FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Rede de Sensores Sem Fios Para Veículo Frigorífico e/ou Industria Alimentar Indoor

Vasco Sotomaior

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Luís Miguel Pinho de Almeida

18 de Abril de 2013

© Vasco Sotomaior, 2013

Resumo

O transporte de produtos perecíveis em veículos refrigerados requer o controlo do ambiente dentro do contentor, em especial a temperatura ambiente e humidade relativa. A flutuação dos valores fora de gamas específicas pode reduzir a qualidade dos alimentos e criar problemas de saúde pública. Mensurar e gravar toda a cadeia de frio em redes de transporte de alimentos abre a possibilidade de novas formas de gestão, usando os dados recolhidos para a atualização da data de validade dos alimentos transportados e organização dos *stock* de alimentos numa lógica FEFO (*First Expire First Out*), em detrimento do convencional FIFO (*First In, First Out*). As Redes de Sensores Sem Fios (RSSF) são particularmente apropriadas para esta aplicação, devido à sua flexibilidade no posicionamento dos sensores e à capacidade de adaptação a diferentes ambientes, como os encontrados nos recipientes da cadeia de transporte de alimentos perecíveis, com vários tipos de carga, com diferentes tamanhos e características.

O foco principal deste trabalho é a eficiência energética das RSSF, neste caso particular implementada com motes *Iris*. Os *motes* Iris estão programados com o *TinyOS*, um sistema operativo de código aberto destinado para RSSF. Para tempos de execução mais alargados e uso mais eficiente da energia disponível, os motes implementam uma malha de realimentação do RSSI, usado para adaptação da potência de transmissão. Cada pacote recebido pelo *gateway* é depois reencaminhado de volta para o remetente original, que usa o valor RSSI calculado para reajustar a sua potência de transmissão. Além disso, os *motes* entram num estado de descanso entre as tarefas de medição e transmissão, aumentando ainda mais a economia de energia.

Para a avaliação do desempenho energético da rede de sensores implementada, foi montada uma instalação com os motes e o analisado o consumo de corrente. O trabalho realizado mostra que o consumo de corrente associado à comunicação rádio tem um impacto muito reduzido no consumo global da rede de sensores. O fator de maior consumo de corrente é o tempo de atividade de cada nó.

Abstract

The transportation of perishable foods in refrigerated vehicles requires the control of its ambient conditions, in particular the room temperature and relative humidity. The fluctuation of these values outside specific windows can reduce the quality of the food and create public health issues. Monitoring and recording the entire cold chain in food transportation networks opens the possibility for new forms of management C using historical data to upgrade the food expiration date and organizing the food stocks in a logical FEFO (First Expire First Out) instead of the conventional FIFO (First In, C First Out). Wireless Sensor Networks (WSN) are a sensing technology particularly appropriate for this application due to its flexibility on sensors placement and adaptability to different environments as those found in the containers of the perishable food supply chain with multiple types of load with different sizes and characteristics.

The focus of this work is on the energy efficiency of the WSN implemented with Iris motes. The Iris motes are running TinyOS an open-source operating system targeted for WSN. For increased running times and efficient energy use the motes implement an RSSI feedback loop used for transmission power adaptation. Every packet received by the gateway is then replied with its RSSI measurement which is then used by the original sender to readjust its transmission power. In addition the sensing motes enter a sleep state between measuring and transmitting tasks increasing even further the energy savings.

For the evaluation of the energy performance of the implemented sensor network a testbed was setup and the current consumption was monitored. The work undertaken shows that the current consumption associated with the radio communication has a small impact on the overall consumption of the sensor network. The major current consumption factor is the uptime of each node.

"The path of history is not paved like Nevsky Prospekt, it runs across fields,
either dusty or muddy, and cuts through swamps or forest thickets.
Anyone who fears being covered with dust or muddying his boots,
should not engage in social activity."

Nikolay Gavrilovich Chernyshevsky

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ADT	Abstract Data Type
CSMA	Carrier sense multiple access
CSMA/CA	Carrier sense multiple access with collision avoidance
IP	Internet Protocol
OSI	Open Systems Interconnection
MAC	Media access control
FIFO	First In First Out
FEFO	First Expire First Out
WISA	Wireless Interface for Sensor and Actuators
WSN	Wireless Sensor Networks
RSSF	Redes de Sensores Sem Fios
RF	Radiofrequência
SO	Sistema Operativo
TDMA	Time Division Multiple Access
SCM	Supply Chain Management
ERP	Enterprise Resource Planning

Capítulo 1

Introdução

1.1 Motivação

Com o aumento da exigência e preocupação relativamente à qualidade e segurança de alimentos, e no que ao seu transporte diz respeito, aumenta a utilidade e procura de ferramentas que auxiliem o controlo e validação das condições desse transporte, assim como simultaneamente dotem os agentes envolvidos de instrumentos de informação e alarme que impeçam *a priori* a deterioração dos alimentos. O transporte de alimentos em veículos refrigerados é um caso particular de transporte de alimentos, mas altamente frequente nas cadeias de distribuição atuais. Neste tipo de transporte garantir que a temperatura e humidade se mantenham dentro de uma gama de variação reduzida é de importância crucial e o não cumprimento desta condição pode trazer consequências graves tanto do ponto de vista económico, com a deterioração e perda de valor dos alimentos, como do ponto de vista da segurança alimentar, pondo em risco a saúde dos consumidores finais. As especificidades dos contentores frigoríficos representam desafios na recolha e transmissão da informação recolhida, no entanto as redes de sensores sem fios possuem um grande potencial para colmatar essas dificuldades[4, 5, 6].

A monitorização e registo de toda a cadeia de frio nas redes de transportes de alimentos abre também caminho a formas de gestão do transporte de alimentos mais eficientes. Por exemplo o histórico das variações de temperatura e humidade mensuradas em cada transporte, ao qual um dado lote de alimentos esteve exposto, pode ser utilizado para a atualização do seu prazo de validade e assim permitir uma gestão de *stocks* organizada numa lógica FEFO(*First Expire First Out*) em detrimento da convencional FIFO (*First In, First Out*)[7]. Por outro lado a mesma monitorização associada a outras tecnologias como instrumentos de posicionamento (GPS) e a redes de comunicação sem fios de ampla difusão geográfica (HSPA, UMTS, etc) permitem uma gestão integrada e em tempo real de múltiplas cadeias de distribuição [8].

As Redes de Sensores Sem Fios (RSSF) são particularmente apropriadas para esta aplicação, devido à sua flexibilidade no posicionamento dos sensores e à capacidade de adaptação a diferentes ambientes, como os encontrados nos recipientes da cadeia de transporte de alimentos perecíveis, com vários tipos de carga, com diferentes tamanhos e características. Apesar do desenvolvimento

recente das tecnologias associadas aos sistemas embarcados no geral e às RSSF em particular, as limitações energéticas continuam a ser uma das suas principais limitações [9]. Por isso, o desenvolvimento de RSSF tem em particular atenção estas limitações, tendo havido recentemente estudos sobre o impacto que diferentes topologias e diferentes políticas de sincronização e de encaminhamento têm na eficiência energética destas redes[10, 11].

1.2 Objectivos

Com este trabalho pretende-se desenvolver um sistema de monitorização para aplicação num veículo frigorífico de transporte de alimentos, que seja capaz de fazer a aquisição e registo da temperatura e humidade de uma arca frigorífica, assim como detetar a abertura/fecho de porta da mesma. Dada a diversidade de componentes e áreas que o sistema completo abordaria e tendo em conta trabalhos anteriores executados em torno do mesmo conceito e aplicação, este trabalho focar-se-á sobretudo na rede de sensores sem fios, a sua projeção, construção de um protótipo e posterior teste do seu funcionamento num ambiente real, com especial atenção às questões de consumo e conservação de energia em função da topologia de rede adotada. O trabalho a ser desenvolvido no âmbito da dissertação está dividido nas seguintes tarefas e o tempo, ordem e escalonamento das mesmas, que se prevê levar a cabo, encontra-se esquematizado na Figura 1.1.

1. Pesquisa Bibliográfica;
2. Estudo das Redes de Sensores Sem Fios
3. Especificação dos objetivos do sistema
4. Levantamento de plataformas (*Hardware*) orientadas a RSSF
5. Elaboração do relatório de Preparação da Dissertação
6. Projeção do sistema (*Hardware* e software)
7. Desenvolvimento da RSSF
8. Desenvolvimento da Interface
9. Teste das aplicações
10. Análise de resultados
11. Escrita da dissertação

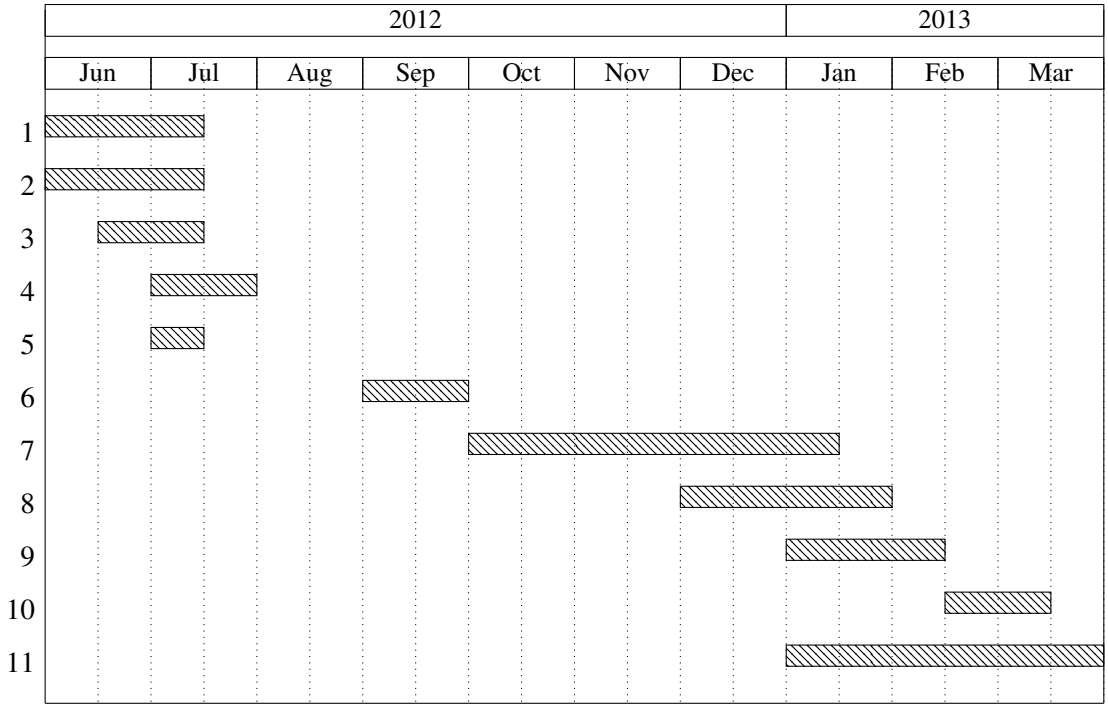

Figura 1.1: Diagrama de Gantt do Plano de Trabalho

1.3 Análise e descrição do Sistema

O sistema deverá ser capaz de fazer a aquisição e registo da temperatura e humidade da arca frigorífica de um veículo de transporte de alimentos durante todo o transporte, assim como detetar e registar a abertura e fecho da porta da arca. Para que o sistema seja de fácil instalação e permita simultaneamente uma distribuição dos sensores, quanto à sua disposição e número, em função do tamanho e forma da arca[12], o tipo de alimentos transportados e a sua disposição dentro da arca, os sensores devem operar sobre uma rede sem fios. No entanto, devido às características da arca frigorífica, a comunicação das medições para fora da arca será feita por cabo. Para tal deve existir dentro da arca um elemento (*gateway*) que recolhe os dados das medições dos sensores sem fios e os envia por cabo para a central presente na frente do veículo (habitáculo). A central será responsável por armazenar os dados e depois de terminado o transporte centralizá-los de forma autónoma numa base de dados presente no local de destino, através de uma rede sem fios *wifi*. Esta central terá ainda um interface gráfico para o condutor, disponibilizando assim informação sobre o estado dos alimentos dentro da arca e emitindo alarmes quando assim se justificar. Todo o sistema deverá ser implementado cumprindo as normas que se apliquem, nomeadamente a NP EN 12830, que regula os registadores de temperatura para transporte, armazenagem e distribuição de alimentos refrigerados. Esta norma especifica que o sensor de temperatura deverá ser capaz de medir temperaturas desde -25^{o}C até 15^{o}C e simultaneamente garantir a uma amplitude de medição superior ou igual a 50^{o}C.

1.4 Estrutura da Dissertação

Para além da introdução, este documento contém mais 5 capítulos. No capítulo 2, é descrito o estado da arte e os principais conceitos relacionados com Redes de Sensores Sem Fios. No capítulo 3, é feito um levantamento das plataformas de *hardware* e *software* existentes e que poderão ser usadas na construção do protótipo. No capítulo 4 é feita a descrição do trabalho desenvolvido para o projeto e implementação da RSSF e da aplicação de interface. No capítulo 5 é feita uma análise dos resultados experimentais obtidos. No capítulo 6 faz-se um levantamento das conclusões tiradas com a execução deste trabalho e uma análise dos objetivos atingidos.

Capítulo 2

Conceitos

2.1 Redes de Sensores Sem Fios

Com o desenvolvimento da tecnologia e da técnica é hoje possível a construção de dispositivos que agregam, num único elemento, instrumentos de medição, micro-controladores, interfaces RF e fontes de energia por um custo reduzido. Apesar de do ponto de vista individual estes dispositivos serem limitados, tanto a nível do alcance das comunicações como da capacidade de processamento, a sua agregação em redes colaborativas abre um enorme conjunto de possibilidades. Estas redes são conhecidas como Redes de Sensores Sem Fios, ou na literatura anglo-saxónica como *Wireless Sensor Networks*, e são hoje aplicadas nas mais variadas áreas: agricultura, monitorização ambiental, monitorização de gado, controlo fronteiriço, deteção de violação de segurança em aplicação militares, medição de sinais vitais, monitorização de linhas de montagem etc [13]. A natureza colaborativa das RSSF confere-lhes simultaneamente flexibilidade, tolerância a falhas e fácil implementação, que associadas ao baixo tamanho e custo as torna interessantes para aplicação na monitorização das cadeias de frio, tal como foi proposto por Qingshan et al [14].

2.1.1 Arquiteturas de RSSF

Os sensores numa RSSF estão normalmente espalhados geograficamente. Cada um deles tem a capacidade de recolher dados e centralizá-los numa estação central (*gateway*), a partir da qual estão disponíveis para os utilizadores. As várias camadas por onde se realiza a comunicação entre os sensores e o *gateway* podem ser decompostas segundo o modelo OSI em camada de ligação de dados, camada de rede e camada de transporte [15].

