam-se essencialmente ao nível da melhoria do *hardware* e expansão das funções possíveis no portal de registo.

Como melhoria as próximas versões do *hardware* deverão ser implementadas diretamente em placas PCB desenhadas à medida e feitas e ambiente industrial onde se consegue fazer várias unidades a um preço muito competitivo.

A criação de uma solução mais eficaz para o *gateway* do que a utilização de dois processadores ou a utilização de tecnologia unidirecional 433Mhz, deverá ser conseguida com mais tempo de investigações e desenvolvimento que está para além das possibilidades, do âmbito do projeto e dos recursos disponíveis para este trabalho académico. Caso a utilização de dois processadores seja realmente a opção de futuro, ela não é na realidade uma solução de custo muito elevado, sendo muito mais significativa para o *design* do *gateway* a aquisição do integrado de comunicações ENC28J60 (DIP) para integração da *ethernet* numa única placa sem recurso a *shields* externos. Uma outra solução para esta questão passa por utilizar o módulo ENC28J60 já com a porta RJ45 à semelhança do que foi feito com o módulo RF24L01 em que foi colocado um interface para ligação do módulo.

Em termos de melhorias futuras, seria realmente interessante possibilitar uma comunicação bidirecional com os equipamentos e a integração de relé. O equipamento de nó sensor poderia então ser transformado numa espécie de tomada que liga diretamente à corrente, onde podemos integrar um transformador que alimente o nó sensor e um relé que permita desligar ou ligar remotamente o equipamento monitorizado.

Em termos de *software*/portal web poderão ser criados *triggers* que despoletem ações em *hardware* próprio que permita proactivamente verificar algumas dessas situações relatadas, como por exemplo, se um equipamento futuro tiver relés que permitam ligar e desligar um equipamento, poderemos no portal criar uma zona de agendamentos que ligue e desligue os equipamentos conforme esteja a programação nessa área para o equipamento em questão. Esta característica pode ser interessante porque existem arcas de apoio que só são ligadas alguns dias antes dos eventos e são desligadas após os mesmos, um sistema que automatize esta tarefa ajuda na economia energética e na redução de falhas, por exemplo, esquecimento de ligar um equipamento que será necessário para um evento.

Outra melhoria possível seria a criação de sensores de gás, que permitam por exemplo analisar fugas de gás ou mesmo excesso de monóxido de carbono nas instalações que tenham equipamentos a gás, cozinhas, casa das bilhas, etc.

No portal não foi possível por falta de tempo elaborar um gestor de relatórios que permita retirar do sistema relatórios por data com a identificação do loca e os respetivos valores que é certamente uma melhoria a implementar no futuro, que nesta altura e apenas prova de conceito de todo o sistema, com tempo limitado de desenvolvimento não foi possível explorar.

O potencial de expansão é elevada, penso que consegui atingir o objetivo a que me propus inicialmente de criar um sistema, ainda que protótipo, que permita automatizar as leituras de temperaturas dos equipamentos, ainda que tenha deparado com um problema sério que não consegui resolver na leitura da corrente elétrica/consumo, o objetivo principal foi conseguido, integrando os equipamentos e controlando todos a partir de um portal.

Capítulo 11
Bibliografia

Maxim Integrated. (08 de 04 de 2009). *I'm OOK. You're OOK?* Obtido de Maxim Integrated: http://pdfserv.maximintegrated.com/en/an/AN4439.pdf

Akyildiz, I. F., & Vuran, M. C. (2010). *Wireless Sensor Networks.* West Sussex, UK: Wiley.

Allegro MicroSystems LLC. (2013). *ACS712.* Obtido de Allegro MicroSystems LLC: http://www.allegromicro.com/~/media/Files/Datasheets/ACS712-Datasheet.ashx

AOSONG. (s.d.). *Temperature and Humidy Module: DHT11.* Obtido de ODDWires: http://www.oddwires.com/content/DHT11.pdf

Apache Foudation. (n.d.). *Apache About Page.* Obtido em 03 de 05 de 2014, de Apache Foudation: http://httpd.apache.org/ABOUT_APACHE.html

atmel. (2012). *Atmel 8-bit Microcontroller.* Obtido de Atmel.com: http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

Berger, R. (2009). *Introduction to Wireless Sensor* . Obtido de IEEE Long Island Section: http://www.ieee.li/pdf/viewgraphs/ni_introduction_wireless_sensor_networks.pdf

Bettschen, K. (09 de 2012). *Monitoring, Verification and Validation*. Obtido de Government of Saskatchewan: http://www.agriculture.gov.sk.ca/Default.aspx?DN=3b6394ed-c078-42db-b1ab-879a52a93873

Bolton, D. J., & Manusell, B. (2004). *Guidelines for Food Safety Control in European Restaurants.* Julho: Teagasc - The National Food Centre.

