Faculdade de Engenharia da Universidade do Porto

Sistema Inteligente para controlo de iluminação suportado por redes de sensores sem fios

Bruno Filipe Branco Ferreira

Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Prof. Dr. José Ruela
Co-orientador: Engº Carlos Pinho

Janeiro de 2010

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2009/2010

A Dissertação intitulada

"SISTEMA INTELIGENTE PARA CONTROLO DE ILUMINAÇÃO SUPORTADO POR REDES DE SENSORES WIRELESS"

foi aprovada em provas realizadas em 18/Março/2010

o júri

Presidente Professor Doutor Luís António Pereira de Meneses Corte-Real
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Rui Silva Moreira
Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Fernando Pessoa

Professor Doutor José António Ruela Simões Fernandes
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

AUTOR: BRUNO FILIPE BRANCO FERREIRA

Faculdade de Engenharia da Universidade do Porto

Resumo

Este trabalho apresenta o estudo, arquitectura e implementação de um sistema inteligente de iluminação, com suporte de uma rede de sensores sem fios, com o objectivo de melhorar a eficiência energética de edíficos (particularmente escritórios e armazéns) tendo sido usado como cenário de teste o *open space* do 3º andar do edíficio do INESC Porto. O sistema procura melhorar os actuais sistemas de iluminação, apresentando uma solução dinâmica, baseada em perfis de iluminação definidos pelo administrador, que corresponde neste caso à percentagem de iluminação desejada. O dinamismo do sistema baseia-se na actuação em lâmpadas fluorescentes, tendo como base o perfil de iluminação definido pelo administrador, a obrigatoriedade de determinadas lâmpadas estarem sempre ligadas (quando existem pessoas), o conhecimento de presença/ausência de pessoas em determinados blocos e a iluminação vinda do exterior. Ao longo do documento é apresentada uma proposta de arquitectura do sistema, é descrita a sua implementação e são apresentados os resultados obtidos. Por fim, prova-se o funcionamento do sistema e determina-se a sua eficiência, com base nos resultados obtidos para diversos cenários de teste: em comparação com o sistema de iluminação actual do INESC Porto, o sistema porposto é bastante mais eficiente (poupando não só energia como a vida das lâmpadas). Como não foi feita a implementação do sistema de comutação das lâmpadas, não é possível quantificar de forma precisa o ganho em eficiência do sistema (i.e., cenários de teste real poderiam conduzir a variações nos resultados de simulação obtidos), mas ficaram identificados os factores essenciais na elaboração de um sistema inteligente de iluminação. Adicionalmente, e apesar de não terem sido feitos testes de satisfação dos utilizadores, crê-se que pelo facto do sistema operar em blocos e não redefinir a iluminação total do espaço em cada alteração (e.g., sinais de presença/ausência de pessoas) como apresentado em algumas soluções da literatura, tem potencial para criar uma boa base de satisfação dos utilizadores (i.e., as luzes não acendem e apagam frequentemente criando perturbações nos utilizadores no *open space*).

Abstract

This thesis presents the study, architecture and implementation of an intelligent lighting system, based on a wireless sensor network, used to improve the energy efficiency of the buildings (namely offices and warehouses with fluorescent lamps); the open space of the 3rd floor of the INESC Porto building was used in test scenarios. The system aims at improving the actual illumination systems, based on a dynamic solution, wich uses illumination profiles - in this case the desired illumination percentage. This document presents a proposal for the system architecture, describes the implementation and shows the experimental results. It is proved that the system works and the system efficiency is determined based on experimental results using different scenarios. Comparing with the current illumination system installed at INESC Porto, this proposal is more efficient, saving not only energy but also extending lamps lifetime. Since the switching system for the lamps has not been implemented, the real system efficiency cannot be quantified in a rigorous way (i.e., some differences could be found regarding the real test results and the simulated results), but the critical factors for the elaboration of an intelligent lighting system were identified.

Índice

Lista de figuras

Lista de tabelas

Abreviaturas e Símbolos

Lista de abreviaturas

AC	*Alternating Current*
ANA/CC	*Adaptive Neighborhood Algorithm using Correlation Coefficient*
DEEC	Departamento de Engenharia Electrotécnica e de Computadores
INESC	Instituto de Engenharia de Sistemas e Computadores do Porto
FEUP	Faculdade de Engenharia da Universidade do Porto
GUI	*Graphical User Interface*
JVM	*Java Virtual Machine*
PHP	*Hypertext Preprocessor*
PLC	*Power Line Communications*
UPnP	*Universal Plug and Play*
VLC	*Visible Light Communication*
WSN	*Wireless Sensor Networks*
ADC	*Analog-to-Digital Converter*
SO	Sistema Operativo
AM	*Active Messages*
SI	Sistema Internacional de Unidades
TXT	Ficheiro de texto

Capítulo 1

Introdução

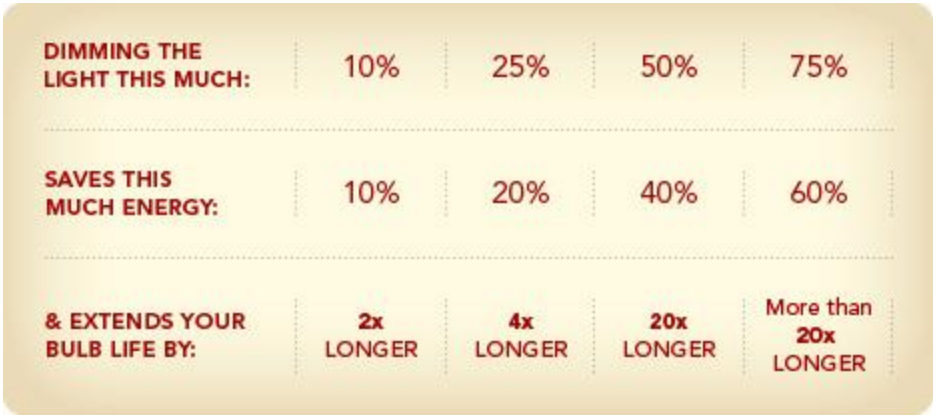
1.1. Contextualização

O consumo de energia apresenta uma tendência crescente, em boa parte devido ao rápido crescimento dos países em desenvolvimento, tal como aponta a *Frost & Sullivans,* dizendo que regiões como a China, Índia, Médio Oriente, América Latina e África irão contribuir para o crescimento insustentável de energia, já em 2030 [1].

Existem três problemas actuais, que necessitam de especial atenção a nível mundial e que constituem novos focos de trabalhos de investigação: custos dos combustíveis, as questões ambientais e o aumento do consumo de energia. Para isso é necessário que se promova um desenvolvimento sustentável.

Estes aspectos intersectam os objectivos do projecto, na medida em que com um sistema de iluminação inteligente promove-se a eficiência energética. De facto, o consumo de energia na iluminação não é negligenciável - o impacto da iluminação numa factura de electricidade doméstica nos Estados Unidos é de 25%, enquanto no sector comercial é de 60% [2].

Um sistema de controlo inteligente da iluminação permite poupanças económicas e ambientais não só pela redução de consumo como também pelo aumento da duração das lâmpadas. De seguida apresenta-se um quadro que relaciona a regulação da intensidade da luz, com o consumo de energia e a vida da lâmpada [3].



DIMMING THE LIGHT THIS MUCH:	10%	25%	50%	75%
SAVES THIS MUCH ENERGY:	10%	20%	40%	60%
& EXTENDS YOUR BULB LIFE BY:	**2x** LONGER	**4x** LONGER	**20x** LONGER	More than **20x** LONGER

Figura 1.1 – Tabela de relação entre a intensidade luminosa/poupança energética/vida das lâmpadas[3]

Analisando o quadro, verifica-se a importância da regulação da intensidade da luz no consumo energético, poupança das lâmpadas e consequente redução de impacto ambiental.

1.2. Motivação

O actual contexto macroeconómico mundial, nomeadamente o energético, que se caracteriza pela contínua subida dos preços da energia que possivelmente continuará a ocorrer no médio-longo prazo, bem como o aumento da consciência das pessoas para a conservação ambiental, está a despoletar um novo tipo de comportamento perante o consumo energético. O novo paradigma de consumo energético, que inclui o consumo de energia eléctrica, assenta no uso mais eficiente da energia, gerando benefícios para o utilizador e para o ambiente. Uma das componentes mais propícias à exploração do controlo e conservação da energia eléctrica é a iluminação. Apesar de existirem actualmente dispositivos relativamente baratos para controlar de forma simples a iluminação (i.e., sensores de movimento/presença capazes de ligar/desligar as luzes), este tipo de controlo ainda se apresenta como bastante limitado e pouco dinâmico perante as possibilidades oferecidas por tecnologias recentes que se apresentam como capazes de suportar sistemas mais eficientes.

1.3. Objectivos

Tendo em conta o contexto económico e ambiental descrito na secção 1.1, torna-se imperativo criar mecanismos inteligentes para o controlo do consumo energético de forma a, satisfazendo as necessidades de qualidade de vida actual, diminuir o consumo médio recorrendo ainda a uma maior percentagem de utilização de energias renováveis.

Os sistemas de controlo inteligentes podem ser aplicados em diversas áreas e vertentes. Dentro da área doméstica eles podem ser aplicados ao controlo de temperatura ambiente, aquecimento de água, iluminação, entre outros. Este trabalho foca-se na vertente de iluminação, apresentando-se de seguida os objectivos gerais e específicos deste projecto.

1.3.1. Gerais

- Identificar os factores críticos e desenvolver um sistema inteligente para controlo de iluminação suportado por Redes de Sensores Sem fios.

1.3.2. Específicos:

- Programar dispositivos com sistema de captação de luminosidade.
- Elaborar interface de comunicação entre interface gráfico e sensores.
- Criar interface gráfico com utilizador de gestão do sistema.
- Criar algoritmo de decisão, no controlo da iluminação.
- Considerar no algoritmo a iluminação actual e presenças.
- Desenvolver o sistema de actuação/controlo de lâmpadas em ZigBee.
- Desenvolver o sistema de interligação dos sensores de presença com o sistema global
- Testar o sistema completo no ambiente do *open space*.

1.4. Sistema inteligente de Iluminação

Um sistema inteligente de iluminação pode ser considerado um sistema em que várias lâmpadas estão ligadas a uma rede, e em que as necessidades em cada momento, os

requisitos e/ou preferências do utilizador/administrador são correspondidos pelo sistema de iluminação.

Num sistema inteligente de iluminação é necessário que haja controlo e que se automatize o controlo das lâmpadas; para isso, através de algoritmos sofisticados, o sistema só liga as luzes necessárias para uma determinada tarefa, sem necessidade de excesso de luz, conservando assim energia [4].

1.4.1. Caracterização e requisitos do sistema

Tendo em atenção os objectivos genéricos propostos, foi necessário caracterizar de forma rigorosa o problema a resolver. Este problema consistiu no desenvolvimento de um sistema de iluminação inteligente para actuar sobre lâmpadas fluorescentes e capaz de contemplar um índice de iluminação (definido pelo administrador), considerando matrizes de obrigatoriedade (i.e., lâmpadas que devem estar obrigatoriamente ligadas), as presenças das pessoas em zonas (e idealmente também a iluminação vinda do exterior), tendo como cenário de teste o *open space* do 3º piso do INESC Porto.

Figura 1.2 – *Open space* do 3º piso do INESC Porto

O sistema a desenvolver tem assim de ter em conta algumas características fundamentais: o tipo de lâmpadas não permite regulação da intensidade da luz (apenas *on*/*off*). O administrador do sistema deve ter acesso a uma interface gráfica de gestão e controlo, que proporcione o controlo da luminosidade em todo o espaço, definindo a percentagem de luminosidade que pretende (esta percentagem é referida ao máximo de

iluminação que se consegue com todas as lâmpadas ligadas e cujo valor em lux foi medido no cenário de teste). O sistema tem que ser dinâmico e autónomo, ou seja, definido um valor de intensidade luminosa, o sistema deve ajustar-se apenas quando necessário e num determinado intervalo de tempo ajustável pelo administrador. O sistema terá que receber as indicações de presenças, captar os valores de luminosidade em tempo real, verificar se os valores estão dentro do valor pretendido, e por fim proceder ao ajuste das lâmpadas se necessário. O sistema tem que ser obrigatoriamente melhor que a solução actual do espaço, no que respeita à poupança de energia e tempo de vida das lâmpadas.