É nestas camadas onde se encontram alguns dos principais desafios ao desenvolvimento e implementação de RSSF, alguns dos quais transversais a várias camadas. A divergência no tipo de organização de rede por exemplo, entre uma lógica centrada no endereço, comum nas redes tradicionais, e a natureza das redes de sensores, em que grupos de sensores partilham áreas geográficas comuns e a identificação de cada um é uma identificação geográfica, relativamente ao resto do grupo ou ao *gateway* da rede, em detrimento de um identificador lógico. Por outro lado a própria distribuição não linear dos sensores, por vezes em grande escala, exige protocolos de comunicação

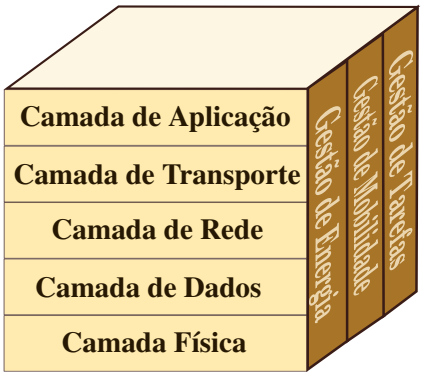

Figura 2.1: Modelo OSI

altamente escaláveis que sejam capazes de dinamicamente eleger sensores para funções de encaminhamento, preenchendo assim "buracos"na conectividade da rede, mas mantendo a estrutura hierárquica da rede. E finalmente a preocupação com a gestão eficiente da energia consumida, dadas as limitações de armazenamento e geração de energia neste tipo de dispositivos, que é uma preocupação de todas as RSSF e um dos principais objetivos a aprofundar neste trabalho .

2.1.1.1 Camada de Ligação de Dados (MAC)

A camada de ligação de dados trata da transferência de dados entre dois nós que partilhem a mesma ligação (*link*). Como esta ligação é feita sobre uma rede sem fios, logo partilhada, é necessário um mecanismo de gestão e controlo de acesso ao meio (MAC). Nas RSSF esta camada deve cumprir requisitos de eficiência energética, sincronização dos *frames*, gestão eficiente da largura de banda e controlo de fluxo e de erros. A redução do consumo de energia aumentando o tempo de vida da rede é o seu principal objetivo. O protocolo implementado nesta camada deve evitar o desperdício de energia devido a colisões de pacotes, *overhearing*, retransmissões excessivas, *overheads* de controlo e *idle listening*. Um conjunto alargado de protocolos MAC tem sido proposto para atender a estas questões [16]:

- TRAMA (*Traffic-Adaptive Medium Access*): Este protocolo atinge a eficiência energética evitando colisões e adaptando o escalonamento de transmissão de acordo com os padrões de tráfico. Usa apenas um canal escalonado temporalmente para dados e controlo da transmissão e divide as secções de escalonamento em dois tipos: acesso aleatório e acesso escalonado. Para fazer esta gestão sub-divide-se em três componentes fundamentais: *Neighbor Protocol*(NP), *Schedule Exchange Protocol* (SEP), e *Adaptive Election Algorithm* (AEA). No período de comunicação aleatória, durante o qual os nós podem juntar-se à rede, o NP envia pequenos pacotes de sinalização e usa-os para manter a conectividade com a vizinhança. O SEP activa o tráfego escalonado, durante este período emite a informação de escalonamento à vizinhança no raio de um salto (*hop*), a partir do qual os nós se sincronizam. O ultimo componente, AEA, determina o estado de cada nó. Por razões de eficiência

Tabela 2.1: Comparação de protocolos da Camada de Ligação de Dados para RSSF.

	TRAMA	B-MAC	Z-MAC	LPR-MAC	LPD-MAC	CC-MAC
Channel access mode	Time-slotted random and scheduled access	Clear channel assessment (CCA)	Time-slotted random and scheduled access	Time-slotted contention based slot reservation	Multi-channel access	Time-slotted contention based slot reservation
Time synchronization	Yes	No	Yes	Yes	No	No
Protocol type	TDMA CSMA	CSMA	TDMA CSMA	TDMA	CSMA/CA	CSMA/CA
Protocol specifics	Achieves adequate throughput and fairness through transmitterelection algorithm and channel re-use	Bi-directional interface for reconfiguration of system services to optimize performance	Exploits the strengths of TDMA and CSMA while offsetting their weaknesses	Increases the probability of success in packet transmission by adapting to traffic requirements to maximize data throughput	Combines CSMA and spread spectrum techniques to achieve higher power efficiency and bandwidth	Filters out correlated data and ensures prioritization of packets to the sink which results in achieving higher network performance
Energy conservation	Schedule sleep intervals and turn radio off when idle, collision avoidance scheduling	Low power listening (LPL) time for energy efficiency	Low power listening (LPL) time for energy efficiency	Nodes sleep and wake up based on assigned data slot	Power saving mode with low power wake up radio for channel listening and normal radio for data transmission	Dropping highly correlated information packet to reduce energy use in transmission

energética os nós são mantidos no modo sleep na maioria do tempo. O protocolo TRAMA garante eficiência energética e na entrega, à custa de atrasos nos pacotes, assim como atinge elevadas taxas de transferência e evita colisões.

- B-MAC: Ao contrário do TRAMA, é um protocolo *carrier-sense multiple acess* (CSMA) que alcança baixo processamento, alta utilização do canal e evita colisões. O B-MAC otimiza a performance do sistema ao empregar um esquema de amostragem adaptativo. Um conjunto de interfaces bidirecionais são usados para reconfigurar o protocolo em função da carga da rede. Para evitar colisões procura valores atípicos nos sinais de amostra recebidos (*clear channel assessment*, CCA). Os valores atípicos são considerados quando durante o período de amostragem a energia do canal estiver abaixo do nível de ruído, nesse caso o canal está livre, caso contrário é considerado ocupado e inicia-se um período de espera (*packet back off*) inicialmente definido ou aleatório. Todas as funcionalidades do B-MAC tais como os *acknowledgements*, o CCA, e o *packet back off* podem ser alterados através de um conjunto de interfaces bidirecionais. A sua ativação ou desativação altera a taxa de transferência e o consumo de energia do nó.

- Z-MAC: Em comparação com o B-MAC, o Z-MAC é um protocolo MAC híbrido que combina as vantagens do TDMA e CSMA. o Z-MAC alcança alta utilização do canal e baixa latência sob alta contenção. Reduz, com baixo custo, as colisões entre vizinhos num raio de dois saltos (*hops*). É robusto contra alterações dinâmicas da topologia e falhas de sincronização que ocorrem frequentemente na rede. o Z-MAC usa o CSMA como esquema base

e o escalonamento TDMA para melhorar a resolução da contenção. Estas opções resultam num aumento inicial do *overhead* que é eventualmente amortizado e ao longo do tempo acaba por melhorar a taxa de transferência e a eficiência energética. Ao misturar CSMA e TDMA torna-se mais resistente a falhas de pontualidade (*timing*), variações temporais das condições do canal, falhas na atribuição de *slot* e alterações da topologia. Entre média e alta contenção o Z-MAC apresenta melhor performance que o B-MAC, no entanto para baixa contenção o B-MAC tem um comportamento ligeiramente melhor em termos energéticos.

- LPR-MAC (*Low power reservation-based MAC*): o protocolo *Low power reservation-based MAC* atende o problema da conservação de energia e adaptação do tráfego. Para o fazer usa uma rede *cluster* hierarquizada e uma estrutura de frame do tipo TDMA. Numa rede *cluster* os nós organizam-se autonomamente em *clusters* e atribuem a tarefa de *cluster head* a um nó em cada *cluster*, estes nós ficam então responsáveis pela sincronização de todos os outros nós nesse *cluster*, segundo um escalonamento TDMA. De acordo com o protocolo o *cluster head* aumenta o tamanho do *frame* se o número de falhas exceder um valor predeterminado. Caso contrário, se o número de falhas foi baixo, o tamanho do *frame* diminuirá e dessa forma aumentará a probabilidade de sucesso da transmissão de pacotes.

- LPD-MAC (*Low power distributed MAC*): O protocolo *Low power distributed MAC* combina CSMA/CA e técnicas multi-canal de espectro alargado. Para uma dada frequência, a banda é repartida em múltiplos canais, que são atribuíveis na rede a cada nó juntamente com um código. Tanto o código como o canal tem de ser únicos para cada canal num raio de dois saltos (*hops*). O principal objetivo desta estratégia é evitar colisões e reduzir o desperdício de energia. O protocolo também incorpora dois interfaces RF, um de baixa potência destinado a acordar o nó quando há dados para enviar ou receber e um de potência normal para as comunicações normais. Como a energia consumida durante a monitorização do meio é muito baixa, o valor médio da energia consumida é reduzido significativamente mantendo o interface RF normal no modo sleep sempre que possível.

- *Spatial correlation-based collaborative MAC (CC-MAC)*: Este protocolo explora a correlação espacial dos dados presentes na camada MAC para prevenir e regular transmissões redundantes. O CC-MAC tem dois componentes: o *event MAC* (E-MAC) e o *network MAC* (N-MAC). Enquanto o E-MAC filtra pacotes de dados redundantes, o N-MAC prioriza os pacotes a encaminhar. O CC-MAC é implementado em cada nó. Numa RSSF, o protocolo E-MAC forma regiões de correlação para filtrar informação de eventos correlacionados. Em cada região um único nó representativo é selecionado para transmitir os seus dados enquanto os restantes nós esperam. No fim de cada período todos os restantes nós dessa região entram numa fase de contenção para serem eleitos como o novo nó representativo. Enquanto esta filtragem é feita o protocolo N-MAC encaminha os pacotes de dados para o *sink* recorrendo a um método baseado em prioridades. Em termos de performance o CC-MAC apresenta poupanças de energia significativas, baixas latências e baixas perdas de pacotes.

2.1.1.2 Camada de Rede

A camada de rede é responsável pelo encaminhamento dos dados ao longo rede, desde a origem destes até ao seu destino. Os protocolos de encaminhamento usados em RSSF diferem dos tradicionais protocolos de encaminhamento devido às características particulares que estas redes apresentam, por exemplo os nós não têm endereços IP, os protocolos de encaminhamento baseados em IP não podem ser usados. Outra característica particular é a necessidade de todos, ou de uma parte, dos nós que compõem a rede serem capazes de reencaminhar pacotes *(multi-hop)*, ao invés de estes serem transmitidos diretamente à estação base (*single-hop*). Esta característica é maioritariamente implementada na camada de rede, aumenta o alcance das redes e pode contribuir para a redução do consumo de energia ao permitir que os nós comuniquem com potências mais baixas. No entanto o *overhead* introduzido por alguns destes protocolos pode comprometer os ganhos energéticos que se pensava adquirir [17].

- O Encaminhamento geográfico (*Geographical routing*) usa um mecanismo de reencaminhamento ganancioso ao escolher os nós que estão mais perto do destino para encaminhar os pacotes. Assume para isso que a rede é suficientemente densa, que os nós sabem a sua localização e que o encaminhamento por vários saltos (*multi-hop*) é confiável. Existem um conjunto de propostas para este tipo de encaminhamento que se podem dividir entre as baseadas na distância (*distance-based*) e as baseadas na receção (*reception-based*). Na *distance-based* um nó apenas conhece a distância que o separa dos nós vizinhos, enquanto que na *reception based* as taxas de receção de pacotes dos vizinhos também são conhecidas. Estudos mostram que as estratégias de encaminhamento *reception-based* são mais eficientes que as *distance-based*, por isso faz-se agora o levantamento de quatro soluções *reception based*: 1) *Absolute reception-based blacklisting*: Cada nó excluí os nós vizinhos que tenham uma taxa de receção inferior a um dado valor. Apenas os vizinhos mais próximos do destino com boas taxas de receção receberá o pacote para encaminhar; 2) *Relative reception-based blacklisting*: Cada nó excluí um conjunto diferente de nós vizinhos para cada destino dependendo do ranking de cada nó nesse conjunto. O nó de cada *ranking* depende da sua distância ao destino e da sua taxa de receção. Desta forma impede-se que todo os nós vizinhos sejam excluídos como acontece na *absolute reception-based blacklisting*; 3) *Best reception neighbor*: Os pacotes são encaminhados para os nós com a taxa de receção mais alta e que estão mais perto do destino; 4) *Best reception rate and distance*: Esta estratégia baseia-se no produto da taxa de receção e da distância. Um nó calcula este produto para todos os nós vizinhos que estão mais perto do seu destino e escolhe aquele para o qual este produto for maior.

- O *Anchor Location Service* (ALS) é uma estratégia baseada em malha (grid) que suporta encaminhamento entre fontes e destinos em movimento. Primeiro o ALS constrói uma estrutura em malha predefinida para a rede. Antes da instalação (*deployment*) cada nó tem informação sobre o tamanho de cada célula da malha e as suas coordenadas. A instalação de cada sensor é aleatória e estes podem obter a sua posição a partir de mecanismos de

posicionamento existentes, por exemplo o GPS. Ao saber a sua posição cada nó determina a posição da malha em que se encontra e decide se quer ou não ser um nó da malha. Os nós da malha (grid nodes) estabelecem as ligações para os nós de malha vizinhos.

- *Secure routing* (SecRout). Este protocolo garante a entrega segura dos pacotes entre a fonte e a estação base (*sink*), ao empregar um rede *cluster* com dois níveis. O nível inferior contém os nós ou membros do *cluster*, enquanto que o nível superior contém os *cluster heads*. Cada *cluster* contém um *cluster head* e é por estes que os nós fazem o encaminhamento dos pacotes. Para uma entrega segura dos pacotes, o *SecRout* usa criptografia simétrica ao longo do percurso. Cada nó possui um identificador único (ID) com uma chave pré-distribuída (KEY) usada para encriptar os pacotes. O *SecRout* garante que os pacotes chegam à estação base mesmo que existam nós maliciosos no seu percurso. Os pacotes de encaminhamento e os de dados contém apenas parte da informação do percurso. Cada nó mantém uma tabela de encaminhamento contendo uma parte do percurso (o nó anterior e o próximo). Desta forma quando um nó é comprometido não conseguirá obter informação dos nós intermediários. O SecRout fornece manutenção do percurso e atualização da tabela de encaminhamento e dispara a construção de percursos alternativos quando deteta nós maliciosos.

- *Secure Cell Relay* (SCR) é um protocolo projetado para proporcionar resistência a ataques contra a segurança da rede. É baseado num algoritmo *cluster* onde os nós formam um *cluster* em função da sua localização. SCR difere de outros algoritmos baseados em *cluster* pois não há eleição de *cluster head*. Um nó ativo torna-se o nó de retransmissão dependendo da energia que tem disponível. No SCR a totalidade da rede é dividida em células de igual tamanho e cada sensor sabe a sua própria localização e a das estações base. Assume-se que estas estações são seguras mas os nós podem ser comprometidos.