Bolton, D. J., & Maunsell, B. (n.d.). *Guia para o Controlo da Segurança Alimentar em Restaurantes Europeus*. (I. N. Jorge, Editor) Obtido em 15 de Março de 2014, de www.insa.pt: http://www.insa.pt/sites/INSA/Portugues/Publicacoes/Outros/Documents/AlimentacaoNutricao/GuiaControloSegurancaAlimentar.pdf

Bugnion, E., Devine, S., Rosenblum, M., Sugerman, J., & Wang, E. Y. (11 de 2012). *Bringing Virtualization to the x86 Architecture with the Original VMware Workstation.* Obtido em 19 de 04 de 2014, de ACM Transactions on Computer Systems: http://www.cse.iitb.ac.in/~puru/courses/autumn12/cs695/downloads/vmware.pdf

Cloud Security Alliance. (2013). *The Notorious Nine – Cloud Computing Threats in 2013.* Obtido em 19 de 04 de 2014, de Cloud Security Alliance: https://cloudsecurityalliance.org/research/top-threats/

CR Magnetics. (s.d.). *Split-Core Current Transformer.* Obtido de CR Magnetics: http://www.crmagnetics.com/pdf/3110.pdf

Cross, R. (3 de 4 de 2008). *What HACCP Really Means.* Obtido de HACCP Alliance: http://www.haccpalliance.org/sub/food-safety/whatitmeans.pdf

DB Engines. (06 de 2014). *DB Engines Ranking.* Obtido de DB Engines: http://db-engines.com/en/ranking

Dobkin, D. M. (26 de 09 de 2007). *Radio Basics for RFID: Modulation and Multiplexing.* Obtido de EE Times: http://www.eetimes.com/document.asp?doc_id=1276305

Faludi, R. (2011). *Building Wireless Sensor Networks.* USA: O'Reilly.

FrSky. (24 de 04 de 2012). *FAS-100 FrSky Ampere Sensor*. Obtido de FrSky: http://www.frsky-rc.com/download/down.php?id=94

Genkin, D., Shamir, A., & Tromer, E. (18 de 12 de 2013). *RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis.* Obtido em 19 de 04 de 2014, de Universidade de Tel Aviv: http://www.tau.ac.il/~tromer/papers/acoustic-20131218.pdf

Hope Microelectronics Co. Lda. (2006). *RFM12B Universal ISM Band FSK Transceiver*. Obtido de HopeRF: http://www.hoperf.com/upload/rf/rfm12b.pdf

Howard, J. (03 de 04 de 2013). *Arduino Forum.* Obtido de arduino.cc: http://forum.arduino.cc/index.php?topic=158284.0

Leung, K. (n.d.). *Arduino: A Brief History*. Obtido de Keu Leung: http://www.kenleung.ca/_portfolioassets/PDF/HistoryOfArduino_KenLeung.pdf

Lewis, F. L. (2004). *Automation and Robotics Research Institute.* Retrieved from The University of Texas at Arlington: http://210.32.200.159/download/20100130212654891.pdf

List of countries by carbon dioxide emissions. (22 de 03 de 2014). Obtido em 20 de 04 de 2014, de Wikipedia: http://en.wikipedia.org/wiki/List_of_countries_by_carbon_dioxide_emissions

Lucente, E. J. (15 de 06 de 2010). *The Coming 'C' Change in Datacenters*. Obtido em 20 de 04 de 2014, de HPCWire: http://www.hpcwire.com/2010/06/15/the_coming_c_change_in_datacenters/

Mather, T., Kumaraswamy, S., & Latif, S. (2009). *Cloud Securyty and Privacy, 1ª Edição.* California: O'Reilly Media Inc.