1.4.2. Estratégia

Foi necessário definir uma estratégia que permitisse, de forma sistemática, resolver o problema proposto, demonstrar os conceitos e validar a solução adoptada. Começou por se fazer uma análise das tecnologias e soluções existentes no mercado, que serviu de base à especificação da arquitectura do sistema e respectivos componentes. De seguida procedeu-se à elaboração do protótipo funcional e por fim foi feita a validação do sistema com um conjunto de testes realizados em diferentes cenários.

1.5. Estrutura do documento

Este documento está dividido em cinco capítulos: no presente capítulo é feita uma introdução ao tema, indicando objectivos, motivação e contextualização. O segundo capítulo faz uma breve referência às tecnologias utilizadas num sistema de iluminação inteligente, e são também apresentadas alguns sistemas inteligentes de iluminação, sendo no fim apresentada uma proposta de arquitectura. No terceiro capítulo é descrito o sistema e a implementação dos principais componentes necessários à validação do conceito. No quarto capítulo, são descritos os testes realizados, os resultados obtidos e é feita a respectiva discussão. Finalmente, o quinto capítulo apresenta as conclusões e indica algumas possíveis melhorias a introduzir em trabalho futuro.

Capítulo 2

Análise de tecnologias e sistemas

Neste capítulo será feita uma abordagem sucinta às tecnologias utilizadas em sistemas inteligentes de iluminação, e serão também apresentados exemplos de sistemas inteligentes de iluminação que se situam no âmbito do presente trabalho.

2.1. Sensores

Numa primeira abordagem, é descrito o conceito de (*WSN)*, e um dos sistemas operativos mais utilizado nos sensores sem fios.

2.1.1. RSSF – Redes sensores sem fios

Redes de sensores sem fios são hoje usadas em diversos ambientes, sendo um dos principais no domínio da domótica, pois satisfazem os requisitos de comunicação dos diversos sensores com uma central de controlo, como é típico dessa área. A rede adopta normalmente uma topologia em malha (*mesh network*), em que cada nó comunica apenas com os seus vizinhos, no processo de troca de informação com a central. Assim sendo, cada nó comporta-se como um encaminhador de pacotes de informação recebida. [5]

Normalmente, quando as pessoas falam em dispositivos sem fios pensam em dispositivos como telemóveis ou computadores com 802.11 (Wi-Fi). Para além destes dispositivos custarem centenas de euros, são direccionados para determinadas aplicações, e estão dependentes de infra-estruturas de suporte previamente construídas para funcionarem. Pelo contrário, uma rede de sensores usa aparelhos pequenos e de baixo custo que podem ser usados em várias áreas, sem necessidade de construir infra-estruturas de suporte.

O desenvolvimento das redes de sensores sem fios converge para a junção de sensores, computação e comunicação num aparelho simples (mote). Estas redes procuram sempre que a informação chegue ao seu destino, não interessando o nó por onde passa, desde que atinja o seu destino. O poder destas redes baseia-se na possibilidade de espalhar aparelhos num espaço, e eles conseguirem organizar-se para adquirir a informação, processá-la e comunicá-la convenientemente. Como este tipo de aparelhos tem um custo muito baixo, pois o tipo de informação que processa é simples, pode ser usado para cobrir grandes áreas espaciais, e também usado em diversas áreas de aplicação.

Tipicamente são usados em cenários de monitorização das condições ambientais, localização em tempo real, monitorização do estado de estruturas ou equipamentos, etc.

Os aparelhos usados não necessitam de ser passivos, ou seja apenas recolherem informação, podem também controlar actuadores, interagindo assim com o meio (e.g., os módulos ZigBee tem por base uma *WSN*).

Os protocolos usados nesta rede têm de ser suportados por plataformas de hardware de baixo consumo energético, eficientes e flexíveis.

Apesar destas redes possuírem imensas vantagens, também têm algumas desvantagens a ter em conta, nomeadamente o facto de serem dispositivos pequenos implicando que a capacidade da bateria também seja pequena, sendo uma das principais características a ter em conta na elaboração de protocolos para estas redes. [6]

Um dos sistemas operativos criados para os dispositivos destas redes, que promove este tipo de abordagem é o TinyOS, descrito de seguida.

2.1.2. TinyOS

TinyOS é um sistema operativo *open-source*, desenvolvido para redes de sensores sem fios. O sistema operativo, as suas bibliotecas, e aplicações são programadas em nesC, linguagem que é considerada uma extensão do C. Apresenta uma arquitectura baseada em componentes, reduzindo assim o tamanho do código. O *TinyOS* pode ser instalado numa plataforma LINUX, ou Windows 2000/XP recorrendo ao programa Cygwin. [7]

A escolha deste SO para a generalidade dos sensores usados em rede, deve-se às limitações de hardware dos mesmos, necessitando ter em conta o consumo energético, a memória limitada, o processador lento, o tamanho, e a comunicação fazer-se via radiofrequência. O *TinyOS* consegue dar uma resposta a essas limitações, apresentando um sistema robusto, orientado a eventos, aumentando assim eficiência do sistema, pois só está activo quando é lançado um evento.

2.1.2.1. nesC

Como foi dito antes, a linguagem utilizada pelo SO é o nesC, ao nível da sintaxe é bastante parecida com o C, mas apresenta diferenças ao nível da estrutura, pois as aplicações nesC consistem em um ou mais componentes ligados entre si, sendo que esses componentes disponibilizam ou usam interfaces. Esses interfaces são os únicos pontos de acesso de um componente, e declaram um conjunto de funções, *commands* e *events*, que para serem utilizados têm de ser implementados no interface do componente que disponibiliza (*commands*) e no interface do componente que o utiliza (*events*). Um componente pode usar ou disponibilizar vários interfaces, e várias instâncias de um interface.

Ao nível da implementação dos componentes, existem dois tipos de ficheiros em nesC: módulos e configurações. Os módulos são constituídos pelo código da aplicação, implementando um ou mais interfaces. Os ficheiros de configuração são usados para "ligar" vários componentes entre si, nomeadamente interfaces usados por uns, e disponibilizados por outros. Concluindo, as aplicações nesC são compostas por 2 tipos de ficheiros, um de configuração, e outro com o módulo da aplicação. De seguida, é apresentado um extracto do código usado no projecto:

Configuração (Sensorx.nc)

```
configuration Sensorx { }
implementation
{
  components Main, SensorxM
           , TimerC
           , LedsC
           , DemoSensorC as Sensor
           , GenericComm as Comm;

  Main.StdControl -> SensorxM;
  Main.StdControl -> TimerC;
  //Main.StdControl -> Sounder.StdControl;

  SensorxM.Timer -> TimerC.Timer[unique("Timer")];
  SensorxM.Leds -> LedsC;
  SensorxM.SensorControl -> Sensor;
  SensorxM.ADC -> Sensor;
  SensorxM.CommControl -> Comm;
  SensorxM.DataMsg -> Comm.SendMsg[AM_OSCOPEMSG];
  //SensorxM.SounderControl -> Sounder;
}
```

Módulo (SensorxM.nc)

```
module SensorxM
{
  provides interface StdControl;
  uses {
    interface Timer;
    interface Leds;
    interface StdControl as SensorControl;
    interface ADC;
    interface StdControl as CommControl;
    interface SendMsg as DataMsg;
    interface ReceiveMsg;
  }
}
implementation
{
.
.
.

  /**
   * Signalled when the clock ticks.
   * @return The result of calling ADC.getData().
   */

  event result_t Timer.fired() {

      return call ADC.getData();

      }

  .
  .
  .
```

No código apresentado, pode-se notar as diferentes interfaces utilizadas (Sensorx.nc), tal como o código desenvolvido, neste caso para obter leituras do sensor de luminosidade (SensorxM.nc). A configuração é como um pedido aos componentes para utilizar as respectivas interfaces, e o módulo faz uso dos mesmos. [8] Neste caso verifica-se a necessidade de interfaces de *Leds, Timer, ADC, SendMsg, ReceiveMsg, CommControl, SensorControl*. Também se verifica o uso de eventos, nomeadamente neste caso quando o *timer* é activado, é lançado um evento que irá captar os dados do ADC.

2.2. Protocolos de comunicação

Nesta secção serão descritos os protocolos de comunicação que poderiam ser usados para a interface GUI – lâmpadas. Foram divididos em 2 tipos de soluções, com e sem fios.

2.2.1. Com fios

2.2.1.1. X10

X-10 é a norma adoptada para transmissão *Power Line*, sendo o protocolo base da arquitectura, e usado em grande parte dos equipamentos existentes no mercado. Foi introduzido em 1978, e é composto por dois módulos, um de transmissão, e outro de transmissão-recepção; ambos são ligados à rede eléctrica através das tomadas e ligam aos dispositivos por um cabo RJ11, típico de linha telefónica.

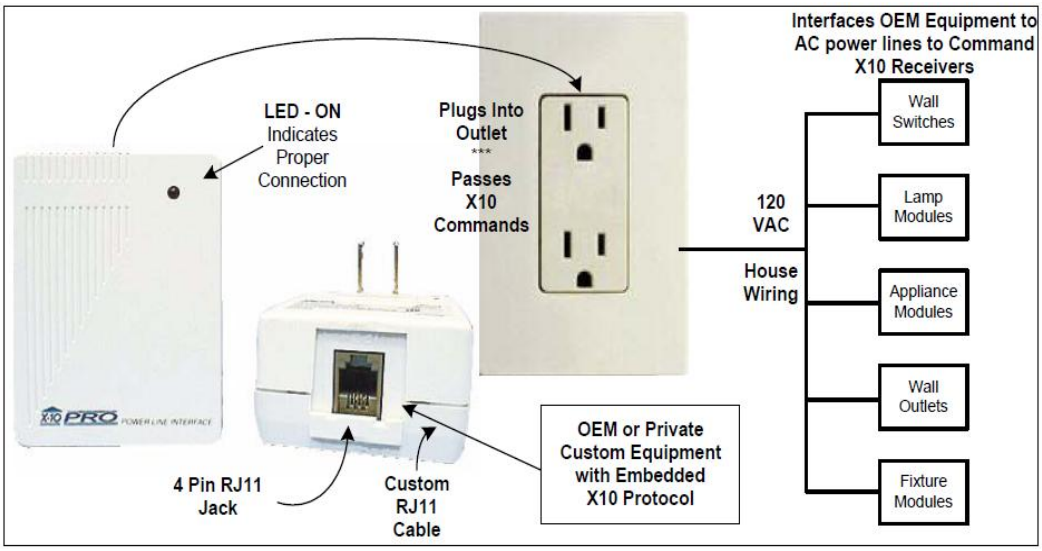


Figura 2.1 – Esquema de uma ligação usando o módulo PSC05[18]

As transmissões são sincronizadas no ponto de passagem por zero da rede AC, sendo o objectivo transmitir o mais próximo possível desse ponto. Os módulos PSC04 e PSC05 [9] transmitem através de uma onda quadrada de 60 Hz, sendo um número binário representado por uma sequência de 1 ms a 120 kHz.

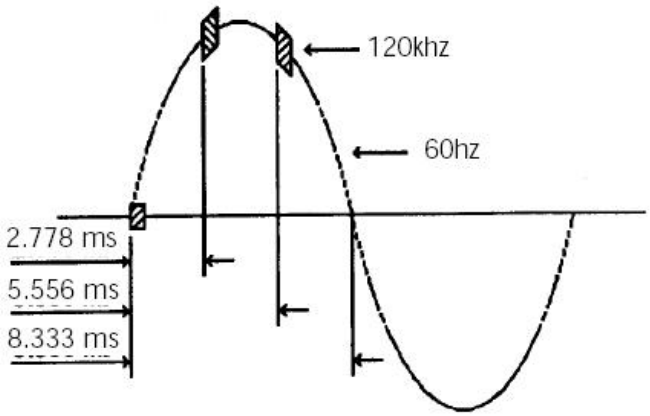


Figura 2.2 – Esquema das rajadas enviadas na rede eléctrica, assumindo um sistema trifásico.[18]

As tramas enviadas neste protocolo têm o seguinte formato:

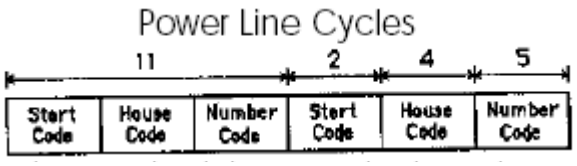


Figura 2.3 – Trama X-10[18]

Como se encontra ilustrado na Figura 2.3, é enviado primeiro um *Start Code* que indica início da trama, de seguida o *House code* e *Number Code*, responsáveis por indicar o tipo de função a executar, por exemplo, Ligar/Desligar luzes.