2.1.1.3 Camada de Transporte

A camada de transporte assegura a solidez e qualidade dos dados entre a fonte e a estação base. Os protocolos que implementam a camada de transporte em RSSF devem suportar múltiplas aplicações, solidez variável, recuperação de perda de pacotes e mecanismos de controlo de congestionamento. Assim como devem ser genéricos e independentes da aplicação. Cada aplicação RSSF tolera níveis diferentes de perda de pacotes, que podem ser resultado de má comunicação RF, congestionamento, colisão de pacotes e/ou falha de nós. Qualquer perda de pacotes pode resultar no desperdício de energia e degradação da qualidade do serviço (QoS) na entrega dos dados, portanto a sua deteção pode melhorar as taxas de transferência e consumos de energia. Há duas abordagens para a recuperação de pacotes, salto-por-salto (*hop-by-hop*) ou ponto-a-ponto (*end-to-end*). A primeira abordagem de retransmissão requer que um nó intermédio guarde a informação do pacote na sua memória. Este método é mais eficiente energicamente visto que a distância de retransmissão é mais curta. Para a retransmissão ponto a ponto as fontes dos dados guardam toda a informação do pacote e levam a cabo a retransmissão quando há a perda de um pacote. A retransmissão ponto-a-ponto permite robustez variável ao passo que a salto-por-salto tem uma

melhor performance quando os requisitos de fiabilidade são altos. O mecanismo de controlo de congestão monitoriza e deteta a congestão, antes de esta ocorrer, notificando a fonte para que reduza a taxa de transmissão. Também neste mecanismo há duas abordagens para o implementar o salto-por-salto e o ponto-a-ponto. O primeiro exige que todos os nós ao longo do percurso monitorizem o *overflow* do buffer. Por seu turno a ponto-a-ponto baseia-se nos nós finais para detetar o congestionamento da rede, que é sinalizada quando *acknowledgements* expirados ou redundantes são recebidos. Há relações de compromisso entre uma e outra estratégia, dependendo da aplicação uma pode ser melhor que a outra. Os protocolos da camada de transporte de RSSF tentam responder aos problemas agora mencionados.

- *Price-oriented reliable transport protocol* (PORT) O PORT minimiza o consumo de energia, alcança o nível necessário de fiabilidade e disponibiliza um mecanismo de gestão e evasão de colisões. Para o fazer utiliza um mecanismo de controlo da congestão embutido na rede que alivia o tráfico dinamicamente. O PORT difere dos restantes protocolos na sua visão de que a fiabilidade não é uma relação entre a taxa de entrada de pacotes efetiva e a pretendida, mas na assunção de que a estação base obtém informação suficiente sobre cada acontecimento relevante. Sempre que um acontecimento relevante acontece na rede, os nós que estejam mais perto vão conter mais informação e menos erro. O PORT adapta a taxa de envio de pacotes destes nós para aumentar a informação da estação base sobre o acontecimento. Para assegurar esta fiabilidade o PORT usa dois mecanismos, um de gestão dinâmica da taxa de transmissão da fonte dos dados e outro que fornece informação à estação base sobre o custo da comunicação ponto a ponto, desde a fonte até à estação base.

- GARUDA É um protocolo fidedigno de transporte *downstream* de dados para redes sem fios. Responde ao problema da fiabilidade das transferências de dados da estação central para os nós. A fiabilidade pode ser definida em 4 categorias: (1) garantia de envio de todos os campos; (2) garantia de entrega a uma região de sensores; (3) garantia de entrega a um mínimo de sensores para mensurar uma região; (4) garantia de entrega a um subconjunto probabilístico de sensores. O GARUDA é uma infraestrutura de núcleo *loss-recovery* e um processo *NACK-based recovery* de dois níveis. A infraestrutura do núcleo é construível usando o método de entrega do primeiro pacote (*first packet delivery method*), entrega que é garantida através de um impulso *Wait-for-First-Packet* (WFP). Este impulso consiste numa sequência de pequenos pulsos enviados periodicamente pela estação base, que quando recebidos pelos nós dos sensores no raio da estação base os coloca em modo de espera pelo primeiro pacote. A entrega do primeiro pacote servirá depois para determinar o número de saltos (*hop*) entre a estação base e cada nó. Qualquer nó pode se auto eleger como um nó-núcleo, sendo que para isso se não detectou nenhum outro entre os nós vizinhos. Um nó-núcleo recém eleito tem se conectar pelo menos a um nó-núcleo de ordem superior. O GARUDA usa uma estratégia de encaminhamento fora de ordem para superar o problema de sub-utilização na eventualidade da perda de pacotes. Quando um pacote é perdido o encaminhamento de pacotes subsequentes é mantido na mesma. Quando um pacote fora de

ordem é detectado é enviado um pedido a notificar o pacote em falta.

- *Delay sensitive transport* (DST) O protocolo DST aborda o problema do controlo da congestão, fiabilidade e entrega atempada dos pacotes. O DST tem dois componentes: um mecanismo de transporte de eventos confiável e um mecanismo de transporte de eventos em tempo-real. O primeiro mede o atraso limitado do evento contra o atraso pretendido e toma as ações necessárias para assegurar o nível de fiabilidade pretendido na comunicação de eventos à estação base. O mecanismo de transporte de eventos em tempo-real usa o atraso na comunicação do evento à estação base para cumprir os seus objetivos. Este é a medida do atraso do transporte do evento e do atraso do processamento do evento, é o tempo entre o evento ocorrer e a estação base tomar conhecimento da sua existência. Simulações deste protocolo mostram que atinge fiabilidade e deteção de eventos atempada com baixo consumo de energia e baixa latência.

- *Pump slowly, fetch quickly* (PSFQ) O PSFQ é um protocolo de transporte escalonável e robusto. Os seus objetivos são garantir a entrega de segmentos de dados, a minimização do número de transmissões para deteção de dados perdidos e operações de recuperação de dados, operar em ambientes hostis e fornecer um compromisso leve para a entrega dos dados. O PSFQ opera em três modos: bomba (*pump operation*), busca (*fetch operation*) e comunicação dos resultados (*report operation*). O modo de operação bomba baseia-se na utilização de dois temporizadores, Tmin e Tmax. Um nó deve esperar pelo menos Tmin antes de transmitir um pacote. Ao esperar pelo menos Tmin, é dada a oportunidade ao nó de recuperar pacotes perdidos e reduzir desta forma as transmissões redundantes. Tmax é usado como um limite de atraso superior para quando todos os pacotes devem ser recebidos. A operação de busca é chamada quando existe uma falha na sequência de pacotes, esta operação executa um pedido de retransmissão do pacote em falta aos nós vizinhos. Por último a comunicação do resultado fornece informação de estado aos utilizadores. Um relatório de estado é enviado do nó pretendido mais distante. Ao longo do percurso cada nó acrescenta informação sobre o seu estado ao pacote original.

- *Event-to-sink reliable transport* (ESRT) É desenvolvido para deteção fiável de eventos com baixo consumo de energia. ESRT usa um mecanismo de controlo de congestão para reduzir o consumo de energia, mantendo o nível desejado de fiabilidade na estação base. Esta estação calcula periodicamente o fator de fiabilidade e a frequência de amostragem. O fator calculado é depois comparado com o valor desejado e definido pela aplicação, caso a confiabilidade calculada seja maior que a desejada o ESRT reduziria a frequência de amostragem. Em cada intervalo o ESRT envia para todos os nós a nova frequência de amostragem.

- *Congestion detection and avoidance* (CODA) O CODA é um esquema de controlo de congestão energeticamente eficiente que pode rapidamente atenuar a congestão assim que detetada. Tem três componentes: deteção de congestão, *hop-by-hop back pressure* e regulação *multi-source*. O CODA deteta a congestão ao monitorizar a ocupação do *buffer* e medindo

a ocupação do canal.Quando uma congestão é detetada o nó envia para os nós vizinhos uma mensagem de supressão e faz ajustamentos para prevenir um congestionamento no sentido inverso. Testes mostram que o CODA pode melhorar a performance da rede e reduzir o consumo de energia.

Tabela 2.2: Comparação de protocolos da camada de transporte para RSSF.

	STCP	PORT	GARUDA	CODA	DST	PSFQ	ESRT
Congestion control	Yes	Yes	No	Yes	Yes	No	Yes
Congestion detection	Buffer size	Node price and link-loss rates	-	buffer size and channel load	Buffer size and average node delay calculation	–	Buffer size
Congestion mitigation	Traffic redirection or end-to-end rate adjustment	Traffic redirection or end-to-end rate adjustment	–	Drop packets or adjust sending rate at each node	End-to-end rate adjustment	–	End-to-end rate adjustment
Reliability Direction	Sensor to sink	Sensor to sink	Sink to sensor	Sensor to sink	Sensor to sink	Sensor to sink	Sensor to sink
Reliability measure	Packet reliability	Event information reliability	Packet and destination reliability	–	Event reliability	Packet reliability	Event reliability
Reliability	End-to-end	–	Hop-by-hop	–	End-to-end	Hop-by-hop	End-to-end
Packet recovery	Yes	No	Yes	–	No	Yes	No
Cache	Yes	–	Yes	–	–	Yes	–
ACK/NACK	ACK, NACK	–	NACK	ACK	–	NACK	–
Energy conservation	Yes	Yes	Yes	Yes	Yes	–	Yes

2.1.2 Normas de Comunicação Sem Fios

A norma IEEE 802.15.4, conhecida também por *Wireless Personal Area Networks* (WPAN), especifica as camadas física e de controlo de acesso ao meio de redes sem fios com baixas taxas de transmissão, que vão desde as dezenas de kbit/s até alguns Mbits/s. Apesar de limitada neste aspeto, quando por exemplo comparada com a IEEE 802.11, *Wireless Local Area Networks* (WLAN), que permite taxas de dezenas de Mbits/s, a ênfase desta norma reside no baixo custo e baixo consumo, sendo por isso especialmente indicada para RSSF, onde as questões relacionadas com a gestão racionada de energia são particularmente importantes. A implementação das camadas superiores do modelo OSI, desde a camada física ate à camada de aplicação, não são definidas nesta norma, são no entanto desenvolvidas por um conjunto de outras normas de onde se destacam a *ZigBee*, ISA100.11a, *Wireless HART* e WISA. O WISA distingue-se das restantes por ser baseada na norma IEEE 802.15.1 (Bluetooth) oferecendo um compromisso interessante entre consumo de energia e taxa de transmissão, sendo particularmente adequado para aplicações que necessitem de altas taxas de transmissão e com fortes requisitos de tempo real, como por exemplo

automação industrial [18].

Um traço comum às várias normas é o uso da TDMA como mecanismo de controlo de acesso ao meio, no entanto no caso da *Zigbee* o mecanismo principal é o CSMA/CA. Na *Zigbee* assim como na ISA100.11 a e 802.15.4e FA MAC é possível configurar o comprimento do *timeslot*, o que não acontece nas restantes normas. Esta propriedade é uma característica importante para permitir a otimização e suporte de redes em tempo-real. Outra particularidade é a seleção do tipo de *timeslot*, que pode ser dedicado ou partilhado, sendo que todas permitem ambos os tipos à exceção da WISA que permite exclusivamente *timeslots* do tempo dedicado. A escolha entre um tipo e outro é difícil devido à relação de compromisso entre suporte de tempo-real e utilização otimizada do meio.

Tabela 2.3: Características relevantes para aplicações em Tempo-Real.

	WISA	WirelessHART	ISA100.11a	ZigBee	802.15.4e MAC
CSMA/CA				x	
FDD	x				
Superframe	x	x	x	x	x
Fixed timeslot	x	x			
Shared timeslots		x	x	x	x
Dedicated timeslots	x	x	x	x	x
Superframe optimization		x	x		
Message-based priority		x	x		
Contract-based priority			x		

Cada norma suporta a comunicação em tempo real de maneira diferente, observando-se que a WISA e 802.15.4e FA MAC se focam em aplicações com fortes requisitos de tempo-real, ao passo que a *ZigBee* e *WirelessHART* estão mais adaptadas a aplicações com requisitos de tempo mais suaves. A tabela 2.4 compara as ferramentas das várias normas para combater potenciais obstáculos e falhas de sensores. A *WirelessHART* and ISA100.11a oferecem o conjunto de ferramentas mais completo permitindo *blacklisting* e mudança de frequência (*frequency hopping*). Todas as normas usam *acknowledgements* na camada de ligação (data link) e pedidos automáticos de repetição, assegurando transmissões confiáveis e identificação de pacotes perdidos. Na camada de transporte, apenas a *WirelessHART* and ISA100.11a suportam *acknowledgements*.

2.1.3 Fontes e gestão de energia

O aumento da eficiência dos conversores de energia, a possibilidade de construir circuitos integrados de muito baixo consumo, assim como com a grande variedade de sensores de baixo custo e baixa potência tornaram possível as RSSF baseadas na combinação de pequenos dispositivos autónomos. Todos os sub-sistemas que compõe um sensor sem fios têm como requisito primordial um funcionamento energeticamente eficiente, caso contrário a autonomia ficaria comprometida.

Tabela 2.4: Características relevantes para funcionamento Robusto e Fidedigno

	WISA	WirelessHART	ISA100.11a	ZigBee	802.15.4e MAC
Mesh topology		x	x	x	
Channel agility				x	?
Channel hopping	x	x	x		?
Channel blacklisting		x	x		?
DDL acknowledgements	x	x	x	x	x
TL acknowledgements		x	x		
Automatic repeat request	x	x	x	x	?

Em paralelo com o desenvolvimento de topologias e mecanismos *power-aware*, também o conjunto de elementos que gera e processa a energia deu avanços significativos. O desenvolvimento da tecnologia das baterias, de dispositivos de comutação mais eficientes e circuitos reguladores de potência de alto rendimento aumentaram em muito a autonomia das RSSF.

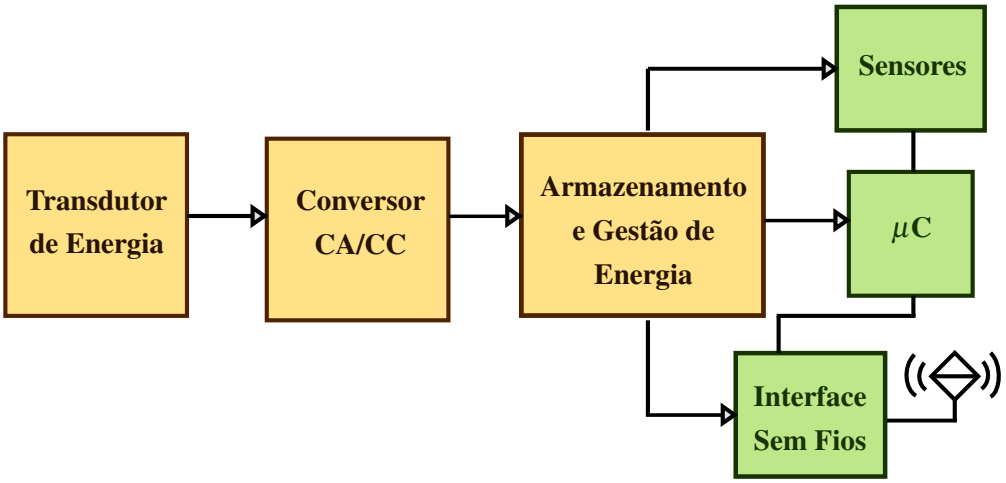

Figura 2.2: Estrutura de um nó autónomo de uma RSSF.