Mattos, D. M. (05 de 06 de 2008). *Virtualização.* Obtido em 20 de 04 de 2014, de Universidade Federal do Rio de Janeiro: http://www.gta.ufrj.br/grad/08_1/virtual/index.html

Maxim Integrated. (2008). *DS18B20 Programmable Resolution 1-Wire Digital Thermometer.* Retrieved from Maxim Integrated: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

Maxim Integrated. (2010). *Low-Cost, Crystal-Based, Programmable,*. Obtido de Maxim Integrated: http://datasheets.maximintegrated.com/en/ds/MAX7032.pdf

McKinsey & Company. (03 de 2010). *Energy efficiency: A compelling global resource.* Obtido em 20 de 04 de 2014, de McKinsey: http://www.mckinsey.com/~/media/mckinsey/dotcom/client_service/sustainability/pdfs/a_compelling_global_resource.ashx

Melexis. (28 de 02 de 2013). *Single and Dual Zone Infra Red Thermometer: MLX90614.* Obtido de Melexis: http://www.melexis.com/Asset/IR-sensor-thermometer-MLX90614-Datasheet-DownloadLink-5152.aspx

NASA, L. B. (2004, 08). *NASA Facts.* Retrieved from NASA: http://www.nasa.gov/pdf/71427main_Space_Food_Spinoff_FS-2004-08-007-JSC.pdf

National Instruments. (05 de 2012). *What Is a Wireless Sensor Network?* Obtido de National Instruments: http://www.ni.com/white-paper/7142/en/pdf

Netcraft. (02 de 04 de 2014). *March 2014 Web Server Survey*. Obtido em 03 de 05 de 2014, de Netcraft: http://news.netcraft.com/archives/category/web-server-survey/

NIST. (2013). *The NIST Definition of Cloud Computing.* Obtido em 19 de 04 de 2014, de NIST: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Nordic Semicondutor. (09 de 2008). *nRF24L01+ Single Chip 2.4GHz Transceiver*. Obtido de Nordic Semicondutor: file:///C:/Users/SynerStation/Downloads/nRF24L01P_Product_Specification_1_0.pdf

Parallax. (13 de 04 de 2009). *HS1101 Relative Humidity Sensor*. Obtido de Parallax: http://parallax.com/sites/default/files/downloads/27920-Humidity-Sensor-Documention-S1101-v1.0.pdf

Raspberry Pi. (n.d.). *About Us*. Obtido de Raspberry Pi: http://www.raspberrypi.org/about/

Sensirion. (12 de 2011). *SHT1x Humidity and Temperature Sensor IC*. Obtido de Sensirion: http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT1x_Datasheet_V5.pdf

Shanes.net. (1 de 02 de 2014). *How to use an NRF24L01 (RF24) with an Arduino ethernet shield*. Obtido de shanes.net: http://shanes.net/how-to-use-an-nrf24l01-rf24-with-an-arduino-ethernet-shield/

Sing, J., & Rice, P. (2008). *XTelco*. Obtido de Modulation: Making the Message Fit the Medium: http://ironbark.xtelco.com.au/subjects/DC/lectures/7/

The PHP Group. (n.d.). *PHP History*. Obtido em 03 de 05 de 2014, de The PHP Group: http://www.php.net/manual/pt_BR/history.php.php

Vaz, A., Moreira, R., & Hogg, T. (15 de 03 de 2014). *Rede de Centros de Recursos em Conhecimento*. Obtido de Instituto do Emprego e Formação Profissional: http://www.crcvirtual.org/vfs/old_crcv/biblioteca/manual4/_Manual4.pdf

VMWare. (n.d.). *Virtualization History*. Obtido em 23 de 03 de 2014, de VMWare: http://www.vmware.com/br/virtualization/virtualization-basics/history.html

W3Techs. (03 de 05 de 2014). *Historical trends in the usage of server-side programming languages for websites*. Obtido em 03 de 05 de 2014, de W3Techs Web Technology Surveys: http://w3techs.com/technologies/history_overview/programming_language

Wilson, J. S. (2005). *Sensor Technology Handbook*. USA: Newnes.

Winkler, F. (2007). *Arduino Workshop*. Obtido de Purdue University: http://web.ics.purdue.edu/~fwinkler/590E/Arduino_workshop_sensors.pdf

YHDC. (2011, 07 26). *SCT-013-000 Split-core transformer*. Retrieved from OpenEnergergy: http://openenergymonitor.org/emon/sites/default/files/SCT013-000_datasheet.pdf

Zhang, Y., Reiter, M. K., Juels, A., & Ristenpart, T. (2012). *Cross-VM Side Channels and Their Use to Extract*. Obtido em 19 de 04 de 2014, de Universidade da Carolina do Norte: http://www.cs.unc.edu/~yinqian/papers/crossvm.pdf

ZigBee Aliance. (19 de 04 de 2014). *ZigBee Aliance*. Obtido de ZigBee Aliance: http://www.zigbee.org/