Entre as desvantagens deste protocolo, uma vez que a rede eléctrica é usada como meio de transmissão, refere-se a reduzida imunidade ao ruído, o que origina erros. Devido também à sua simplicidade, não permite realização de controlo avançado, e devido ao seu débito baixo as operações realizadas pelo mesmo podem tornar-se lentas.

De seguida, é apresentada uma lâmpada com um módulo X-10 instalado:

Figura 2.4 – Lâmpada com módulo X-10[18]

Na figura 2.4, é ilustrada uma característica importante, pois dada a forma da lâmpada, ela encaixa na maioria dos apliques existentes, não sendo necessário substituir componentes para introduzir controlo.

2.2.1.2. Homeplug

Homeplug™ Powerline Alliance é uma organização formada em Março de 2000, com o objectivo de criar normas para produtos baseados na rede eléctrica. Foram criadas duas normas de alta velocidade e banda larga para transporte sobre a rede eléctrica: Homeplug 1.0 e Homeplug AV.

HomePlug 1.0 foi criada primeiro: possuí uma taxa de transmissão de 14 Mbps, e para transmissão na camada física usa ODFM (*Orthogonal Frequency Division Multiplexing*), a mesma que é usada em tecnologia DSL.

HomePlug AV, que é mais avançada que a anterior com uma taxa de transmissão de 200 Mbps, sendo capaz de transportar informação multimédia e *streams* de HDTV através da rede eléctrica.

Apesar de apresentar débito elevado, este tipo de transmissão ainda não garante total qualidade de serviço, pois para já as soluções com tecnologia rádio ou WiFi encontram maior aceitação por parte dos consumidores. [10]

2.2.2. Sem fios

2.2.2.1. ZigBee

ZigBee é uma das normas mais conhecidas, usada em redes sem fios. Surgiu em Dezembro de 2004, e a entidade que se apresenta responsável por este protocolo intitula-se ZigBee™ Alliance. Visto que na altura, apenas existiam normas para débitos elevados (Wi-fi e Bluetooth), o ZigBee pretende associar a transmissão de dados sem fios a um reduzido consumo energético e com elevada fiabilidade. Tendo uma vasta área de aplicação, desde o controlo industrial à automação de residências (domótica), o protocolo possui então determinadas características que o tornam distinto dos restantes: reduzido consumo de potência, pilha protocolar de implementação simplificada, conduzindo a interfaces de baixo consumo, suporta uma elevada densidade de nós por rede, admite várias topologias de rede (estrela, malha, árvore) e baixa latência.

O ZigBee opera em três bandas de rádio, nomeadamente 2.4Ghz, 915 MHz (Estados Unidos da América) e 868 MHz (Europa). Consoante a banda, varia a taxa de transmissão possível, a 2.4 GHz podem ser obtidas taxas de transmissão de 250 kbps, com 16 canais disponíveis; a 915 MHz está disponível uma taxa de transmissão de 40 kbps e 10 canais de comunicação; no caso de 868 MHz, possibilita 1 canal e uma taxa de transmissão de 20 kbps.

Em termos de modulação, é utilizado O-QPSK (*Offset quadrature phase-shift keying*) para a banda 2.4 Ghz e BPSK (*Binary phase shift keying*) para 915 e 868 MHz.

Os nós de rede Zigbee apresentam dois modos de operação: *beaconing* e *non-beaconing*. *Beaconing* apresenta vantagem a nível energético, pois os nós Routers transmitem periodicamente sinalização (*beacons*) a confirmar presença aos outros nós da mesma rede, sendo que os outros nós só necessitam estar activos no momento da sinalização.

Os nós da rede são de 2 tipos: FFD (*Fully Function Device)* e RFD *(Reduced Function Device)*. Os nós FFD são usados em todo o tipo de redes, capazes de actuar como *coordinator*. Os nós RFD apenas actuam em redes com topologia em estrela e para um FFD, não podendo ser um *coordinator*.

Na figura 2.5 apresenta-se o esquema de rede utilizado nos módulos ZigBee.

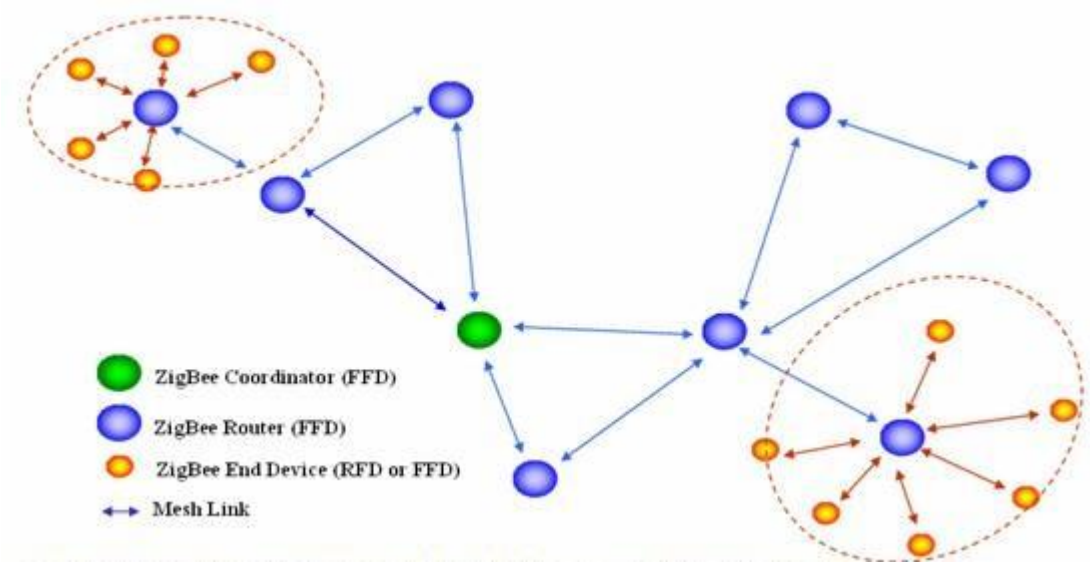


Figura 2.5 – Rede ZigBee[11]

Observam-se 3 tipos de nós de rede. O *Coordinator* e o *Router* são responsáveis por funções de gestão, tais como, encaminhar mensagens na rede e configurar a própria rede, e os *End devices*, que são nós com dispositivos de baixo consumo, fazem pedidos aos nós de gestão, e podem estar inactivos durante um largo período de tempo.

Por fim, destaca-se a pilha protocolar desta norma, com as duas primeiras camadas baseadas na norma IEEE 802.15.4, nomeadamente as camadas físicas e de acesso ao meio, sendo que as restantes (Rede e Aplicação) são definidas pela ZigBee™ Alliance. [11]

2.2.2.2. Z-Wave

Z-Wave é uma norma desenvolvida pela Zenzys, sendo que actualmente também é representada por um conjunto de fabricantes chamado Z-Wave™ Alliance. Como o ZigBee, é uma norma criada para soluções sem fios de baixo custo. Ao contrário do ZigBee apenas opera numa frequência (868.42 MHz) e para propagação do sinal recorre à modulação FSK (*Frequency shift keying)*. Podem ser obtidas taxas de transmissão de 40 Kbps. Apenas usa topologia de rede em malha, e as redes podem ter até 232 nós, enquanto no ZigBee podem ter até 64000 nós. [12]

2.3. Análise e selecção de tecnologias

Apresentadas algumas tecnologias envolvidas num sistema inteligente de iluminação, é possível retirar algumas conclusões relativamente às tecnologias que poderão ser mais apropriadas para o nosso sistema. Como o sistema terá de obter os valores de luminosidade de diversas lâmpadas, o uso de *WSN* parece o mais adequado, pois assim será possível ter diversos sensores espalhados pelas mesas do *open space*, sem necessidade de grandes modificações no estado actual. Recorrendo ao *WSN*, também se garante uma solução de baixo custo e baixo consumo energético. Relativamente aos protocolos de comunicação, uma solução sem fios é a melhor opção, pois as soluções com recurso à rede eléctrica ainda são demasiado sensíveis ao ruído, não garantindo a qualidade de serviço suficiente para o trabalho proposto, sendo que a solução ZigBee apresenta as melhores garantias.

2.4. Sistemas Inteligentes de Iluminação

Para verificar a melhor maneira de abordar o problema proposto, foi feita uma revisão da literatura e análise do estado-da-arte, no que se refere a sistemas inteligentes de iluminação. Pretendia-se verificar se existia alguma solução documentada capaz de resolver o problema proposto, analisar as eventuais limitações das soluções descritas na literatura e identificar soluções (ou meras sugestões) para os diferentes módulos envolvidos no projecto: rede de sensores, algoritmo de decisão, interface gráfica.

2.4.1. "A WSN-based Intelligent Light Control System Considering User Activities and Profiles"

Este projecto apresenta-se bastante semelhante ao proposto para esta dissertação, visto recorrer a uma rede de sensores sem fios, e também a perfis de utilizadores. Este projecto recorreu ao uso de redes de sensores sem fios, para determinar e controlar o nível de luminosidade de um espaço predefinido. Para isso, o espaço é subdividido em blocos, em formato de grelha, sendo cada espaço G_i (i =1,2,..n) composto por lâmpadas de iluminação geral, lâmpadas de iluminação local, e por um sensor fixo. Cada utilizador dispõe de um sensor sem fios, tornando o controlo mais dinâmico. De seguida, é apresentado o esquema de uma sala usada no cenário:

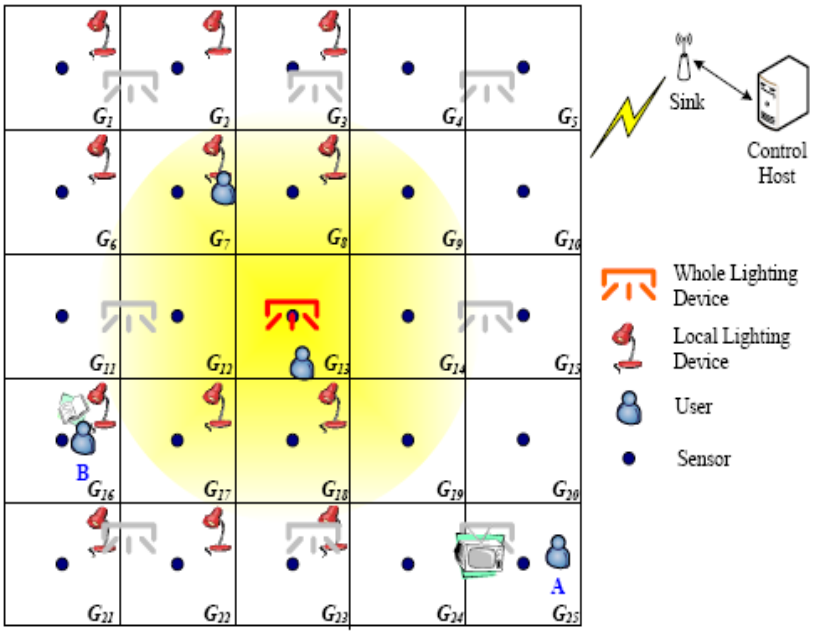


Figura 2.6 - Esquema do espaço a controlar iluminação[13]

Este sistema assumiu diversas variáveis, tais como intensidade da luz contribuída pelas lâmpadas gerais e locais, e intensidade da luz medida pelos sensores móveis do utilizador, valores esses que eram enviados para um sensor da rede chamado "sink", que efectuava a comunicação com a estação de controlo.