Todos estes desenvolvimentos levaram a uma diminuição considerável do consumo geral dos sensores para valores muito baixos, da ordem dos uW[fonte]. De tal forma que a recolha de energia presente no ambiente dos sensores passou a ser considerada como potencial fonte de energia para RSSF autónomas[19][20]. Nesse sentido há um conjunto de projetos e investigações que propõe o uso de "*energy scavenging*"ou "*power harvesting*"como solução para alimentar as RSSF levando a sua autonomia para valores tendencialmente ilimitados. De entre as várias soluções propostas para a recolha de energia salienta-se a energia solar, vibrações mecânicas, variações de temperatura, energia eólica e ruído eletromagnético.

Capítulo 3

Definição de Protótipo

Para a escolha dos elementos constitutivos da rede a implementar foi feito um levantamento do estado da arte das várias tecnologias necessárias. Tanto as diretamente relacionadas com RSSF e concebidas com esse propósito, como as ferramentas com campos de aplicação mais abrangentes, por exemplo dispositivos móveis e/ou interfaces homem-máquina.

3.1 Hardware

3.1.1 Plataformas de Hardware para Redes de Sensores Sem Fio

Um nó de uma RSSF reúne num único dispositivo, sistemas de medição, recolha de dados e respetivo processamento, comunicação RF assim como uma fonte de alimentação e circuitos reguladores e de gestão de energia. Apesar de haver uma vasta oferta para cada um dos componentes, existe hoje uma oferta considerável de soluções que integram de origem todos os componentes com a perspetiva de aplicação em RSSF. Apesar da adoção de plataformas já construídas limitar de certa forma o projeto do sistema, o facto destas terem sido concebidas com este tipo de aplicações em mente ultrapassa largamente essa desvantagem pois o seu uso traduz-se numa redução do custo, redução do trabalho de projeto e implementação, utilização de uma plataforma amplamente testada noutros projetos mais ou menos relacionados e finalmente a garantia de integração a nível de *hardware* de ferramentas importantes para as RSSF, como por exemplo a poupança de energia. Na Tabela 3.1 comparam-se algumas das plataformas que mais se têm destacado na área das RSSF[21][22].

Tabela 3.1: Comparação de plataformas orientadas a RSSF.

	XBee	M1030 Mote	M2135 Mote	MicaZ	Mica2	Iris
Fabricante	Digi	Dust Networks	Dust Networks	Crossbow	Crossbow	Crossbow
Frequência	2,4 GHz	900 MHz	2,4 GHz	2,4 GHz	900 MHz	2,4 GHz
Largura de Banda [Kbps]	250	76,8	250	250	40	250
Corrente Consumida (em escuta / Rx / Tx) [μA]	-/40/40	-/14/28	-/22/50	8/20/18	8/10/17	8/16/10
Consumo em "sleep" [μA]	1	8	10	27	19	8
CPU @ [MHz]	-	-	-	8bit Atmel @ 8	8bit Atmel @ 8	8bit Atmel
Memoria Flash [kB]	-	-	-	4	4	128

3.1.2 Sensores

Para a monitorização da cadeia de frio em veículos refrigerados as duas grandezas que interessa medir são a temperatura e a humidade.

3.1.2.1 Sensores de temperatura

Para a medição da temperatura em veículos refrigerados os sensores utilizados têm sido essencialmente de 3 tipos: termístores, termo-resistências (RTD) e termo-pares. Os termístores são feitos à base de compósitos cerâmicos ou polímericos com óxidos de metais cuja resistência diminui com o aumento da temperatura. Em comparação as termo-resistências são normalmente feitas de metais puros. Os termo-pares baseiam-se no princípio de que uma junção de dois metais diferentes produz uma diferença de potencial proporcional à temperatura. A Tabela 3.2 mostra as principais diferenças entre os três tipos.

Tabela 3.2: Características dos principais sensores de temperatura.

	Termístor	Termopar	Termoresistência
Gama de temperatura	-100^{o}C 500^{o}C	-267^{o}C 2316^{o}C	240^{o}C to 649^{o}C
Tempo de resposta	Rápido	Moderado	Lento
Numero de fios	2	2	3 ou 4
Sensibilidade	Alta	Baixa	Moderada
Estabilidade	Moderada	Baixa	Alta
Fonte de alimentação	Corrente	Não Precisa	Corrente
Custo	Baixo	Baixo	Alto

3.1.2.2 Sensores de Humidade

À semelhança dos sensores de temperatura, os principais sensores usados para medir a humidade são de três tipos: capacitivos, resistivos(resistência CC ou impedância AC) e piezoresistivo.

Podem ainda ser divididos quando à sua construção. Na Tabela 3.3 podemos ver uma comparação das principais características de vários sensores de humidade.

Tabela 3.3: Características dos principais sensores de humidade.

Tipo de Sensor:	Indutivo	Indutivo	Resistivo	Capacitivo	Capacitivo	Capacitivo	Piezo-resistivo
Tipo de Substrato:	Poli-eletrólito	Polímero	Cerâmico Poroso	Polímero	Polímero	Polímero	Polímero
Quantidade Medida	HR	HR	Ponto de gelo e de orvalho	HR	HR	HR e T	HR e T
Gama de Medida	0 a 100	0 a 100	-80^{o}C a 60^{o}C	0 a 100	0 a 100	0 a 100	20 a 100
Custo	Alto	Alto	Alto	Médio a Alto	Médio a Alto	Muito Baixo	Baixo
Precisão	+/- 5	+/- 5	+/- 3^{o}C	5	1 a 5	+/- 3.5	+/- 1
Gama de Temperatura	-40^{o}C a 100^{o}C	0^{o}C a 60^{o}C	-80^{o}C a 60^{o}C	-40^{o}C a 100^{o}C	-40^{o}C a 185^{o}C	-40^{o}C a 120^{o}C	-15^{o}C a 185^{o}C
Envolucro	Não Standard SIP	Não Standard SIP	Sonda	Não Standard	Não Standard SIP	SMD/-MEMS	SMD/-MEMS

3.1.3 Interfaces de utilizador

O uso de dispositivos móveis na gestão de cadeias de transporte (*Supply Chain Management*) é uma tendência que tem vindo a aumentar nas ultimas décadas [23], diretamente relacionada com o desenvolvimento e multiplicação destes dispositivos. Os dispositivos móveis atuais, vulgarmente chamados de *smartphones* são o resultado da fusão do mercados dos *Personal Data Assistants* (PDA) e dos telemóveis.

No início dos anos 90 os primeiros PDAs (p.e. Pilot 1000 e Pilot 5000) forneciam aplicações simples como base-de-dados de contactos, calendário e bloco de notas enquanto que os primeiros telemóveis tinham como única funcionalidade realizar chamadas. As aplicações foram um elemento chave diferenciador e impulsionador do mercado dos telemóveis, sendo cada vez mais complexas e exigindo cada vez mais recursos. No final da década os telemóveis tinham absorvido as funcionalidades dos PDA e exigiam sistemas operativos que permitissem tirar melhor partido dos recursos agora disponíveis, tais como o Pocket PC 2000 da Microsoft, o BlackBerry OS ou o Symbian. Ao contrário dos outros, o Symbian nasceu da colaboração dos principais fabricantes de telemóveis da altura (Nokia, Ericsson, Panasonic e Samsung) uma das razões que o levou a atingir uma posição dominante.

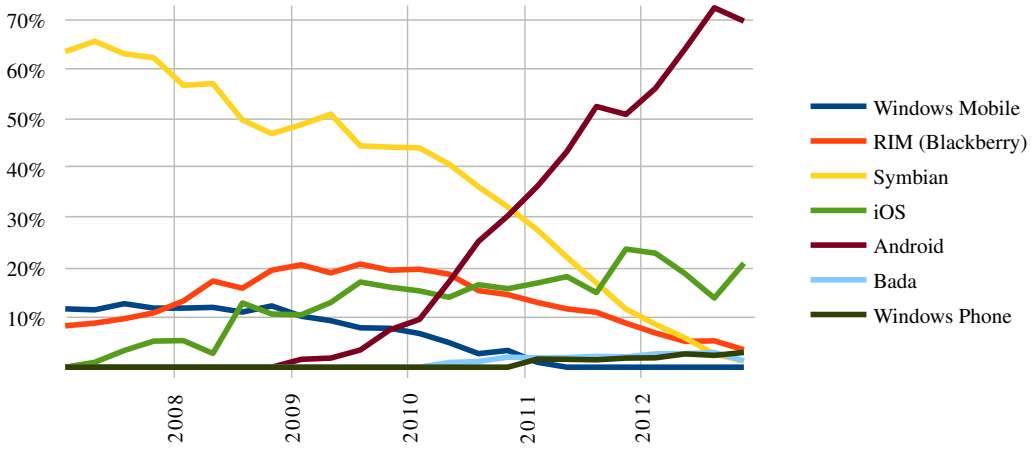

Figura 3.1: Evolução da Quota de Mercado dos Sistemas Operativos em *Smartphones*

Mais uma vez a necessidade de novas e inovadoras aplicações impulsionou o mercado e foram o iOS da Apple e o Android da Google os SO que tiraram melhor partido disso. O SO da Google tem a particularidade de ter origem num consórcio de fabricantes (*The Open Handset Alliance*). E é simultaneamente baseado em Linux e num modelo *open source*, sendo assim uma plataforma mais fácil e apetecível para o desenvolvimento de aplicações quando comparado por exemplo com o iOS da Apple, que assenta num modelo fechado e proprietário[24]. Estas são algumas das razões que permitiram que seja hoje o sistema operativo dominante nos dispositivos móveis como se pode constatar na Figura 3.1.

3.2 Software

3.2.1 Sistemas Operativos orientados a RSSF

As redes de sensores sem-fios são constituídas pela implantação em grupo de muitos pequenos nós, vulgarmente chamados de motes ou *sensor nodes*. Tipicamente estes motes tem um micro-controlador de alguns bits (p.e. 8 bits), memória flash (memoria de programa) na ordem das centenas de KB e memória RAM na ordem de algumas dezenas de KB. Apesar da simplicidade do hardware dos motes as aplicações de redes de sensores sem fios são vastas e exigentes. A ponte entre os motes com recursos limitados e RSSF vastas e complexas é feita pelos sistemas operativos orientados a este tipo de redes, cujas principais funcionalidades são a abstração dos vários dispositivos físicos, a gestão de interrupções, o escalonamento de tarefas, o suporte e gestão da rede e a gestão de concorrência, na execução de tarefas e de acesso aos recursos. Simultaneamente com estas funcionalidades e resultado da camada de abstração criada, os SO das RSSF facilitam o desenvolvimento e implantação destas redes a quem as programa, ao mesmo tempo que garante uma gestão otimizada dos recursos disponíveis.

Durante o levantamento deste estado da arte verificou-se que os SOs existentes à data dão resposta, no todo ou em parte, às principais preocupações com o seu projeto e desenvolvimento,

a saber: baixo impacto (*footprint*) nos recursos; eficiência energética; fiabilidade; garantias de tempo-real; reconfigurabilidade; facilidades de programação. No entanto, apesar do considerável desenvolvimento feito nos últimos anos a implementação de escalonamentos com preempção por prioridade, a segurança dos SO ou a existência de memória virtual são necessidades que ainda não foram atingidas com robustez e que necessitam de maior desenvolvimento para poder sem usadas com facilidade nas RSSF[2]. Uma exceção relativamente ao escalonamento com preempção é o *Nano-RK* que o implementa com prioridades, o que lhe permite ser um dos poucos SO de RSSF que cumpra requisitos de tempo-real.

Tabela 3.4: Comparação de sistemas operativos orientados a RSSF.[1][2]

	TinyOS	**Contiki**	**Mantis**	**Nano-RK**	**LiteOS**
Publicação (Ano)	ASPLOS (2000)	EmNets (2004)	MONET (2005)	RTSS (2005)	IPSN (2008)
Modelo de Programação	Eventos & Threads (TinyThread, TOSThreads)	Eventos & Threads & Protothreads	Eventos & Threads (TinyMOS)	Threads	Eventos & Threads (através de *callback*)
Arquitetura	Monolítico	Modular	Modular	Monolítico	Modular
Alocação e Protecção de Memória	Static Memory Management with memory protection	Dynamic memory management and linking. No process address space protection.	Gestão dinâmica da memoria é suportada mas desencorajada, no memory protection.	Static Memory Management and No memory protection	Dynamic memory management and it provides memory protection to processes.
Partilha de Recursos	Virtualization and Completion Events	Serialized Access	Through Semaphores	Serialized access through mutexes and semaphores	Through synchronization primitives
Protocolo de Comunicação	*Active Message*	uIP, uIPv6, Rime	"comm"	Socket	*File-Assisted*
Tempo-real	Não	Não	Não	Sim	Não
Linguagem de Programação	nesC	C	C	C	LiteC++
Reprogramação	Sim (Deluge, FlexCup)	Sim	Não(a partir da versão 1.1)	Não	Sim
Debugging Remoto	Sim (Clairvoyant)	Não	Sim (NodeMD)	Não	Sim (DT)
Escalonamento	FIFO	Events are fired as they occur. Interruts execute w.r.t. priority	Five priority classes and further priorities in each	Rate Monotonic and rate harmonized scheduling	Priority based Round Robin Scheduling

Das dezenas de SO que existem atualmente destacam-se, pela quantidade de implementações testadas, académicas ou "reais", os que se encontram na Tabela 3.4. Variam entre si na forma como deram resposta às principais preocupações levantadas no quadro da arquitetura do sistema e suas especificidades. Os SO em que as aplicações são compiladas juntamente com o SO são denominados monolíticos e permitem a análise total do programa e sua otimização, durante a compilação, mas dificultam a modificação da rede.

Em aplicações onde se pretende mudar a rede com maior frequência os SO modulares são vantajosos, pois dividem o programa em vários módulos que são por sua vez carregados por um *kernel* mais pequeno. Relacionado com esta divisão está também a forma como os recursos são alocados, p.e. memória. Sistemas Operativos monolíticos, como por exemplo o *TinyOS*, são muitas das vezes também estáticos, ou seja, a alocação dos recursos é feita uma única vez durante a elaboração do programa. Por outro lado, em aplicações com requisitos mais dinâmicos, essa alocação e desalocação é feita durante a execução do programa. Apesar de mais flexível, a alocação dinâmica de recursos, é menos fiável e pode resultar em que aplicações mal projetadas absorvam todos os recursos levando o sistema a um bloqueio. Outro elemento diferenciador está no modelo de programação adotado.