O valor da intensidade da luz num determinado ponto, era calculada através da seguinte fórmula:

$$Sf = W.Ld + Ssun \ , \qquad (1.1),$$

em que W é uma matriz, cujo conteúdo indica o peso de cada lâmpada no sensor, Ld é a intensidade da luz, e S_{sun} é a intensidade da luz solar medida nesse ponto.

A arquitectura de sistema proposta é a seguinte:

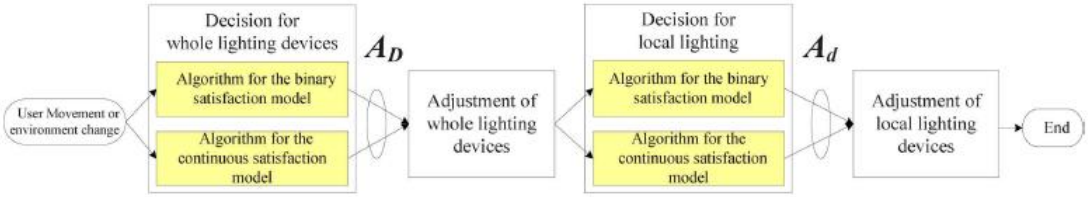


Figura 2.7 – Diagrama de blocos do sistema[13]

Numa primeira fase é tomada a decisão de controlo das lâmpadas gerais, e consequente ajuste, e na fase seguinte das lâmpadas locais. Também para cada ajuste, podem ser

utilizados dois tipos de algoritmos, um chamado "Modelo de satisfação Binário" e outro "Modelo de satisfação contínuo" cujas diferenças essenciais são: no algoritmo binário os requisitos de iluminação de cada utilizador são intervalos fixos de valores, e no contínuo não. Essas diferenças podem ser melhor compreendidas na referência [13].

De seguida, é apresentado o diagrama da arquitectura e dos componentes do sistema:

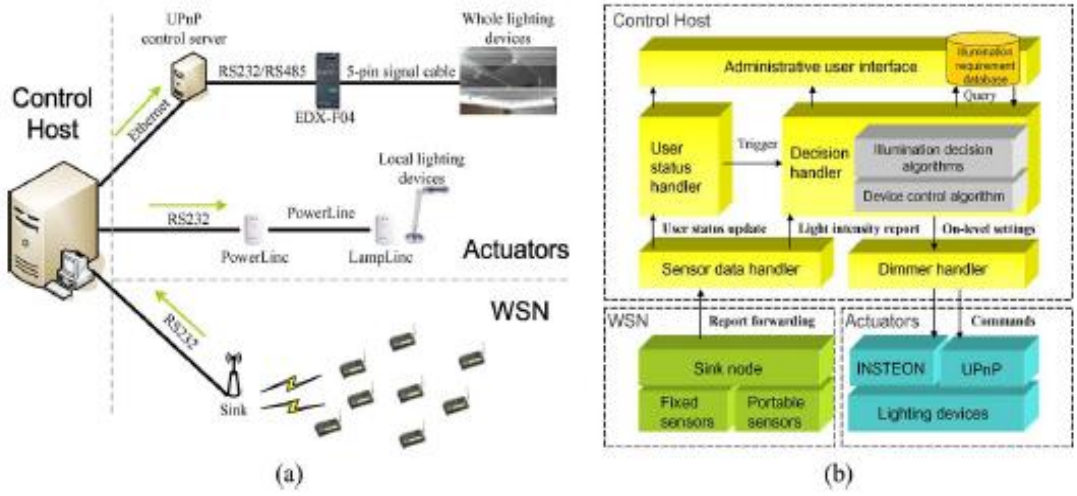


Figura 2.8 - (a)Diagrama de Arquitectura (b) Diagrama de componente [13]

Como se verifica na Figura 2.8(a), o sistema pode ser dividido em 3 partes: rede de sensores sem fios, actuadores, e sistema de controlo, a seguir descritos.

2.4.1.1. Sensores sem fios

Os sensores foram desenvolvidos usando Jennic JN5121 como módulo rádio, e fotodíodo Si como foto sensor. Os utilizadores podem indicar as suas actividades actuais carregando nos botões do sensor. Os sensores fixos e móveis enviam periodicamente informações para o "sink", encaminhando este depois para o sistema de controlo através de interface RS-232.

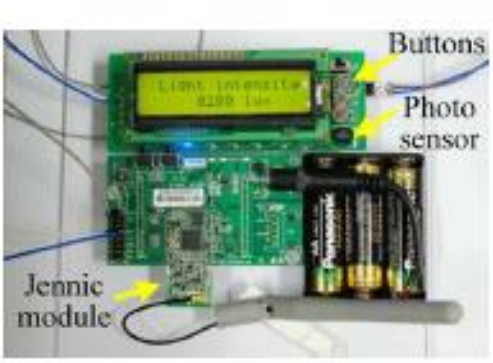


Figura 2.9 - Placa com sensores implementados[13]

2.4.1.2. Actuadores

No diagrama da arquitectura (Fig. 2.8(a)) pode observar-se que as lâmpadas gerais e locais utilizam meios de comunicação diferentes. As lâmpadas gerais usam o standard *UPnP Lighting Controls* V1.0 para controlo; o sistema de controlo envia comandos através duma interface Ethernet a um servidor de controlo UPnP que, por sua vez, envia para um regulador de intensidade da luz (*dimmer)*, neste caso o EDX-F04, que está ligado às lâmpadas gerais.

Para as lâmpadas locais, é usada a rede *PowerLine* para comunicar com os reguladores de intensidade, que neste caso são da INSTEON Lamp Linc [14]. É usado também um *link* entre a comunicação RS232, vinda do sistema de controlo, e a PowerLine, chamado *PowerLInc*. As soluções da *PowerLine* são fornecidas pela *SmartHome* [15].

2.4.1.3. Sistema de Controlo

O sistema de controlo está implementado em JAVA, e é constituído por 5 componentes:

- "*Sensor data handler*": módulo que recebe dados do "sink", para classificar como evolução do estado do utilizador, ou como um relatório de intensidade luminosa.

-"*User status handler*": este módulo por mudanças de estado do utilizador, e informa o "*Decision handler*" para efectuar as mudanças no estado das lâmpadas.

-"*Decision handler*": tem como função aplicar os algoritmos de ajuste relativamente às lâmpadas, mencionados anteriormente como "Modelo de satisfação binário" e "Modelo de satisfação contínuo". Estes algoritmos são implementados em MatLab, sendo depois inseridos no JAVA. Depois de tomadas as decisões, este envia para o "*Dimmer handler"*.

-"*Dimmer handler*": este *handler* serve de interface entre o sistema de controlo e os actuadores (UPnP e PowerLinc).

-"*Administrative user interface*": interface com o utilizador, em que é possível obter a localização dos utilizadores, sensores e lâmpadas, também configurar parâmetros do sistema, e obter leituras dos sensores, estados de ligações com os nós, etc.

Ao nível de desempenho do sistema, o autor argumenta que o sistema é bastante eficaz, tendo sido feitos diversos testes, em que as principais preocupações eram o grau de satisfação dos utilizadores e o baixo consumo de energia, não sendo sempre possível satisfazer os dois. [13]

2.4.2. "Intelligent Lighting System using Visible-Light Communication Technology"

Este sistema tem por base um sistema de iluminação inteligente, com um novo tipo de comunicação, através de luz visível, entre os sensores e o sistema de iluminação. Este tipo de comunicação tem algumas vantagens, pois a electricidade usada nas lâmpadas pode ser usada juntamente para a comunicação com equipamento simples. No entanto, pode ser necessário instalar filtros para impedir a propagação dos sinais de comunicação para as instalações vizinhas. Para este projecto foram usadas lâmpadas fluorescentes com tecnologia de comunicação de luz visível instalada.

Outro aspecto importante deste projecto é o facto de não haver nenhuma unidade de controlo, visto que, cada lâmpada tem um módulo de controlo próprio, usado para implementar o algoritmo, que neste caso é o algoritmo de adaptação de vizinhança usando coeficientes de correlação.

A arquitectura adoptada neste projecto é apresentada na Figura 2.10.

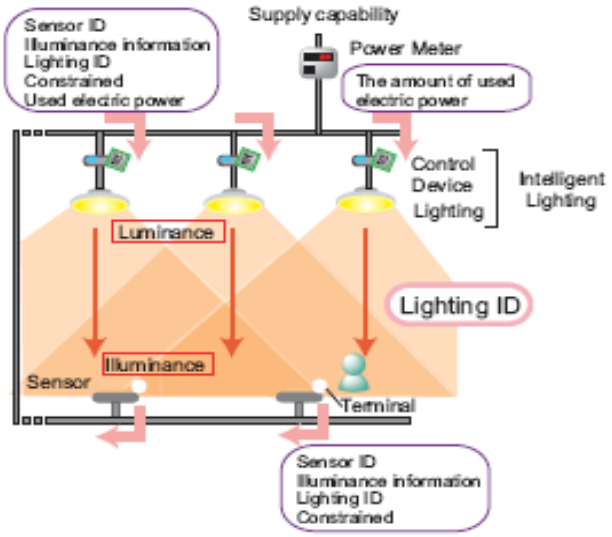


Figura 2.10 – Configuração do sistema de iluminação inteligente usando VLC [16]

As lâmpadas possuem dispositivos que transformam sinais eléctricos em luz visível, que seguidamente é modulada, para enviar informação para os sensores de iluminação, nomeadamente o identificador da sua posição. De seguida o sensor de iluminação envia informação de volta para o módulo da lâmpada, que procede à execução do algoritmo, através da correlação de dois factores, a iluminação da lâmpada e a iluminação medida no sensor. Este algoritmo procura minimizar o consumo de energia, e atender ao valor de iluminação indicado pelo sensor. Depois, através dos resultados do algoritmo, são accionadas as lâmpadas necessários para satisfazerem o utilizador, como se pode ver na Figura 2.11, onde estão indicadas as percentagens de funcionamento das lâmpadas, para melhor perceber que na zona do sensor as lâmpadas apresentam maior percentagem de intensidade luminosa que nas restantes.

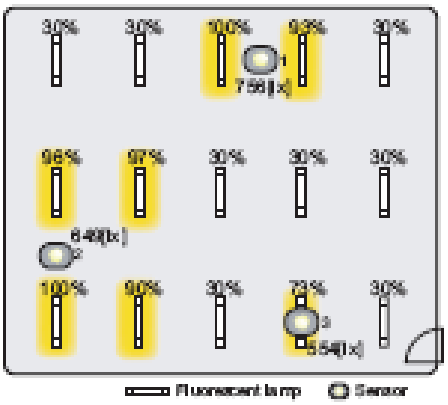


Figura 2.11 – Esquema de luzes e sensores indicando o seu valor requisitado de iluminação (lux) [16]

Ao nível de desempenho, o autor do projecto também argumenta que este se revelou bastante eficaz, pois o recurso à luz visível permite comunicação com as lâmpadas mais rápida e mais simples; também se observou uma boa aproximação entre o nível de iluminação requerido e o apresentado. [16]

2.5. Considerações finais

Analisado o estado da arte e tecnologias, todos os sistemas inteligentes de iluminação recorrem ao uso de lâmpadas com reguladores de intensidade luminosa, proporcionando algoritmos de decisão com base em equações matemáticas complexas, pois cada lâmpada pode ter vários níveis de intensidade luminosa. Não contemplam uma matriz de obrigatoriedade, nem resolvem o factor da ocupação do espaço (presenças). Têm mais em atenção o perfil de utilizador, ao contrário deste caso, pois apenas uma pessoa terá acesso ao sistema (administrador). Assim a implementação de qualquer um dos sistemas identificados não resolve o problema proposto. No entanto, algumas ideias para a abordagem do nosso problema podem ser aproveitadas e adaptadas para o nosso sistema, nomeadamente ao nível da arquitectura, que pode ser definida da seguinte forma, como ilustra a Figura 2.12:

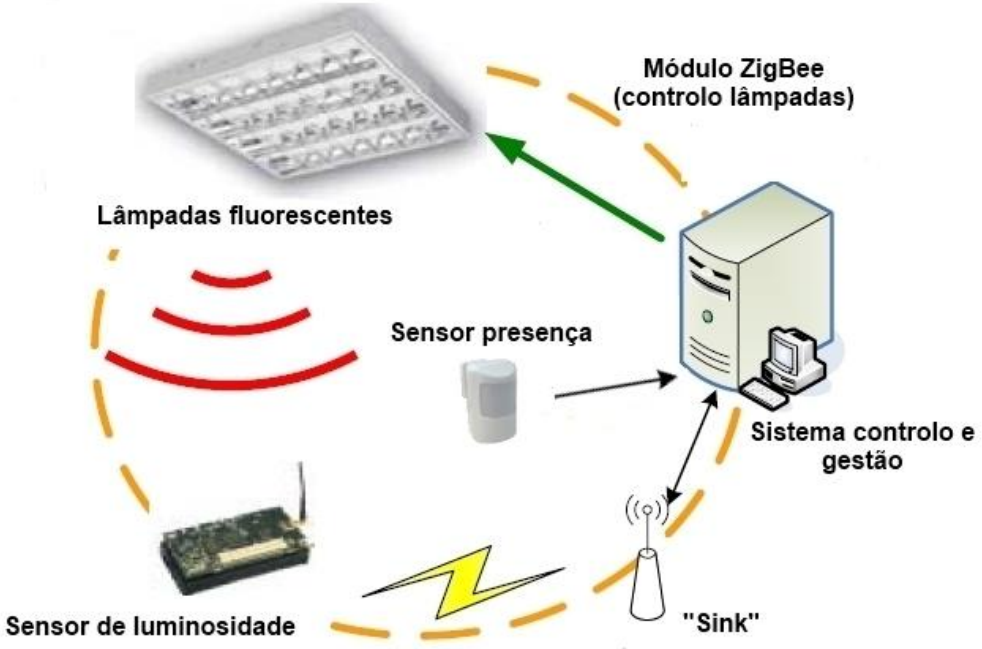


Figura 2.12 – Arquitectura do sistema

Será necessário recorrer a uma rede de sensores para captar o valor da intensidade luminosa em várias posições, sendo toda a informação enviada para um coordenador (*sink*) que por sua vez encaminha a informação para o computador. A fase seguinte, que no nosso

caso não foi implementada, seria o envio de comandos através de módulos ZigBee, ou através da rede eléctrica, recorrendo ao protocolo X10, para comutação do estado das lâmpadas. Visto que não foi implementado o sistema de comutação, foi dado um foco maior ao algoritmo de decisão do estado das lâmpadas. Para tratamento das lâmpadas, foi necessário definir-se um conjunto de matrizes para retratar o espaço e a posição de cada uma, caso contrário seria difícil identificá-las e tratar os dados convenientemente.

Tabela 2-1 – Matrizes utilizadas no desenvolvimento do algoritmo

Matriz	Objectivo
Matriz de saída	Identifica a posição e o estado das lâmpadas, depois do ajuste feito pelo algoritmo.
Matriz de Influências	Indica a contribuição de cada lâmpada numa determinada posição (sensor).
Matriz de Obrigatoriedade	Indica a posição das lâmpadas que o administrador quer manter ligadas, independentemente do ajuste do algoritmo.

Capítulo 3

Descrição do sistema proposto

Neste capítulo é apresentada a descrição do trabalho desenvolvido, dividida em dois módulos: o módulo da rede de sensores, e o módulo do sistema de gestão e controlo com a respectiva interface com o utilizador.

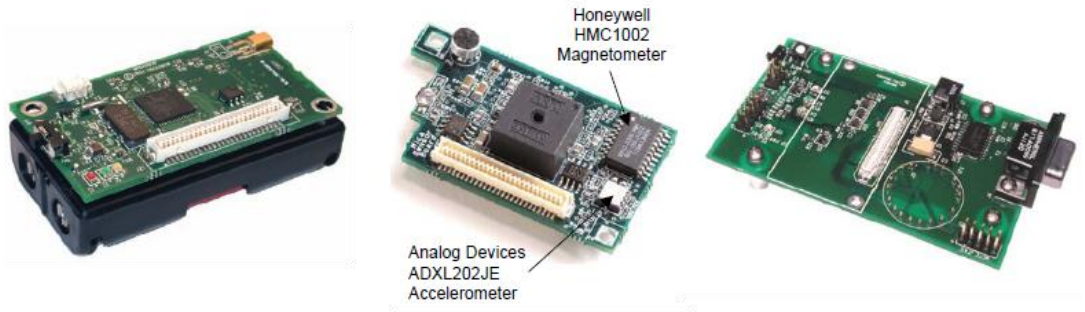
Relativamente ao primeiro módulo é descrita a instalação e distribuição dos sensores, bem como o código instalado nos mesmos. Relativamente ao módulo do sistema de gestão e controlo é descrita a interface gráfica com o utilizador, a interface de comunicação com os sensores, e por fim o algoritmo do sistema de iluminação. Antes da explicação técnica, será feita um breve introdução de como o problema foi proposto e qual a solução encontrada.

3.1. Introdução

Dado o conceito de sistema inteligente de iluminação, embutido num contexto de eficiência energética, foi proposto desenvolver um protótipo que melhorasse a eficiência energética de edifícios, nomeadamente escritórios, que apresentam sempre elevados consumos de energia. [2] No caso deste sistema, foi proposto o desenvolvimento de uma aplicação de controlo e gestão da luminosidade de um espaço do INESC Porto. Para isso, desenvolveu-se uma rede de sensores sem fios para captação da luminosidade, juntamente com uma interface gráfica, e um algoritmo de controlo e decisão, responsável pela inteligência do sistema, tendo como objectivo satisfazer os requisitos de luminosidade do utilizador, minimizando o consumo energético.

3.2. Rede de sensores

Para a implementação da rede de sensores recorreu-se a sensores sem fios da empresa Crossbow [17], nomeadamente módulos MICA2 integrados com placas de sensores MTS310, que possuem o necessário sensor de luminosidade. Para programação dos sensores e comunicação com o computador, usou-se uma placa de programação MIB510CA. De seguida são apresentadas algumas ilustrações dos módulos referidos anteriormente:



(a) **(b)** **(c)**

Figura 3.1 (a) MICA2 (b) placa de sensores MTS310 (c) interface MIB510CA [17]

A escolha destes módulos da *Crossbow* deveu-se essencialmente ao facto de serem portáteis, de apresentarem baixos consumos de energia, possibilidade de integração com placas compostas por diversos sensores e de estarem disponíveis no INESC Porto; neste caso, apenas se fez uso do sensor de luminosidade.

A interface de programação, que efectua a comunicação com o computador, recorre ao protocolo RS-232 para transferência de dados. Para programação dos sensores, os mesmos são colocados sobre a placa na interface de 51 pinos; no computador a programação dos mesmos recorre aos comandos do *TinyOS*. O comando para fazer a compilação do programa e carregamento no mote é o seguinte:

```
>>make mica2 install.(endereço_mote) mib510
```

A rede utilizada tem uma topologia em estrela, pois existe um mote (“sink”) para onde é enviada toda a informação dos restantes sensores, fazendo depois o encaminhamento para o computador. A escolha dessa topologia deve-se essencialmente ao facto do espaço de teste não ser demasiado grande, e o sensor com localização mais afastada do “*sink*” conseguir comunicar com o mesmo. Evitando assim o recurso a algoritmos de *routing* de pacotes entre os sensores.

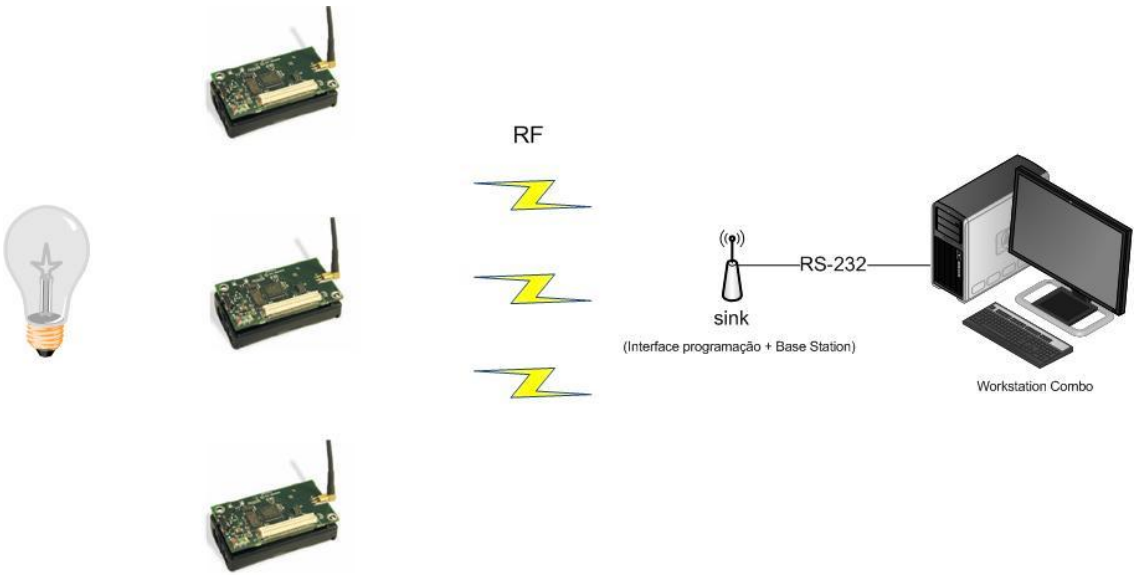


Figura 3.2 – Arquitectura do sistema

Para os motes, foram usados dois programas, um para os motes que recorriam à placa de sensores para medir a intensidade luminosa, e outro para a base central. O programa usado para captação da intensidade luminosa foi adaptado com base num programa já existente nos exemplos do *TinyOS*, nomeadamente o Osciloscópio [8]. O programa recorre à interface ADC (Conversor Analógico-Digital) do sistema operativo, para recolher os valores de tensão medidos pelo sensor, e transformá-lo num valor binário de 10 bits. A recolha dos valores acontece de acordo com um temporizador; foi definido um período de 0,1 segundos (por configuração, podendo estes valores ser alterados), e após recolha de 10 valores (correspondente a 1 segundo) procede-se ao envio de um pacote, contendo cabeçalho, informação, e CRC. O cabeçalho contém a identificação do pacote na rede, a informação contém as 10 leituras do sensor de luminosidade e a identificação do mote, e o CRC é usado para controlo de erros.

Para uma melhor explicação do programa, apresenta-se de seguida um fluxograma:

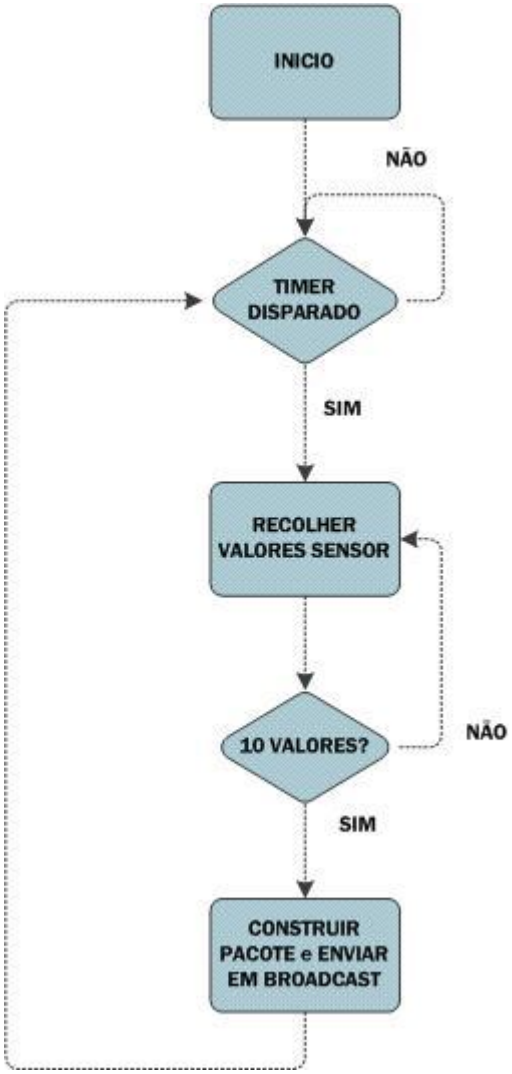


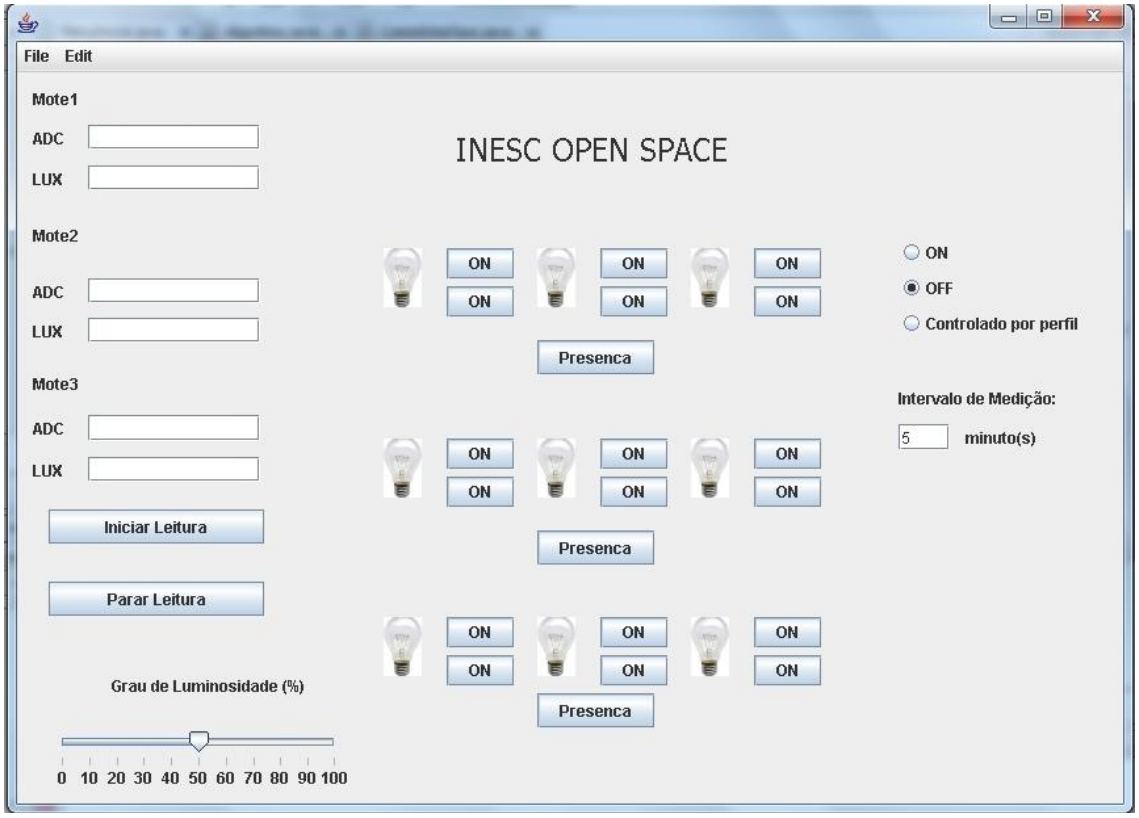
Figura 3.3 – Fluxograma dos programas dos motes com sensores

O outro programa instalado foi o da estação base, o "sink" de informação, que tinha simplesmente a função de encaminhamento dos dados recebidos via RF, para a porta UART do computador.

3.3. Sistema de gestão e controlo

3.3.1. Interface gráfica com o utilizador

A interface gráfica com o utilizador, permite ao administrador do sistema controlar o sistema de iluminação do espaço definido. De seguida, é apresentado um *screenshot* da interface gráfica:



Como se pode observar, a interface permite ao administrador obter informação em tempo real dos valores medidos pelo sensor, sendo também feita a conversão em tempo real dos valores dados pelo sensor para a unidade SI da luminosidade, lux (como vai ser visto mais à frente). Também é dada a possibilidade do administrador de ligar as lâmpadas que pretende, ou simplesmente ligar e desligar todas as lâmpadas. A função inteligente do sistema entra em funcionamento quando é activada a opção "Controlado pelo perfil", em que o administrador define o grau de luminosidade que pretende, definindo de seguida o intervalo de ajuste do algoritmo; assim, o sistema ajusta automaticamente o valor de luminosidade para o valor pretendido pelo administrador. De realçar que esta interface serviu para os testes de simulação real, e como tal não se achou necessário representar todas as lâmpadas do espaço usado para o teste, mas apenas uma zona, para comprovar a eficiência do algoritmo (daí também a interface apresentar apenas dados para 3 motes), pois as restantes zonas serão simétricas e funcionarão da mesma maneira. Também devido à não implementação dos sensores de presença, considerou-se para efeito de testes que seria o administrador a definir quem estava presente em determinada zona, podendo no futuro o sistema incorporar, com o mínimo de alterações, esta informação dinamicamente.

3.3.2. Comunicação PC-motes

A comunicação entre a interface JAVA e os motes, como foi antes referido, é através da porta UART usando o protocolo RS-232. Para isso, é necessário criar um canal de informação entre o JAVA e a porta série. O sistema operativo *TinyOS* possui classes que implementam a comunicação com os motes, nomeadamente *net.tinyos.Message.MoteIF* que, recorrendo a métodos de "escuta", tem como função determinar a presença de pacotes na porta série.

Passou-se à criação de uma classe (CommInterface.java) que, implementando a classe *MessageListener*, faz filtragem das mensagens, determinando se é do tipo OscopeMsg (nome que vem do exemplo que serviu de base à adaptação do programa que corre nestes dispositivos). Sendo a mensagem do tipo definido, é extraída a informação, tal como o identificador do mote, o endereço, e o campo de dados (10 medições de luminosidade). De seguida, é feita a média dessas 10 medições, e apresentado o valor na interface, isto é, o valor convertido em lux.

A comunicação é iniciada na interface gráfica, logo no inicio do programa recorrendo ao comando:

```
net.tinyos.bruno.CommInterface.MessageReceive listener = new net.tinyos.bruno.CommInterface.MessageReceive();
```

Este comando inicia o *listener* na porta série. De seguida, também são criadas instâncias para o tipo de mensagens a receber:

```
net.tinyos.bruno.OscopeMsg m = new net.tinyos.bruno.OscopeMsg();
```

Depois é instanciada uma classe do tipo MoteIF, responsável pela recepção e envio de mensagens para os motes; no construtor da classe são passados dois argumentos: o local para enviar mensagens de erro, neste caso para o *PrintStreamMessenger.err*, e o identificador do grupo dos motes (*Active Message group ID*), definido no nosso caso como 125:

```
net.tinyos.message.MoteIF mote = new net.tinyos.message.MoteIF(net.tinyos.util.PrintStreamMessenger.err, 125);
```

Numa outra fase do programa, nomeadamente quando é iniciada a leitura dos motes, é activado o *listener* de pacotes; no construtor da classe é passado o tipo de mensagem "m" e o *listener*.

```
mote.registerListener(m,listener);
```

3.3.2.1. MIG (Message Interface Generator)

Neste capítulo foi mencionado um tipo de mensagens *OscopeMsg*, que é uma classe que define a mensagem recebida pelo JAVA dos motes. Recorrendo a essa classe obtém-se a informação dos pacotes enviados pelos motes, sem necessidade de conhecer o protocolo de baixo-nível, apenas invocando métodos dessa classe que nos fornecem todos os parâmetros do pacote.

Essa classe é criada recorrendo ao MIG. MIG é uma ferramenta que é usada normalmente para gerar automaticamente classes JAVA que corresponde a mensagens do tipo *Active Message* (AM) que são usados nas aplicações dos motes.

Neste projecto, foi criado uma *Makefile* que efectua a compilação, para além da *OscopeMsg*, efectua a compilação da classe *CommInterface*, que recorre e é dependente da classe *OscopeMsg*.

Makefile:

```
TOS = $(shell ncc -print-tosdir)
PACKAGE = net.tinyos.bruno
APP = $(TOS)/../apps/Oscilloscope
MIG = mig java

# List of message classes to build
MSGS = OscopeMsg.java

INITIAL_TARGETS = $(MSGS)
OTHER_CLEAN = cleanmig

ROOT = ../../..
include $(ROOT)/Makefile.include

OscopeMsg.java:
      $(MIG) -java-classname=$(PACKAGE).OscopeMsg
$(APP)/OscopeMsg.h OscopeMsg -o $@

$(JAVAC) $@

      cleanmig:
rm -f $(MSGS)
```

E como evidenciado na *makefile*, a interface MIG recorre à pasta da aplicação *Oscilloscope* para criação da classe *OscopeMsg.java*. Isso deve-se ao facto de ser nessa pasta que está definida a estrutura da mensagem, o ficheiro *OscopeMsg.h*, que contém a informação dos campos a enviar nos pacotes usados nas aplicações dos motes.

OscopeMsg.h:

```
enum {
  BUFFER_SIZE = 10
};

struct OscopeMsg
{
    uint16_t sourceMoteID;
    uint16_t lastSampleNumber;
    uint16_t channel;
    uint16_t data[BUFFER_SIZE];
};

enum {
  AM_OSCOPEMSG = 10,
};
```

Analisando o ficheiro de estrutura, observa-se os campos de informação, nomeadamente o identificador da mote (*sourceMoteID*), um contador (*lastSampleNumber*), o canal usado (*channel)*, e um vector que no nosso caso irá conter as leituras do sensor data[BUFFER_SIZE] (*BUFFER_SIZE* = 10 leituras).

3.3.3. Conversão ADC-lux

Os valores lidos pelo sensor correspondem a um referencial de tensão, que depois é traduzido para um valor binário, neste caso de 10 bits, o que limita o valor máximo de ADC a 1023. Como a unidade SI da intensidade luminosa é o lux, e pode atingir valores entre 0 e 130.000 lux (exposição directa à luz solar), teve de ser obtida uma curva de conversão, pois a relação entre ambas não era linear. Para isso, foi necessário fazer medições com os sensores e um luxímetro, usando diferentes motes, diversos locais de medição no *open-space* para obter diferentes valores de luminosidade, e construir posteriormente um gráfico de relação entre ambos. Depois de feitas as medições, foi necessário fazer interpolações lineares entre dois valores de medições, de maneira a criar o gráfico.

De seguida, é apresentado o gráfico da relação entre os valores do sensor e os respectivos valores de intensidade luminosa (em lux):

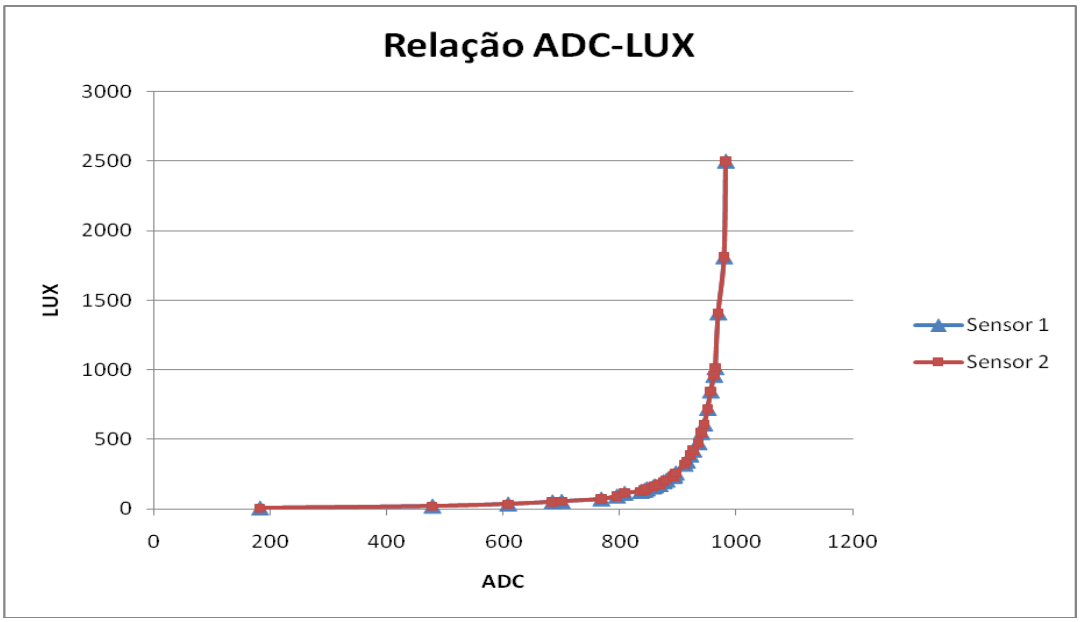


Figura 3.4 – Gráfico de relação entre valores e ADC

Através da análise do gráfico, verifica-se que a relação não é linear, e que para valores elevados a sensibilidade dos sensores não é tão grande como para valores mais baixos. Verifica-se também que os valores medidos entre os dois motes são muito próximos. De acordo com o que foi analisado na literatura, nem sempre esta condição se verifica.