Sistemas organizados em *threads*, apesar de mais familiares para a maioria dos programadores consomem mais recursos e são menos sensíveis temporalmente. Por seu turno os sistemas orientados a eventos representam uma boa solução para aplicações com recursos muito escassos. A maioria dos SO de RSSF, sendo baseados numa ou outra topologia, possuem bibliotecas ou projetos que permitem abarcar ambas as soluções simultaneamente. O *Contiki* é disso um exemplo, sendo um SO modular cujo *kernel* é orientado a eventos, suporta a execução de *threads* com preempção através de uma biblioteca.

Relativamente ao suporte da rede, para os SO generalistas a implementação de TCP/IP é a norma. No entanto nas RSSF não existe uma arquitetura de rede normalizada, fruto da natureza e diversidade dos padrões de comunicação das RSSF(um-para-um, um-para-todos, todos-para-um, etc.). Sem uma norma estabelecida cada SO aborda este requisito de forma diferente, ainda assim a maioria implementa algum tipo de comunicação single-hop, que pode depois servir de base a outras topologias como a multi-hop. O TinyOS, SOS e Mantis são disso exemplo. Uma característica não técnica importante nesta caracterização dos SO é o número de plataformas de hardware suportadas por cada um. Nesse aspeto o TinyOS[25] é de longe o SO com maior compatibilidade de hardware.[1], como se constata na Tabela 3.5

Tabela 3.5: Plataformas suportadas por cada Sistema Operativo.[1][3]

Operating System	Supported Platforms
TinyOS	Mica, Mica2, MicaZ, TelosB/TMote Sky, BTnode, EyesIF X v1, EyesIF X v2, IMote, IMote 1.0, IMote 2.0, KMote, eyes, tinynode, IRIS, Shimmer, Sky, Rene, SenseNode, TelosB, T-Mote
SOS	Mica2, MicaZ, TelosB/Tmote Sky, KMote, avrora, Protosb, Cricket, Cyclops, emu
Contiki	T-Mote Sky, TelosB, avr MCU,ESB, MSP430 MCU, x86, 6502
Mantis	Mica2, MicaZ, Telos, Mantis nymph
Nano-RK	MicaZ, FireFly
RETOS	MicaZ, TelosB/TMote Sky, TI CC2430
LiteOS	MicaZ, IRIS
Microsoft .NET Micro	IMote 2.0

3.3 Sistema Proposto

Com vista ao cumprimento dos requisitos do sistema enunciados na análise e descrição do sistema a implementar e a partir do estado da arte levantado definiu-se o protótipo a implementar recorrendo essencialmente a dois suportes físicos. A escolha da plataforma física para a construção da rede de sensores esteve limitada às disponíveis no Laboratório de Sistemas Embarcados e Distribuídos de Tempo-Real. Das existentes foram escolhidos os motes Memsic IRIS, dado a sua baixa potência de funcionamento, serem amplamente usados em RSSF e de baixo custo. Acoplado com os motes IRIS, foi usado o modulo de sensores MTS310CB [26] que fornece entre outros sensores, um sensor de temperatura (termístor) da *Panasonic* (ERTJ1VR103J) cuja gama de temperatura é de -40 a 125 ^{o}C[27], cumprindo assim os requisitos da norma NP EN 12830. E um sensor de luminosidade baseado numa célula fotossensível de seleneto de cádmio, cuja folha de características não foi encontrada. Para o interface com o utilizador e depois de uma sequência de contactos com empresas do sector da monitorização de frotas, optamos por um *smartphone Android* (Huawei U8160), dada a facilidade de desenvolvimento em *Android* e o facto de ser o SO com maior implantação e crescimento no mercado de dispositivos móveis. Depois de analisados ambos os dispositivos e seus periféricos constatou-se que a forma mais prática e transparente de fazer a ligação entre a rede de sensores e o interface do utilizador seria usar um adaptador Serie-Bluetooth (HC-05), deixando assim a ligação WIFI do *smartphone* liberta para outra aplicação, como por exemplo o escoamento de todos os dados recolhidos no final de um transporte, na chegada ao armazém.

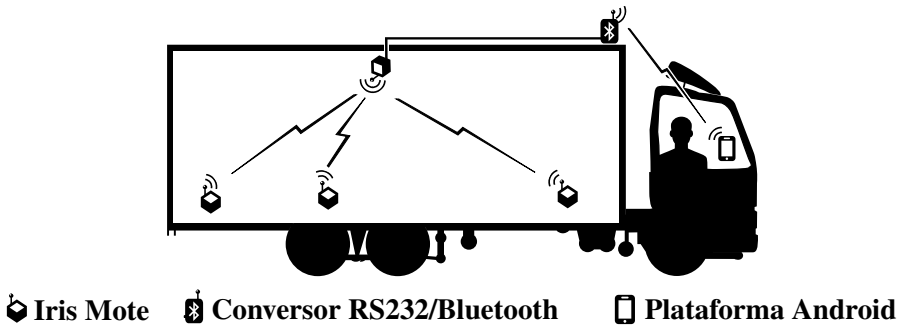

Figura 3.2: Sistema Proposto

3.3.1 Arquitetura

Do ponto de vista da arquitetura de software, o sistema implementado divide-se em três componentes como esquematizado na Figura 3.3. O sistema possui apenas um *gateway*, um interface de utilizador e vários coletores espalhados pelo contentor. As varias instâncias de coletores e o *gateway* configuram a rede de sensores, o foco deste trabalho. A rede de sensores foi programada com base no *TinyOS*, pois alem do mote IRIS ser suportado diretamente por este SO, o *TinyOS* é o sistema operativo com maior implantação nas RSSF como observado na subsecção 3.2.1. Este

sistema operativo entre outras funcionalidades fornece algumas funcionalidades relevantes para a poupança de energia, das quais se destaca o *Low Power Sensing* [28] que faz uma gestão da camada de acesso ao meio implementando o *BoX-MAC*, uma versão melhorada do BMAC [29].

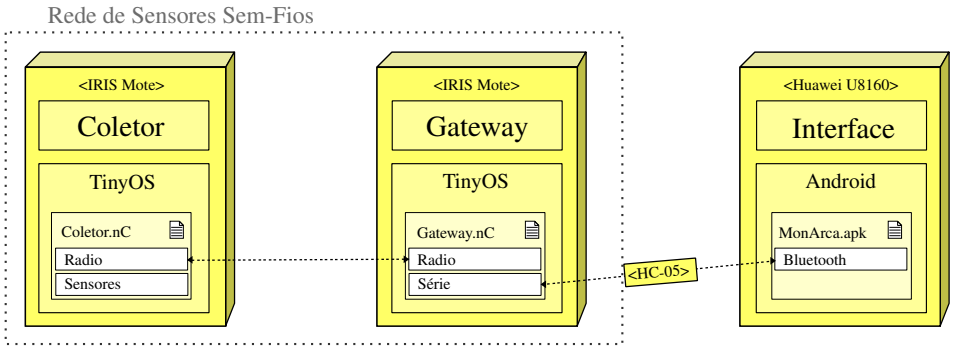

Figura 3.3: Arquitetura do Sistema Proposto

- **Coletor** - Os motes a funcionar como coletores, deverão recolher periodicamente dados (temperatura e luminosidade) e enviá-los para o *gateway*. Por cada pacote enviado o Coletor recebe uma resposta do *gateway* com a indicação de intensidade de sinal (RSSI) com que o pacote chegou ao *gateway*, que é depois usada pelo coletor para adaptar a potência de transmissão ao mínimo possível que permita a comunicação.

- ***Gateway*** - O *gateway* recebe os pacotes com os dados recolhidos de todos os motes e envia-os pela porta série para o interface. Quando um pacote é recebido é imediatamente calculado o seu RSSI e depois introduzido nesse mesmo pacote, sendo esse pacote atualizado o que é enviado de volta para o mote de origem e para o *smartphone*.

- **Interface** - O interface com o utilizador está continuamente ligado ao adaptador série-*bluetooth* e à medida que recebe os pacotes da rede guarda-os numa base de dados e atua alarmes se os valores recebidos ultrapassarem determinados limites pré-definidos.

A sequência da comunicação entre os vários componentes do sistema ocorre como se ilustra na figura 3.4.

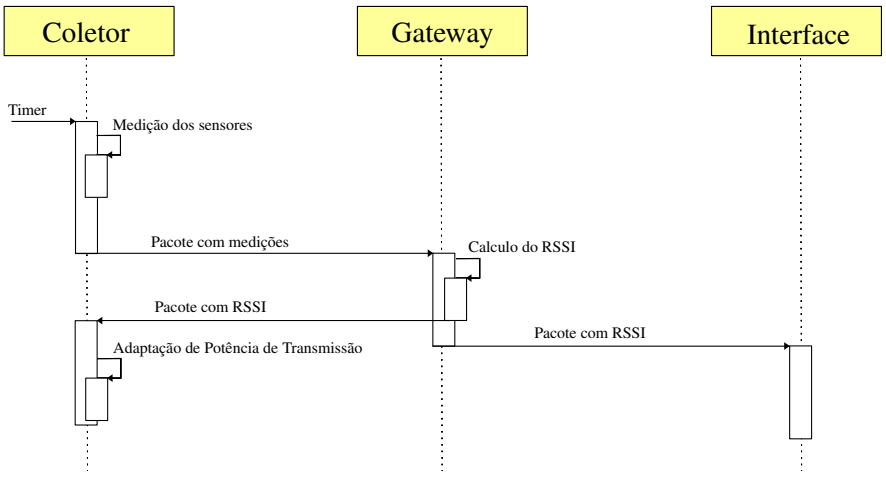

Figura 3.4: Diagrama de Sequência da Comunicação

Capítulo 4

Implementação

O desenvolvimento e implementação de cada componente é neste capítulo descrito, no essencial pela mesma ordem que foi realizado.

4.1 Rede de Sensores

Numa primeira fase da implementação foi feito o estudo e familiarização com o *TinyOS* e a linguagem em que este é programado, *nesC*. Esta linguagem de programação desenvolvida como uma extensão ao C, tem uma execução baseada em eventos, um modelo de concorrência flexível e uma estrutura de aplicação organizada em componentes[30]. No que toca à organização da estrutura, as aplicações em *TinyOS* são construídas a partir da interligação (*wiring*) dos vários componentes necessários. Estes componentes podem tanto ser módulos de software como abstrações de recursos físicos (*hardware*). A abstração de recursos físicos em *TinyOS* faz-se em três níveis: um nível superior que fomenta a portabilidade ao fornecer interfaces independentes da plataforma - HIL (*Hardware Interface Layer*); um nível intermédio que promove a eficiência através de interfaces específicos para cada dispositivo físico - HAL (*Hardware Abstraction Layer*); e um nível inferior que acede diretamente aos recursos dos dispositivos, p.e. registos e interrupções - HPL (*Hardware Presentation Layer*).

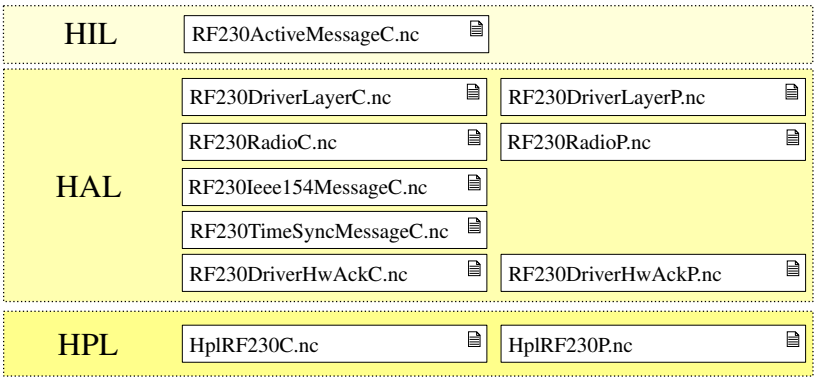

Figura 4.1: Exemplo da Arquitetura de Abstração do módulo AT86RF230 no *TinyOS*

Tanto os motes IRIS como o módulo de sensores MTS310CB são suportados diretamente pelo *TinyOS*, que possui componentes para gerir e aceder a estes recursos. Por exemplo na figura 4.1 podemos ver o conjunto de componentes existentes para utilização com o módulo de comunicação rádio do IRIS (AT86RF230 da Atmel). Cada componente usa ou é usado, dependendo do nível a que pertence, outros componentes, resultando numa rede intrincada de dezenas de componentes. Por isso é vulgar que alguns componentes sejam apenas o encapsulamento das interligações necessárias para construir um dado recurso padrão, libertando o programador de conhecer em todo o detalhe o funcionamento do elementos físicos. Um bom exemplo é o componente RF230ActiveMessageC.nc que, a partir do módulo rádio do IRIS, fornece todos os interfaces necessários para a implementação do ActiveMessageC.nc, o componente que fornece a camada de ligação de dados base do *TinyOS*. Do ponto de vista do programador que vai usar estes recursos é possível aceder a qualquer um dos níveis, no entanto quanto mais "fundo"se faz a ligação, mais complexa se torna a implementação e interligação dos componentes e interfaces necessários.

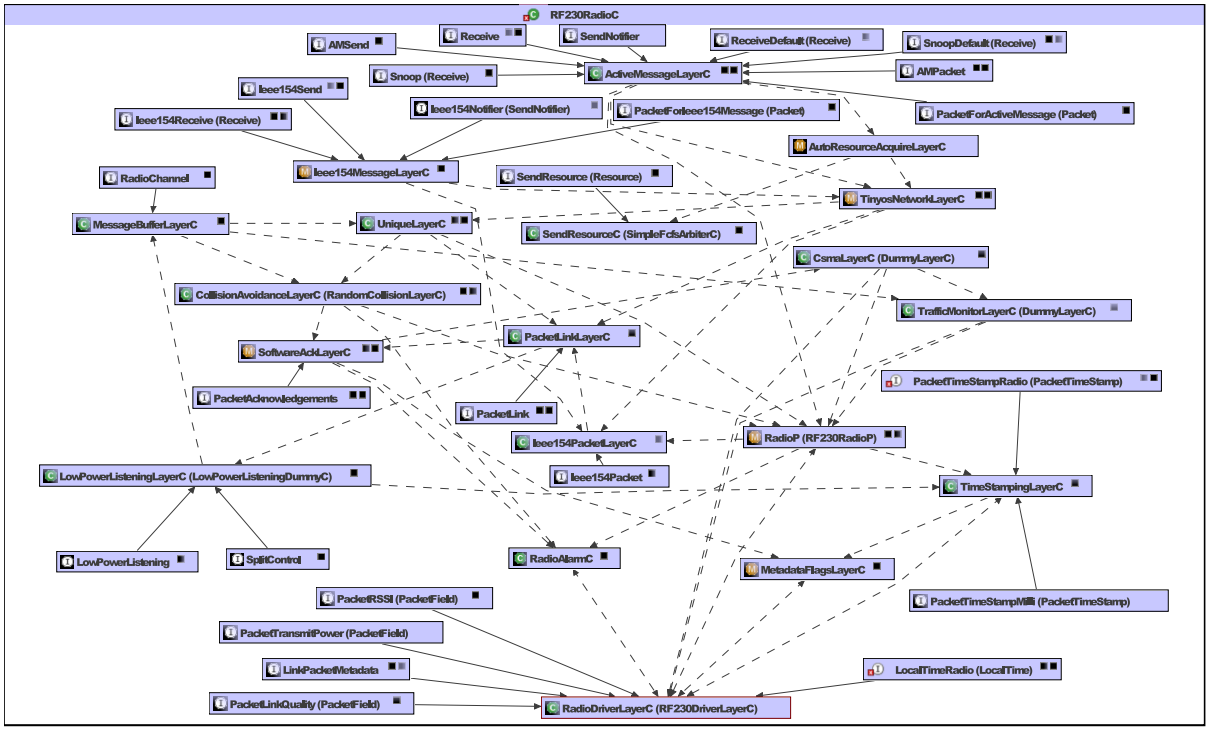

Figura 4.2: Exemplo dos componentes, interfaces e módulos interligados pelo RF230RadioC.nC

Depois da familiarização com o *TinyOS* e o *nesC* deu-se inicio à instalação das ferramentas de software necessárias ao seu desenvolvimento, o ambiente de desenvolvimento do *TinyOS* e respetivo conjunto de ferramentas associadas, num ambiente *linux*. A acrescentar às ferramentas "oficiais"e apesar da maioria da rede desenvolvida ter sido escrita num editor de texto simples (gedit), nalguns momentos o IDE Eclipse, com um *plugin* para nesC/*TinyOS* (Yeti 2 [31, 32]), foi muito útil para a perceção da interligação dos componentes do *TinyOS*, e a partir do qual se pode obter imagens como a 4.2. No entanto porque o desenvolvimento do *plugin* foi interrompido em

2011, existem algumas incompatibilidades com as versões mais recentes do *TinyOS* e do Eclipse, que gradualmente se tornaram incomodativas e portanto limitadoras do uso constante do Eclipse para o desenvolvimento da rede. Finalmente foi configurado e testado o módulo de programação dos IRIS, o MIB600, concretizando assim todo o ambiente de desenvolvimento.