Esta conversão é aplicada na interface gráfica para obter o valor de lux em tempo real. Na interface, são passados os valores das equações das rectas entre os pontos (i.e., é feita uma aproximação para valores de ADC que ficam entre duas medições), nomeadamente o ponto *b* e o declive da recta, calculando assim o valor de *y* (lux). De seguida é apresentado um excerto do código, como exemplo da conversão feita na interface:

MenuInicial.java

```
if (ADC > 890 && ADC <= 893) {
    lux = declive[14] * ADC + pontob[14];
```

Esta conversão também se revela essencial para o algoritmo, como será visto posteriormente, pois a intensidade luminosa é aditiva, ao contrário dos valores lidos pelos sensores.

3.3.4. Algoritmo de controlo de iluminação

O algoritmo utilizado pelo sistema é responsável pela decisão da quantidade de lâmpadas a ligar. Foi visto anteriormente que em todos os trabalhos citados e relacionados com este projecto se utilizavam reguladores de luz (*dimmers*), o que exigia algoritmos com equações matemáticas bastante complexas, mas que permitem chegar a valores óptimos de ajuste de lâmpadas para se chegar a um determinado objectivo de intensidade luminosa. Neste caso, como a lâmpada só pode ter 2 estados, ligada ou desligada (devido às restrições do sistema usado nos testes), a solução encontrada foi determinar qual a contribuição de cada lâmpada no sensor colocado numa determinada posição e, partindo da lâmpada com maior contribuição luminosa, somar as contribuições até alcançar o valor de luminosidade pretendido. Assim, obtido o valor de luminosidade pretendido, sabendo as posições das lâmpadas que contribuíram, passava-se ao accionamento das mesmas. No nosso caso, como não foi implementado o sistema de comutação do estado das lâmpadas, foi definido que o algoritmo passava a informação para um ficheiro *txt*, para se fazer a posterior análise de resultados.

Como as lâmpadas só tinham dois estados, criaram-se matrizes binárias para definir a sua posição e estado. A matriz de saída indica quais as lâmpadas a ligar, a matriz de obrigatoriedade indica quais as lâmpadas que o administrador quer que se mantenham ligadas mesmo antes de o algoritmo entrar em funcionamento. Outra matriz utilizada, mas neste caso não sendo binária, é a matriz de influências composta por valores de luminosidade em lux. Esta matriz corresponde à influência luminosa que um conjunto de lâmpadas, na vizinhança de um sensor colocado no centro das mesas, tem em cada sensor.

3.3.4.1. Matriz de influências

Esta matriz é muito importante no sistema, pois é através dela que o algoritmo define quais as lâmpadas a ligar. Foi criada através de duas medições nos sensores, uma para as bancadas das pontas da sala e outras para as restantes, em que bastavam 2 medições pois as restantes teriam influências simétricas. De seguida é apresentado um esquema ilustrativo da zona de testes:

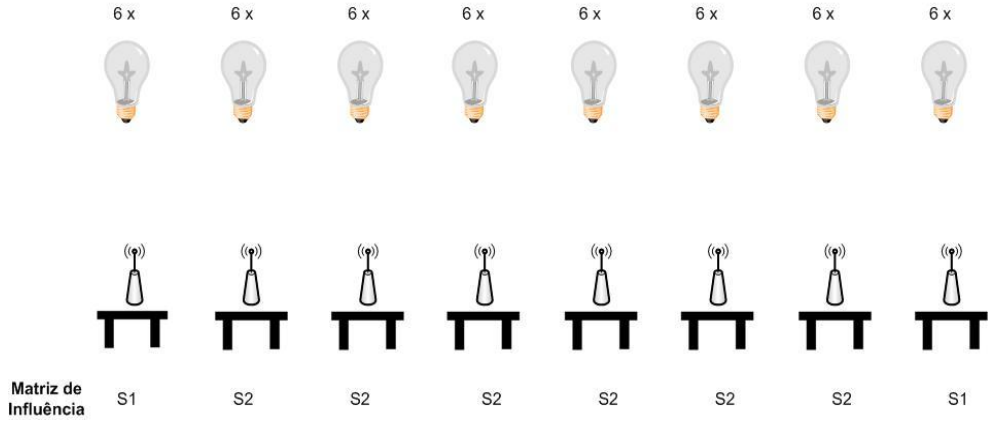


Figura 3.5 – Esquema da posição dos sensores e luzes no open-space do INESC Porto

Pode-se ver através da figura que o espaço é constituído por 8 mesas, sendo que cada mesa tem sobre ela um conjunto de 6 lâmpadas, e cada mesa possui um sensor colocado no centro. Determinou-se que cada sensor era afectado por 18 lâmpadas, e que as 2 mesas das pontas tinham uma matriz de influência S1, e as restantes mesas usavam a matriz de influência S2. As matrizes foram criadas medindo o valor da intensidade da luz com os motes, recorrendo à interface do sistema de gestão e controlo; as lâmpadas eram ligadas uma a uma, sendo registada a respectiva contribuição em cada sensor.

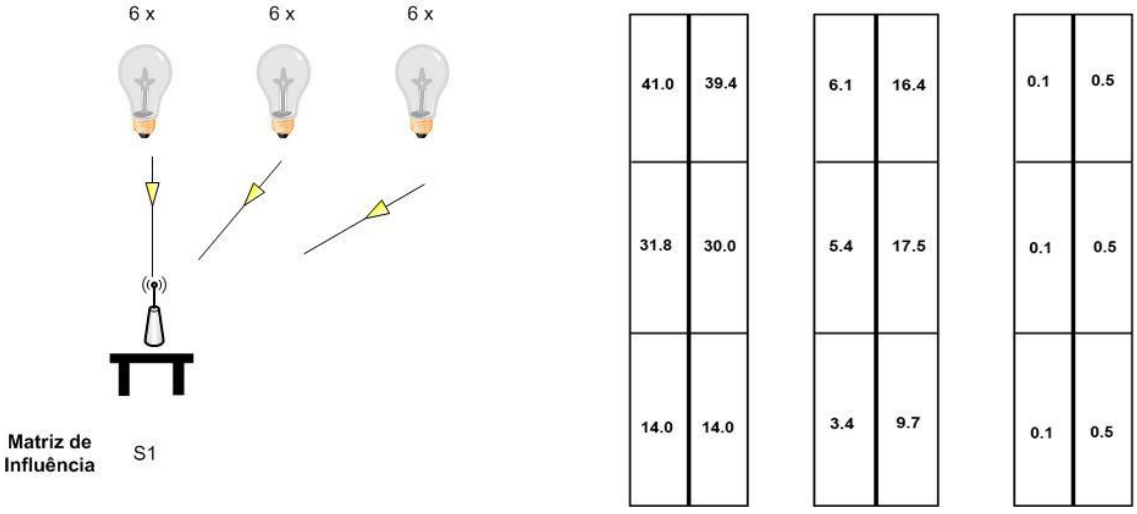


Figura 3.6 – Contribuições para a matriz de influência S1

Na figura anterior são ilustrados os valores (em lux) da contribuição de cada lâmpada na matriz S1. Note-se que as lâmpadas com maior contribuição deveriam ser as do centro, mas como o centro da mesa não se encontrava perfeitamente alinhado com o centro do conjunto de lâmpadas, verificaram-se ligeiras diferenças. De seguida ilustram-se as contribuições para a matriz S2:

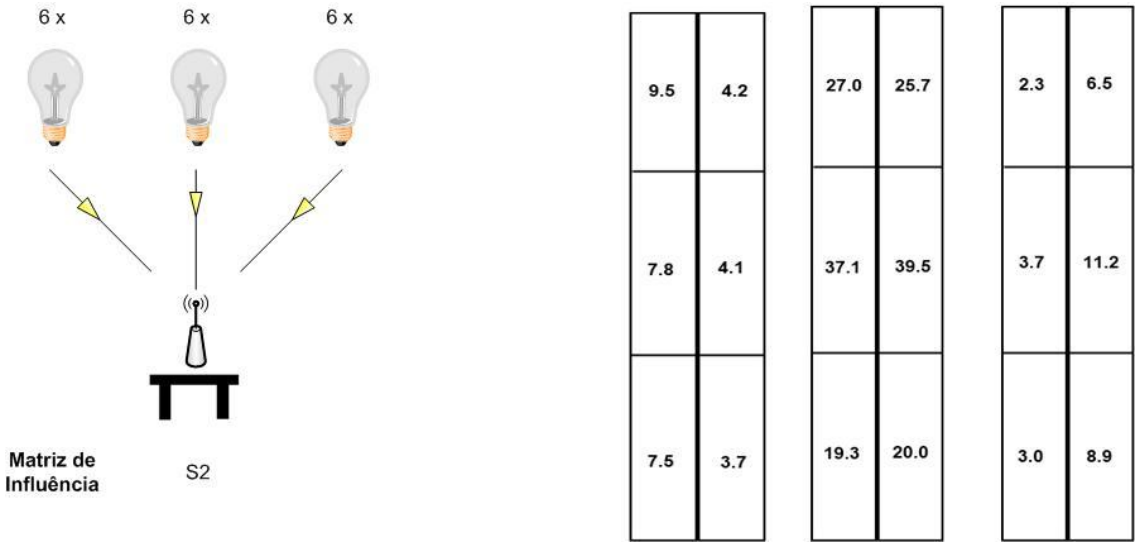


Figura 3.7 – Contribuições para a matriz de influência S2

Depois de definida as 2 matrizes de influências, é possível também determinar o valor máximo de intensidade luminosa para cada uma das matrizes. Assim é definido o valor de intensidade máxima que o administrador pode definir na interface. De seguida é demonstrado na tabela 3-1 como foram obtidos os valores da percentagem de luminosidade na GUI:

Tabela 3-1 – Relação entre percentagem e valor real de intensidade luminosa

Intensidade luminosa (%)	Matriz S1 (lux)	Matriz S2 (lux)
100	228.3	241.2
90	205.47	217.7
80	182.64	192.96
70	159.81	168.84
60	136.98	144.72

50	114.15	120.6
40	91.32	96.48
30	68.49	72.36
20	45.66	48.24
10	22.83	24.12

O valor máximo foi alcançado definido através das somas das contribuições das matrizes. Os restantes foram obtidos por uma regra de 3 simples.

3.3.4.2. Descrição

Para o algoritmo foi criada uma classe "Algoritmo", que era instanciada cada vez que o utilizador seleccionasse na interface a opção "Controlado pelo perfil", e expirasse o *timer* definido pelo administrador. Antes disso, o administrador já tinha definido qual a percentagem de luz que pretendia (0-100%); sendo essa variável pública e estática na classe do Menu, é acedida para obter o valor de intensidade luminosa pretendido. Dentro da classe Algoritmo, o método run() é automaticamente iniciado devido ao *timer*; esse método começa por atribuir a cada matriz de influência o valor respectivo das contribuições, tendo em conta a posição do sensor (garantindo uma matriz por cada posição) e limpa a matriz de saída (lâmpadas todas desligadas). De seguida faz o primeiro teste de condição, em que verifica as presenças nas zonas. Nos casos em que o valor do vector de presenças estiver a 1 (pessoa presente na mesa), verifica se a luminosidade actual se encontra dentro do intervalo pretendido (valor pretendido±10 lux – esta margem evita pequenas variações nos sensores por circulação de pessoas, evitando ajustes frequentes); se sim apenas copia para a matriz de saída a matriz de saída temporária (contendo o estado das lâmpadas anterior) e aguarda novo "*timer*", se não entra no método algoritmo, que é responsável pelo ajuste das lâmpadas. Numa primeira fase, o método verifica se existe alguma lâmpada ligada na matriz de obrigatoriedade, caso verifique que sim, reduz o contributo dessa lâmpada no sensor, ao valor pretendido (variável "target"). O extracto de código seguinte exemplifica esse caso:

```
for (i = 0; i < COLUNAS; i++) {
  for (j = 0; j < LAMPADAS; j++) {
    target = target - (matrizInfluencia[i][j] *matrizOb[i][j]);
    totallux=totallux+(matrizInfluencia[i][j]* matrizOb[i][j]);
    if (target <= 0) {
      matrizsaida[i][j] = 1;
      escreverFicheiro(matrizsaida,"resultados.txt",totallux);
      run();
```

```
} else if (matrizOb[i][j] == 1) {
    matrizsaida[i][j] = 1;
}
```

De notar que, caso o administrador exija (através da matriz obrigatoriedade) que as lâmpadas da zona tenham de estar ligadas, e se essas lâmpadas alcançarem o valor de luminosidade pretendido, o método termina sem necessidade de ajustes posteriores.