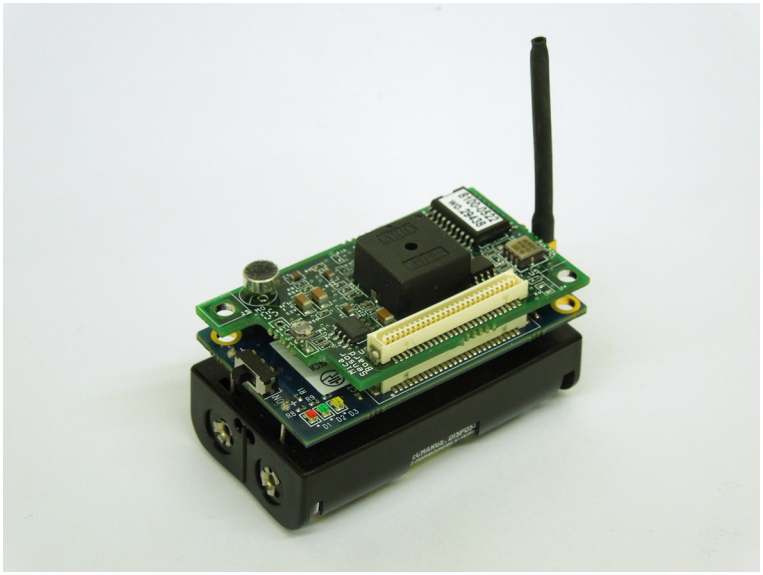

Figura 4.3: O mote IRIS da Memsic, com o módulo de sensores MTS310CB acoplado.

Antes da implementação propriamente dita dos dois elementos constitutivos da rede, o Colector e o *Gateway*, foram testados os vários dispositivos do IRIS em aplicações simples, mas que permitiram de forma isolada perceber o funcionamento de cada um elemento. Para isso recorreu-se aos componentes fornecidos pelo *TinyOS* e aplicações exemplo para os testar. Alem dos componentes mais comuns, como os Leds, comunicação série, ou leitura dos sensores destaca-se o ActiveMessageC. Este componente como já referido anteriormente, implementa e fornece a abstração da camada de ligação de dados. Esta abstração baseada numa comunicação *single-hop* de melhor esforço (*best-effort*) é a base das comunicações em TinyOS, em cima da qual são depois implementados outros protocolos de disseminação. As *Active Messages* possuem um endereço de destino, fornecem *acknowledgments* síncronos, podem ter um tamanho variável até um determinado limite e possuem um identificador de tipo (AMtype), que no fundo serve como identificador de protocolo para componentes construídos em cima da ActiveMessageC. Associado a este componente existem outros como o AMSenderC, o AMReceiverC, o AMSnooperC e o AMSnoopingReceiverC, que fornecem respetivamente serviços para enviar, receber, intercetar e reencaminhar *Active Messages*. A partir desta altura foi possível começar a definir a estrutura das mensagens que circulariam no sistema. Essas mensagens passariam a ser baseadas no ActiveMessageC e o seu conteúdo e estrutura definidos no ficheiro SensorMessage.h (ver 4.4).

Com todos os elementos testados e os componentes necessários para os usar reunidos, entrou-se em consideração com os recursos energéticos da aplicação a implementar. Para isso definiram-se duas estratégias em paralelo, por um lado construir a rede por forma a tirar partido dos vários

```
1  enum {
2   AM_SENSORMESSAGE = 0x74,
3   NREADINGS = 1,
4   DEFAULT_INTERVAL = 500
5  };
6
7  typedef nx_struct sensormessage {
8   nx_uint16_t id;
9   nx_uint16_t rssi;
10  nx_uint16_t tx_power_reading;
11  nx_uint16_t timestamp;
12  nx_uint16_t interval;
13  nx_uint16_t temperature;
14  nx_uint16_t luminosity;
15 } sensormessage_t;
```

Figura 4.4: SensorMessage.h

níveis de potência a que os micro-controladores podem funcionar, no caso do IRIS o ATmega1281, e por outro implementar uma adaptação da potência de transmissão rádio. Relativamente ao níveis de potência o TinyOS faz esta gestão por defeito, a partir do escalonador de tarefas. Sempre que a fila de tarefas a executar é encontrada vazia, o micro-controlador é colocado num nível de potência inferior, calculado com base num *dirty bit* e nos registos de estado e controlo. Para que esta gestão seja o mais eficiente possível a aplicação deve ser estruturada individualizando todas as tarefas a realizar, por forma a que possam ser executadas separadamente e em paralelo, facilitando a gestão de energia realizada pelo *TinyOS*. Para a adaptação de potência é necessário por um lado uma forma de mensurar a intensidade com que cada pacote chega ao *Gateway* e por outro, com base nesse indicador alterar o valor de transmissão. Para isso recorreu-se a dois interfaces, o PacketTransmitPower e o PacketRSSI, ambos fornecidos pelo RF230ActiveMessageC, que por sua vez se interliga aos componentes que implementam as camadas HAL e HPL e fazem a gestão das definições correspondentes fornecidas pelo AT86RF230.

4.1.1 *Gateway*

A partir dos componentes testados anteriormente implementou-se o *Gateway* que, como o nome indica, serve como porta de acesso da rede de sensores para o interface do utilizador. Portanto, como se ilustra na imagem 4.5 este componente do sistema limita-se a encaminhar os pacotes da rede de sensores, para um barramento série a que o interface de utilizador acede de forma transparente e vice-versa. Para ambas as redes, rádio e série, foi usado o mesmo componente de implementação da camada de ligação de dados, a ActiveMessageC. Assim, reduz-se o processamento e latência no encaminhamento de pacotes, pois é o mesmo pacote que circula por ambos os interfaces.

O único processamento realizado pelo *Gateway* é a leitura do indicador de intensidade de sinal do pacote recebido por rádio (RSSI), valor que é depois introduzido na mensagem a reencaminhar.

O *Gateway* alem de necessitar do SensorMessage.h, que define as mensagens que circulam na rede, é composto por dois ficheiros o GatewayC.nc e o GatewayAppC.nc. O primeiro é onde está de facto implementado o componente, enquanto que no segundo é feita a ligação do GatewayC.nc aos componentes do *TinyOS* que fornecem os interfaces que o primeiro utiliza.

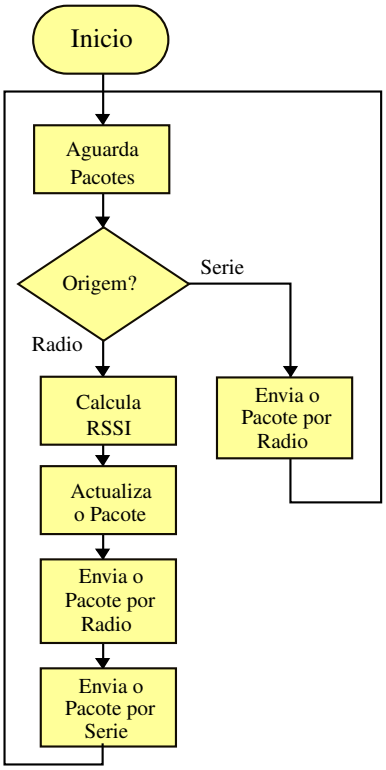

Figura 4.5: Fluxograma do *Gateway*

4.1.2 Coletor

À semelhança do *Gateway*, o coletor usa o ActiveMessageC para implementação da camada de dados, no entanto não usa os mesmos interfaces. Enquanto o *Gateway* necessita de calcular o RSSI, o colector não, precisando antes de alterar dinamicamente a potência de transmissão do módulo de rádio, em função do valor de RSSI que recebe como resposta do *Gateway*, através do interface PacketTransmitPower. Outra diferença está no funcionamento do componente, o Coletor configura um timer do micro-controlador para, periodicamente fazer a leitura dos sensores, construir a mensagem com os dados obtidos e transmitir esse mesmo pacote. Depois de cada pacote transmitido espera pela resposta do *Gateway* com o RSSI. Se o RSSI estiver fora se uma gama pretendida ($RSSI > X_L$ ou $RSSI < X_H$) a potencia de transmissão é adaptada por forma a corrigir esse valor, caso contrario a potência de transmissão não é alterada.

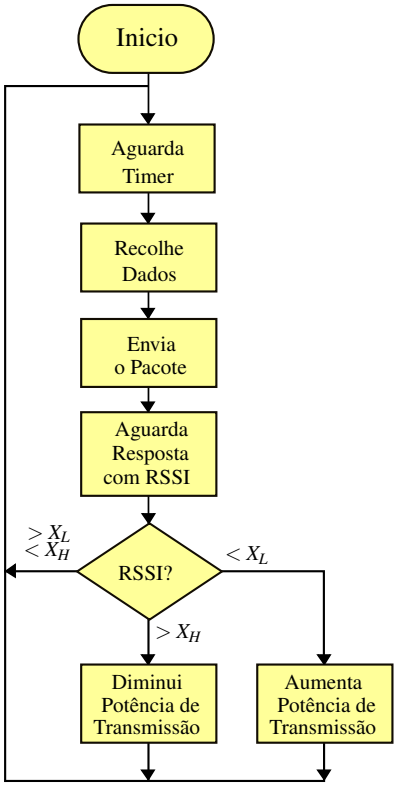

Figura 4.6: Fluxograma do Coletor

4.2 Interface

Para o desenvolvimento e implementação do interface do utilizador a *smartphone Android* foi instalado e usado, também no mesmo ambiente *linux*, o IDE Eclipse, com o *plugin* ADT (*Android Developers Tools*) e o ambiente de desenvolvimento de *Android* (Android SDK). Dada a natureza mais genérica tanto do sistema operativo *Android*, como da linguagem utilizada para o desenvolvimento de aplicações nesta plataforma (Java), o período de estudo e familiarização com o ambiente foi reduzido e breve, em comparação com o despendido durante o desenvolvimento da RSSF. Aquando da criação do projeto *Android* da aplicação, das opções possíveis para a versão da *Application Programming Interface* API, escolhemos a versão 17, com garantia de compatibilidade no máximo até à versão 8 (Android 2.2), devido à compatibilidade na ordem dos 95%.

A arquitetura da aplicação implementada divide-se em três conjuntos de componentes distintos, um principal, onde está incluído o interface gráfico da aplicação e funcionamento principal da aplicação, um componente responsável pela comunicação Bluetooth com o *Gateway* da rede de sensores, através do adaptador Bluetooth-série, e finalmente um conjunto de componentes que implementa uma base de dados onde são armazenados os dados provenientes da rede de sensores.

4.2.1 Comunicação *Bluetooth*

A comunicação com o adaptador *Bluetooth* é implementada numa classe que assegura todo o trabalho de configurar e gerir ligações *bluetooth* com o adaptador Bluetooth-Série 4.7. Depois de instanciado este componente executa uma *thread* que escuta por novas ligações, uma *thread* para manter a ligação com um dispositivo e uma ultima *thread* para realizar as transferência de dados quando conectado. A comunicação através do adaptador é transparente e do ponto de vista da aplicação, este componente recebe e envia *arrays* de bytes. Tirando partido do *Message Interface Generator* do *TinyOS*, gerou-se uma classe em Java para as mensagens trocadas na rede, a partir do qual se faz o *cast* às mensagens recebidas. A mesma classe que define a estrutura da mensagem, possui as funções para ler e escrever campos da mensagem, assim como o construtor do objeto mensagem. Desta forma a personalização da mensagem a trocar entre o interface e a RSSF é bastante simples.

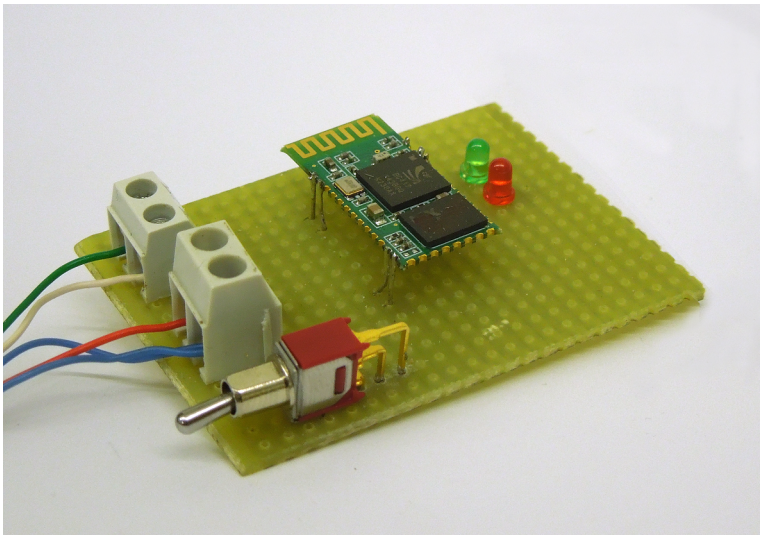

Figura 4.7: O adaptador Bluetooth-Série HC-O5.

4.2.2 Base de Dados

A base de dados foi implementada recorrendo ao SQLite, uma base de dados *open source* incorporada no *Android* que suporta as funcionalidades relacionais padrão das bases de dados. O SQlite suporta apenas 3 tipos de dados: VARCHAR, INTEGER e TIMESTAMP, obrigando a uma conversão de alguns valores, como por exemplo a temperatura, aquando da conversão. A implementação da base de dados fez-se com a criação de uma classe para cada tipo de objeto (Transporte, Mote, Medição, etc). Cada classe implementa a estrutura de cada objeto, a tabela SQL respetiva e as funções e construtores para operar esses mesmos objetos. Finalmente uma outra classe, cria a base de dados e cada uma das tabelas por cada um dos objetos e implementa as funções para inserir e listar os dados guardados.

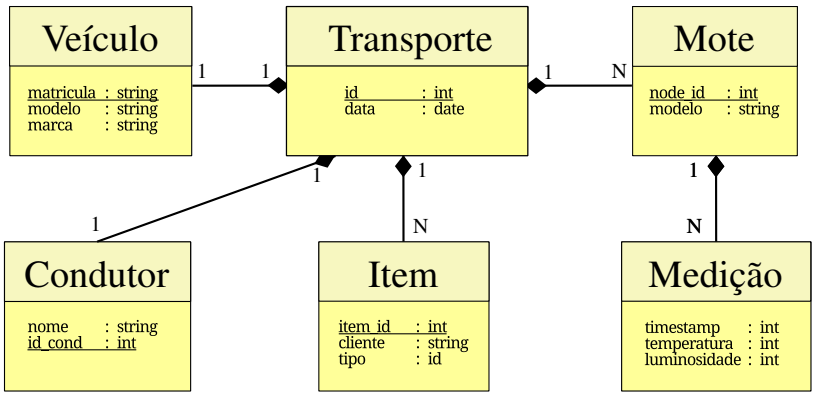

Figura 4.8: Diagrama Relacional da Base de Dados

4.2.3 Interface Gráfico do Utilizador

O interface gráfico com o utilizador é implementado a partir de uma série de *Activities* que se dividem essencialmente em três grupos, *Activities* de configuração, *Activities* de consulta da base de dados e a *Activity* principal onde se encontram as medições mais recentes e são atuados os alarmes.

4.2.3.1 *Activity* principal

Depois de aberta a aplicação a conexão *Bluetooth* com a RSSF é imediatamente estabelecida, expecto se o Bluetooth não estiver ativado, caso em que é pedido ao utilizador que o ative. Depois de estabelecida a conexão, o utilizador pode iniciar a viagem e a respetiva recolha de dados da rede. Os dados mais recentes, a média dos valores da ultima medição de cada sensor, a dispersão desses valores e a média das ultimas 10 medições é mostrada na *Activity*. Se alguma das medições, de qualquer sensor, ultrapassar determinado limite é atuado um alarme de acordo com as configurações dos alarmes.

Figura 4.9: *Activity* principal, à esquerda estado inicial, à direita "em viagem".

4.2.3.2 *Activity* de Configuração

Na *Activity* de configuração definem-se os dados relativos a cada viagem de transporte, o intervalo de amostragem da rede de sensores e os limites para os quais devem ser atuados os alarmes e de que tipo. Todos estes valores são de preenchimento obrigatório e são submetidos assim que se saia da *Activity* de configuração.

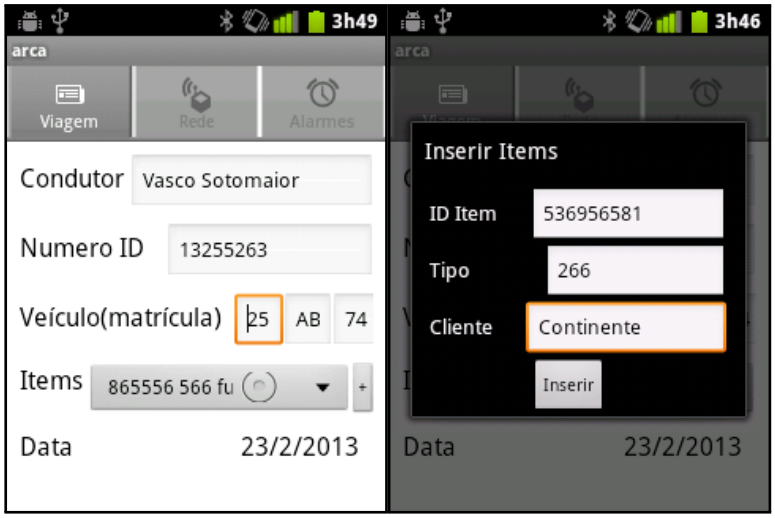

Figura 4.10: *Activity* de configuração da viagem, com pormenor de introdução de campos à esquerda.

Figura 4.11: *Activities* de configuração de rede e alarmes, respetivamente.

4.2.3.3 *Activity* de Consulta da Base de Dados

Finalmente nesta *Activity* podemos ver o histórico dos transportes realizados, os condutores desses transportes e todas as medições realizadas.

Figura 4.12: *Activities* de consulta da base de dados.

Capítulo 5

Resultados Experimentais

Dado o foco do trabalho em questão ter sido a eficiência energética da rede de sensores os testes mais relevantes e realizados primeiro foram ao consumo dos motes. Para isso introduziu-se uma resistência de baixo valor (10ohm) em série com a alimentação. Segundo a datasheet dos motes IRIS a corrente consumida é no máximo da ordem de algumas dezenas de mA, portanto a queda de tensão aos seus terminais seria no máximo da ordem de algumas centenas de mV. Para a leitura do valor das correntes foi usado um *Arduino* com um programa que li-a o valor do ADC de 100 em 100μs e após cada leitura enviava os dados recolhidos pela porta série. Em conjunto com o valor das correntes era enviado também o valor atual micro-segundos de um contador do *Arduino*, iniciado em cada sessão de leitura. Como o ADC do arduino lê tensões de 0 a 5V, amplificou-se a tensão da resistência com um amplificador operacional (OPA350) montado numa montagem não inversora e com ganho igual a 10, permitindo assim a leitura de valores de corrente até 50mA.

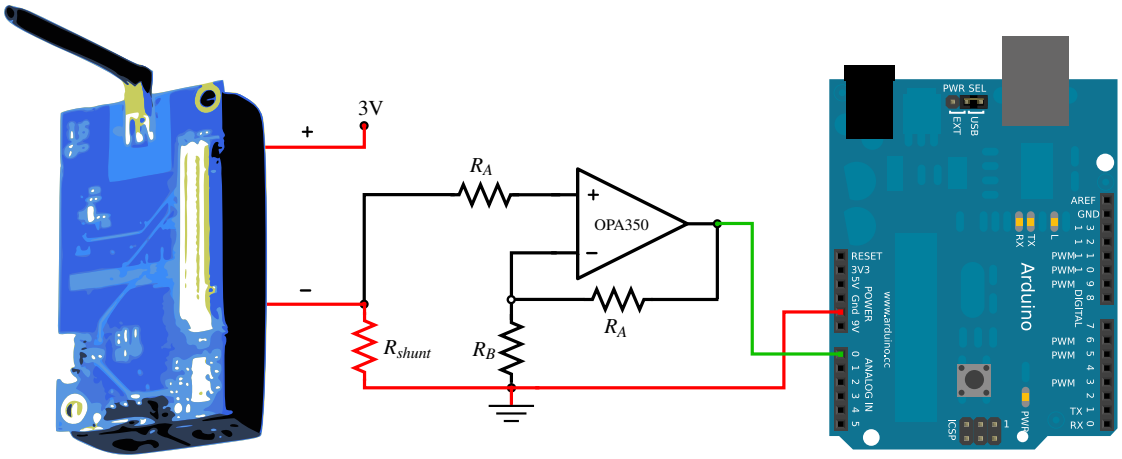

Figura 5.1: Circuito de medição da corrente do *mote* IRIS

Os dados enviados pelo arduino eram depois recebidos por um pequeno programa em python, que li-a a porta serie, converti-a os micro-segundos para milissegundos e a leitura de 10 bits do ADC para miliamperes. Finalmente guardava o resultado num ficheiro CSV (*Comma Separatted*

Values). Simultaneamente com a leitura da porta serie do *arduino*, o programa em *python* lia também a porta serie emulada pelo programador dos motes, MIB600, onde estava montado um mote a funcionar como *Gateway*. Por cada pacote recebido eram extraídos o valor de potência transmitida pelo mote a ser medido (Coletor) e o valor do RSSI com que cada pacote chegava ao *Gateway*, ambos os valores eram guardados no mesmo CSV, no intervalo de tempo correspondente. Os gráficos que se seguem dizem respeito a 2 testes realizados segundo o procedimento agora descrito.

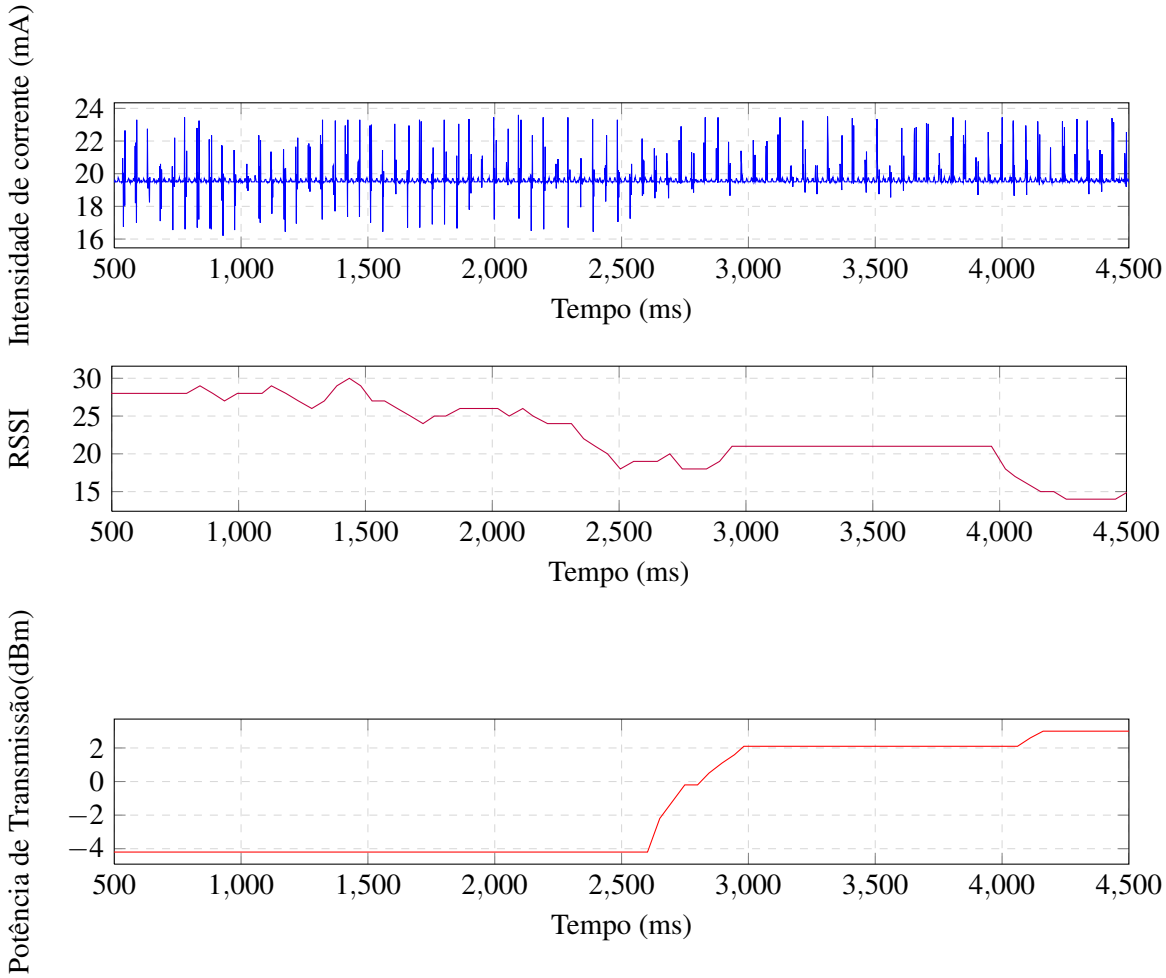

Figura 5.2: Corrente instantânea, RSSI e potência de transmissão de um mote com adaptação de potência

Neste primeiro teste o mote Coletor que estava a ser medido estava com a adaptação de potência ativada e um intervalo de leitura de leitura dos sensores de 50ms. Devido ao facto do intervalo ser muito curto o mote nunca entrava no modo sleep, razão pela qual se observa uma corrente "predominante"de 19 mA. Cada risco vertical corresponde ao envio de um pacote e receção da resposta com o RSSI. Ao obstruir a linha de comunicação entre o Coletor e o *Gateway* foi possível diminui o RSSI e por sua vez obrigar o Coletor a aumentar a potência de transmissão. O impacto dessa alteração no consumo de corrente pode-se observar sobretudo na amplitude das barras verticais

correspondentes a cada comunicação, registando-se uma diferença de aproximadamente 4 mA na amplitude da corrente durante os instantes de comunicação radio. No entanto dado os intervalos de comunicação serem muito curtos, o impacto desta redução é limitada. No gráfico seguinte é calculada a média da corrente num intervalo deslizante e pode-se observar que a adaptação de potência de transmissão em termos médios resulta numa diminuição ou aumento da corrente, em função de um aumento ou diminuição da potência de transmissão, muito reduzida.

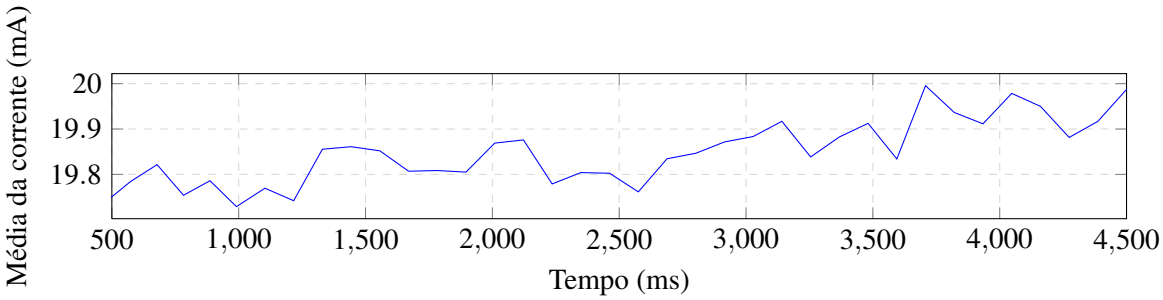

Figura 5.3: Corrente média (intervalo deslizante de 100ms) de um mote com adaptação de potência

No segundo teste configurou-se o mote a ser medido sem a adaptação de potência, com um envio periódico de um pacote a cada 250ms e em cada período a potência de transmissão era diminuída um nível, dos 10 possíveis. Como o período neste caso é superior o mote entra no modo sleep ao fim de 100ms aproximadamente, no entanto é possível observar no inicio de cada período, quando o pacote é enviado, a variação da corrente correspondente à transmissão do pacote radio.

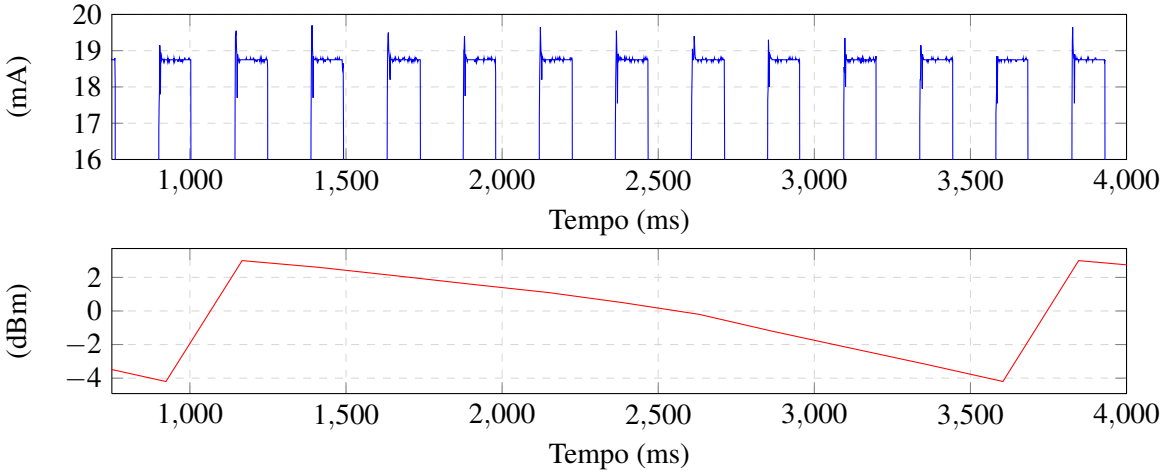

Figura 5.4: Intensidade de corrente e Potência de Transmissão de um mote a percorrer periodicamente todos os níveis de potência possíveis

Como no primeiro teste, também neste é registada uma diminuição da amplitude da corrente correspondente ao modulo radio, na ordem das dezenas de μA. Podemos também observar mais

uma vez que o impacto da corrente consumida pelo modulo radio é muito baixa quando comparada com a corrente que o mote absorve por estar em funcionamento.

Tabela 5.1: Autonomia estimada de um sensor com potência de transmissão fixa para diferentes períodos de leitura (*dutycycle*)

Período (seg)	30	60	300	900
Consumo médio mA	0.0413	0.0247	0.0113	0.0091
Autonomia (Horas)	24201	40552	88246	109761
Autonomia (Semanas)	144	241	525	653
Autonomia (Anos)	3	5	10	13

Para uma corrente média em funcionamento de 20mA e 8μA em *Sleep* e um tempo de leitura e transmissão de 50ms, estimou-se a autonomia de cada nó para diferentes períodos de leitura, como se pode ver na tabela 5.1, não considerando efeitos espúrios como a perda de capacidade nas baterias por idade. Simultaneamente, e para calcular o impacto da adaptação de potência a longo prazo, calculou-se a autonomia para uma corrente média de 19.8mA, ou seja, assumindo um estado ideal em que a potência de transmissão era sempre mínima. Como se pode observar pela comparação das tabelas o ganho de autonomia é muito reduzido, na ordem de 1%.

Tabela 5.2: Autonomia estimada de um sensor com adaptação de potência de transmissão para diferentes períodos de leitura (*dutycycle*)

Periodo (seg)	30	60	300	900
Consumo médio mA	0.0410	0.0245	0.0113	0.0091
Autonomia (Horas)	24398	40827	88506	109895
Autonomia (Semanas)	145	243	527	654
Autonomia (Anos)	3	5	10	13

Capítulo 6

Conclusões

Neste capitulo é feito um apanhado sobre o trabalho desenvolvido durante os últimos meses e de que forma atingiu os objetivos traçados no inicio do semestre. Com base nessa avaliação é depois feito o levantamento das principais linhas de trabalho futuro.

6.1 Avaliação do Trabalho Desenvolvido

O foco do trabalho desenvolvido foi a implementação de uma rede de sensores sem fios num contentor frigorífico. Nesse contexto foi projetada, desenvolvida e testada uma rede de sensores sem fios capaz de monitorizar o ambiente dentro do contentor durante uma viagem. A estrutura e funcionamento da rede foi desenvolvida tendo como principal preocupação o desempenho energético dos sensores e assim permitir que possam funcionar durante largos períodos autonomamente, com renovação de baterias menos regulares, reduzindo assim os custos de manutenção da rede. Para ser atingido este recurso energético tirou-se partido da gestão energética do *TinyOS* e implementou-se uma mecanismo de adaptação da potência de transmissão do modulo de radio, que diminuiu a corrente consumida por cada sensor nas comunicações. Verificou-se no entanto que este ultimo têm um impacto muito reduzido na poupança de energia.

Para o interface com a rede de sensores, assim como o armazenamento e visualização dos dados nela recolhidos, foi desenvolvida uma aplicação para *Android*. Esta aplicação implementou um interface gráfico com a alarmística associada à proteção da carga a ser transportada, um modulo de comunicação bluetooth para comunicar de forma transparente com a rede de sensores sem fios através do *Gateway* e finalmente uma base de dados local onde são armazenados os dados relativos ao ambiente, recolhidos pela rede de sensores sem fios e dados sobre a viagem recolhidos pelo *smartphone*.

6.2 Trabalho Futuro

Devido à variedade de ferramentas utilizadas e de plataformas utilizadas, uma parte considerável do tempo foi despendido com a sua familiarização, diminuindo o tempo disponível para

implementação e testes. Na rede de sensores teria particular interesse a implementação e teste de protocolos de disseminação mais complexos, permitindo maior flexibilidade na implantação da rede, e analisar de que forma é que esses protocolos teriam influência no desempenho energético da rede.

Do ponto de vista da aplicação *Android* e visto não ser o foco do trabalho, seria interessante desenvolver módulos de comunicação de dados (p.e. GPRS) que permitissem a interligação da rede de sensores com a *internet* e monitorização em tempo real do transporte e as condições em que é realizado.

Referências

[1] Wei Dong, Student Member, e Chun Chen. Providing OS Support for Wireless Sensor Networks : Challenges and Approaches. *Communications Surveys & Tutorials, IEEE*, 12(4):519–530, 2010. doi:10.1109/SURV.2010.032610.00045.

[2] Muhammad Omer Farooq e Thomas Kunz. Operating systems for wireless sensor networks: a survey. *Sensors (Basel, Switzerland)*, 11(6):5900–30, Janeiro 2011. URL: `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3231431&tool=pmcentrez&rendertype=abstract`, doi:10.3390/s110605900.

[3] A. K. Dwivedi, M. K. Tiwari, e O. P. Vyas. Operating Systems for Tiny Networked Sensors : A Survey. *International Journal of Recent Trends in Engineering*, 1(2):152–157, 2009.

[4] Luis Ruiz-Garcia, Pilar Barreiro, Jose Ignacio Robla, e Loredana Lunadei. Testing ZigBee Motes for Monitoring Refrigerated Vegetable Transportation under Real Conditions. *Sensors*, 10(5):4968–4982, Maio 2010. URL: `http://www.mdpi.com/1424-8220/10/5/4968/`, doi:10.3390/s100504968.

[5] Carmelita Görg, Walter Lang, Otto Hahn, e Allee Nw. Testing network protocols and signal attenuation in packed food transports Reiner Jedermann *, Markus Becker ,. 9(February 2003):170–181, 2011.

[6] Reiner Jedermann, Javier Palafox-albarran, Amir Jabarri, e Walter Lang. Autonomous Cooperation and Control in Logistics. 2011. URL: `http://link.springer.com/10.1007/978-3-642-19469-6`, doi:10.1007/978-3-642-19469-6.

[7] Ali Dada e Frédéric Thiesse. Sensor Applications in the Supply Chain : The Example of Quality-Based Issuing of Perishables. Em *The Internet of Things*, páginas 140–154. Springer Berlin Heidelberg, 2008. doi:10.1007/978-3-540-78731-0_9.

[8] Walter Lang, Reiner Jedermann, Damian Mrugala, Amir Jabbari, Bernd Krieg-brückner, e Kerstin Schill. The " Intelligent Container "— A Cognitive Sensor Network for Transport Management. *IEEE Sensors Journal*, 11(3):688–698, 2011.

[9] Kássio Machado, Denis Rosário, Eduardo Cerqueira, Antonio a F Loureiro, Augusto Neto, e José Neuman de Souza. A routing protocol based on energy and link quality for internet of things applications. *Sensors (Basel, Switzerland)*, 13(2):1942–64, Janeiro 2013. URL: `http://www.ncbi.nlm.nih.gov/pubmed/23385410`, doi:10.3390/s130201942.

[10] Jin Wang, Yu Niu, Jinsung Cho, e Sungyoung Lee. Analysis of Energy Consumption in Direct Transmission and Multi-hop Transmission for Wireless Sensor Networks. *2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, páginas 275–280, Dezembro 2007. URL: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4618786`, doi:10.1109/SITIS.2007.145.

[11] S. M. Lambor e Sangeeta Joshi. Critical hops calculation for energy conservation in a multi-hop Wireless Sensor Network. *2010 Sixth International conference on Wireless Communication and Sensor Networks*, páginas 1–6, Dezembro 2010. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5712296, doi:10.1109/WCSN.2010.5712296.

[12] J. Moureh e D. Flick. Airflow pattern and temperature distribution in a typical refrigerated truck configuration loaded with pallets. *International Journal of Refrigeration*, 27(5):464–474, Agosto 2004. URL: http://linkinghub.elsevier.com/retrieve/pii/S0140700704000295, doi:10.1016/j.ijrefrig.2004.03.003.

[13] Shivi Singhal, Anil Kumar Gankotiya, Shikha Agarwal, e Tarachand Verma. An Investigation of Wireless Sensor Network: A Distributed Approach in Smart Environment. *2012 Second International Conference on Advanced Computing & Communication Technologies*, páginas 522–529, Janeiro 2012. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6168426, doi:10.1109/ACCT.2012.22.

[14] Qingshan Shan, Ying Liu, Gareth Prossec, e David Brown. Wireless Intelligent Sensor Networks for Refrigerated Vehicle. *IEEE 6th CAS Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, páginas 525–528, 2004.

[15] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, e E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, Março 2002. URL: http://linkinghub.elsevier.com/retrieve/pii/S1389128601003024, doi:10.1016/S1389-1286(01)00302-4.

[16] Jennifer Yick, Biswanath Mukherjee, e Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, Agosto 2008. URL: http://linkinghub.elsevier.com/retrieve/pii/S1389128608001254, doi:10.1016/j.comnet.2008.04.002.

[17] Rex Min e Anantha Chandrakasan. Top Five Myths about the Energy Consumption of Wireless Communication. *Mobile Computing and Communications Review*, 7(I):65–67, 2003.

[18] Delphine Christin, Parag S. Mogre, e Matthias Hollick. Survey on Wireless Sensor Network Technologies for Industrial Automation: The Security and Quality of Service Perspectives. *Future Internet*, 2(2):96–125, Abril 2010. URL: http://www.mdpi.com/1999-5903/2/2/96/, doi:10.3390/fi2020096.

[19] Enrico Dallago, Alberto Danioni, Marco Marchesi, e Giuseppe Venchi. An Autonomous Power Supply System Supporting Low-Power Wireless Sensors. *IEEE Transactions on Power Electronics*, 27(10):4272–4280, Outubro 2012. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6175144, doi:10.1109/TPEL.2012.2190525.

[20] Cian O Mathúna, Terence O'Donnell, Rafael V Martinez-Catala, James Rohan, e Brendan O'Flynn. Energy scavenging for long-term deployable wireless sensor networks. *Talanta*, 75(3):613–23, Maio 2008. URL: http://www.ncbi.nlm.nih.gov/pubmed/18585122, doi:10.1016/j.talanta.2007.12.021.

[21] VC Gungor e GP Hancke. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *Industrial Electronics, IEEE . . .*, 56(10):4258–4265, 2009. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4796311.

[22] Paolo Baronti, Prashant Pillai, Vince W.C. Chook, Stefano Chessa, Alberto Gotta, e Y. Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Communications*, 30(7):1655–1695, Maio 2007. URL: http://linkinghub.elsevier.com/retrieve/pii/S0140366406004749, doi:10.1016/j.comcom.2006.12.020.

[23] Byron Clemens, Teuta Cata, e Gary Hackbarth. Mobile Device Considerations for Supply Chain and ERP Related Systems. *IBIMA Publishing*, 2012(151480):16, 2012. doi:10.5171/2012.151480.

[24] Sharon P Hall e Eric Anderson. Operating Systems for Mobile Computing. *CCSC: Rocky Mountain Conference*, páginas 64–71, 2009.

[25] Philip Levis, Sam Madden, e Joseph Polastre. Tinyos: An operating system for sensor networks. *Ambient ...*, 2005. URL: http://books.google.com/books?hl=en&lr=&id=6eYnS8wnxkIC&oi=fnd&pg=PA115&dq=TinyOS+:+An+Operating+System+for+Sensor+Networks&ots=RPQMd3KtRH&sig=ZnKN3XmqKVCCoklulM4qIaxObmM.

[26] Mccarthy Blvd. *MEMSIC Sensor Boards (Datasheet)*. URL: http://www.investigacion.frc.utn.edu.ar/sensores/Equipamiento/Wireless/MTS-MDA_Series_Users_Manual.pdf.

[27] Panasonic. ERTJ1VR103J R-T (Datasheet). URL: http://industrial.panasonic.com/www-data/pdf2/AUA0000/AUA0000AE244.pdf.

[28] Mo Sha, Gregory Hackmann, e Chenyang Lu. Energy-Efficient Low Power Listening for Wireless Sensor Networks in Noisy Environments Categories and Subject Descriptors. *IPSN'13*, 2013.

[29] Thomas Potsch, Koojana Kuladinithi, Markus Becker, Peter Trenkamp, e Carmelita Goerg. Performance Evaluation of CoAP Using RPL and LPL in TinyOS. *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, páginas 1–5, Maio 2012. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6208761, doi:10.1109/NTMS.2012.6208761.

[30] David Gay, Matt Welsh, Philip Levis, Eric Brewer, Robert Von Behren, David Culler, e Shattuck Ave. The nesC Language : A Holistic Approach to Networked Embedded Systems Categories and Subject Descriptors. páginas 1–11, 2003.

[31] Nicolas Burri, Roland Schuler, e Roger Wattenhofer. YETI : A TinyOS Plug-in for Eclipse. 2006.

[32] Nicolas Burri, Roland Flury, Silvan Nellen, Benjamin Sigg, Philipp Sommer, e Roger Wattenhofer. Demo Abstract : YETI - An Eclipse Plug-in for TinyOS 2 . 1. páginas 295–296, 2009.