Como já foi visto, o objectivo principal deste método é, estipulado um valor da variável "*target*"(valor de luminosidade pretendido), ir subtraindo valores da matriz de influência à variável ao longo do método, até que se atinja o valor 0 ou ligeiramente abaixo dele. Existem dois valores possíveis para o "*target*" máximo do sistema: Um corresponde à soma das contribuições da matriz S1 (Figura 3.6), e outro à soma das contribuições da matriz S2 (Figura 3.7). As restantes percentagens são obtidas através do valor máximo.

Depois de passada a fase da matriz de obrigatoriedade, o método tem que descobrir qual a lâmpada com maior contribuição para aquele sensor, e verificar se são necessárias mais lâmpadas ligadas ou não. O primeiro passo consiste em ordenar a matriz de influências da maior contribuição para a mais pequena, pois assim obtinha-se maior contribuição de intensidade luminosa com menor número de lâmpadas possível (maior eficiência energética).
Para isso, foi necessário passar os valores das matrizes e os índices (que correspondem à posição das lâmpadas) para vectores, e depois aplicar um algoritmo de ordenação de vectores. Neste caso optou-se pelo algoritmo *bubblesort*.

Depois de ordenado o vector, e com as correspondentes posições das lâmpadas, é só necessário aplicar o método de subtracção do "*target*" anterior.

Por fim, com o valor de "*target*" a 0, o método identifica qual o sensor que fez o pedido de ajuste, e passa a "1" as posições da matriz de saída necessárias. Antes da escrita no ficheiro, é feita uma cópia da matriz de saída para uma matriz temporária ("matrizsaidabackup"), para no próximo ajuste não desligar lâmpadas que apesar de lá estar alguém (presença=1), não necessitem de ajuste. O algoritmo termina com a escrita no ficheiro da matriz de saída e da intensidade luminosa total, em lux.

Resumindo, podemos verificar melhor o funcionamento do programa recorrendo ao fluxograma seguinte:

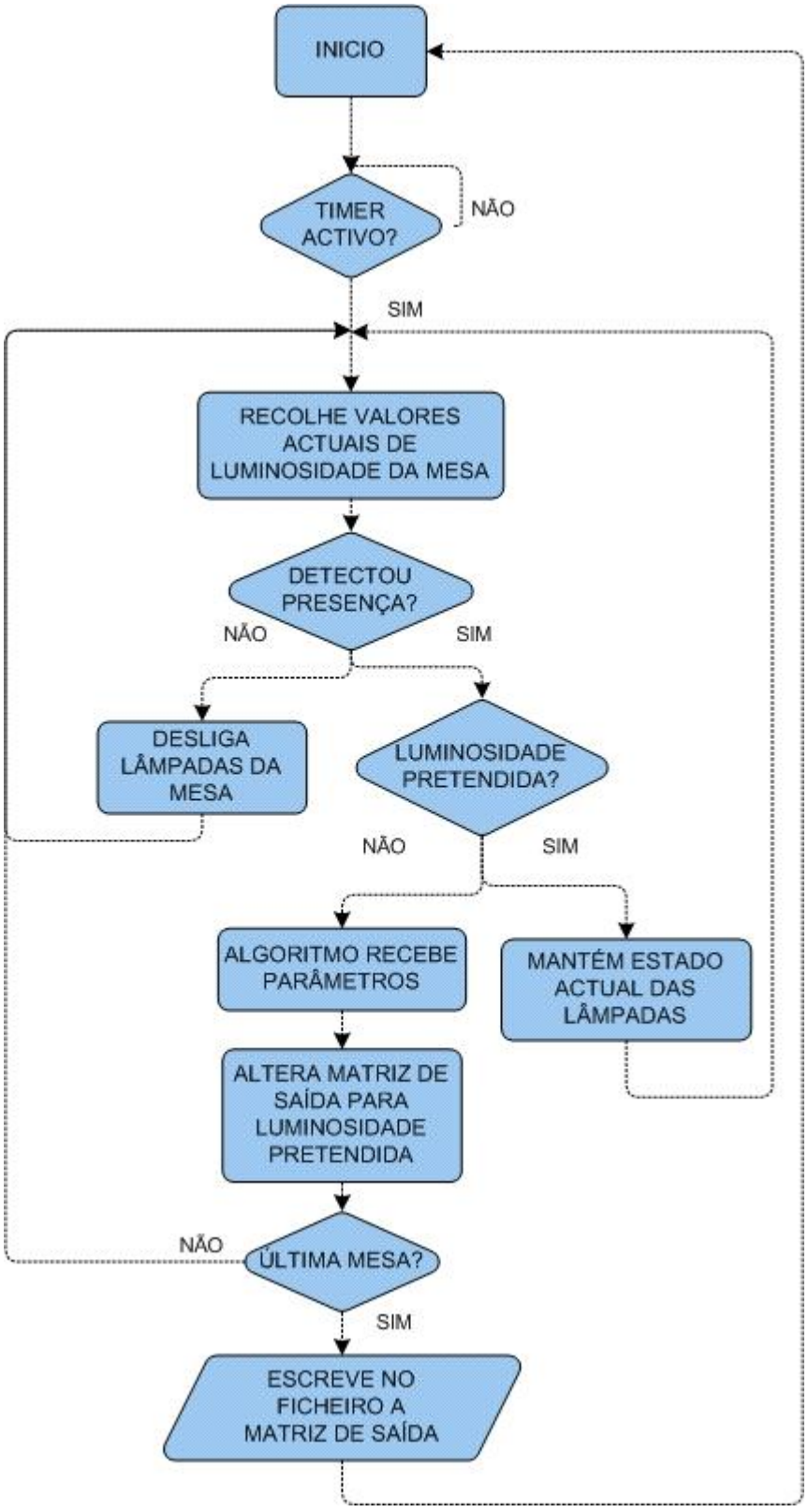


Figura 3.8 – Fluxograma do algoritmo de controlo da iluminação

Capítulo 4

Testes e resultados obtidos

Neste capítulo serão apresentados os testes efectuados em diferentes cenários de iluminação, os resultados correspondentes e a respectiva discussão.

4.1. Introdução

Para efectuar a validação da interface e do algoritmo, optou-se por efectuar dois tipos de testes. Os primeiros testes são feitos em cenário real, usando os motes, e provando a interacção dos motes com a interface, e o dinamismo que o algoritmo apresenta no sistema. Na segunda parte de testes são considerados vários cenários para verificar a eficiência do algoritmo, variando as presenças nas mesas, e a percentagem de luz pretendida, fazendo por fim sempre uma comparação com o cenário actual de referência, ou seja, com as lâmpadas todas ligadas.

Para os testes, os programas desenvolvidos, e disponíveis em anexo, são os programas escritos em JAVA, e os programas escritos em nesC. Em JAVA, existem 3 classes:

- MenuInicial.java – Classe referente ao interface gráfico com o utilizador.
- CommInterface.java – Classe onde são tratados os pacotes recebidos dos motes.
- Algoritmo.java – Classe responsável pelo algoritmo de controlo das lâmpadas.

Nos motes, existem 2 programas:

- OscilloscopeRF – Programa que envia periodicamente leituras do sensor de luminosidade através de RF.
- TOSBase – Programa que serve de ponte entre os motes e o computador. Recebe pacotes via RF e encaminha pela porta série.

As percentagens de luz definidas no interface têm um alcance de 0 a 100%, podendo ser modificado de 10 em 10 %. Foi definido o valor máximo de 100% com base na soma das

contribuições das duas matrizes, sendo para uma o "*target" de* 241,2 lux (Matriz S1) e para outra 228,3 (Matriz S2).

A matriz de saída é a saída do sistema, o resultado da decisão efectuada pelo algoritmo de quantas lâmpadas devem ser ligadas para o valor de luz requerido. Essa saída é apresentada sob a forma de um ficheiro *txt*, como está indicado de seguida:

Resultados.txt:

```
Data: 2010-01-27 21:50:34.311

0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
1 1 1 1 0 0
1 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:258.59999999999997
Lâmpadas ligadas: 8
```

É possível verificar que é demonstrada a matriz de saída, representando as 8 mesas (linhas) e 6 lâmpadas (colunas), a soma total das contribuições de cada lâmpada em lux, e número de lâmpadas ligadas. Cada teste é identificado por um *timestamp* (data e hora actual).

4.2. Simulação Real

Na simulação real, onde já existe interacção dos motes com o interface, é necessário configurar o computador para comunicar com os motes. Para isso, foi necessário um computador com Windows XP e o programa *Cygwin* instalado, onde se passou a instalação dos seguintes pacotes:

- Java SDK 1.4 – Necessário para compilar os programas em JAVA.
- Compiladores nativos dos motes MICA2 e nesC – Os compiladores dos MICA2 para gerar o código assembly correcto, e nesC pois o *TinyOS* é programado nessa linguagem.
- A árvore directórios do *TinyOS* 1.1 – contendo todos os programas, componentes, interfaces, e código java necessário para desenvolvimento das aplicações para os motes.

De seguida, é necessário configurar variáveis de ambiente, nomeadamente a MOTECOM, que identifica qual a porta série com que o sistema vai comunicar.

Depois de configurado o sistema, é possível iniciar a comunicação com os motes, compilando os programas JAVA e iniciando o interface.

4.2.1. Cenário de teste do sistema implementado

O sistema implementado, no qual apenas faltou implementar um módulo (interface JAVA – lâmpadas), permite verificar que os motes comunicam com a interface gráfica, indicando o valor real de ADC e intensidade luminosa (em lux), e que o programa responde às alterações de intensidade luminosa convenientemente, estando determinado um "*target*" de lux pretendido. O valor de intensidade luminosa em lux, convertido pelo programa, é o que é usado pelo sistema para comparar com o "t*arget*" pretendido. A fase que terá um contributo essencial para o desempenho do sistema será a fase de calibração do sensor, ou seja, da conversão dos valores de ADC para lux. Pois cada sensor tem as suas características, e pode estar sujeito a diferentes temperaturas, níveis de bateria, factores que influenciam as leituras do sensor, e introduzem erro no sistema. A forma de minimizar esse erro foi introduzindo uma margem de erro na actuação do algoritmo (*target*±10) garantindo assim que o valor real poderia estar ±10 lux relativamente ao pretendido, evitando que em cada período de verificação seja necessário estar sempre a alterar o número de lâmpadas ligadas, melhorando assim a satisfação do utilizador.

Outro erro associado ao sistema implementado está nas matrizes de contribuição, pois estas foram obtidas recorrendo a medições efectuadas com uma lâmpada auxiliar e o luxímetro, podendo ter influenciado nas contribuições de cada sensor.

Será demonstrado que para um determinado valor de luminosidade pretendido, o sistema conseguirá verificar se o valor nos motes está dentro do limite estipulado e caso não se verifique, actuará o algoritmo de decisão que ilustrará quais e quantas lâmpadas serão necessárias para alcançar aquele valor de luminosidade.

4.2.1.1. Cenário do sistema implementado

Intensidade luminosa pretendida: **60% (144.72/136.98 lux)**
Intensidade luminosa real: **Variável**
Presenças: **Mesa 2**
Período de ajuste: **30 segundos**
Motes em funcionamento: **Mote 2**

Figura 4.1 – Valor lido pelo luxímetro aproximadamente na mesma posição do mote

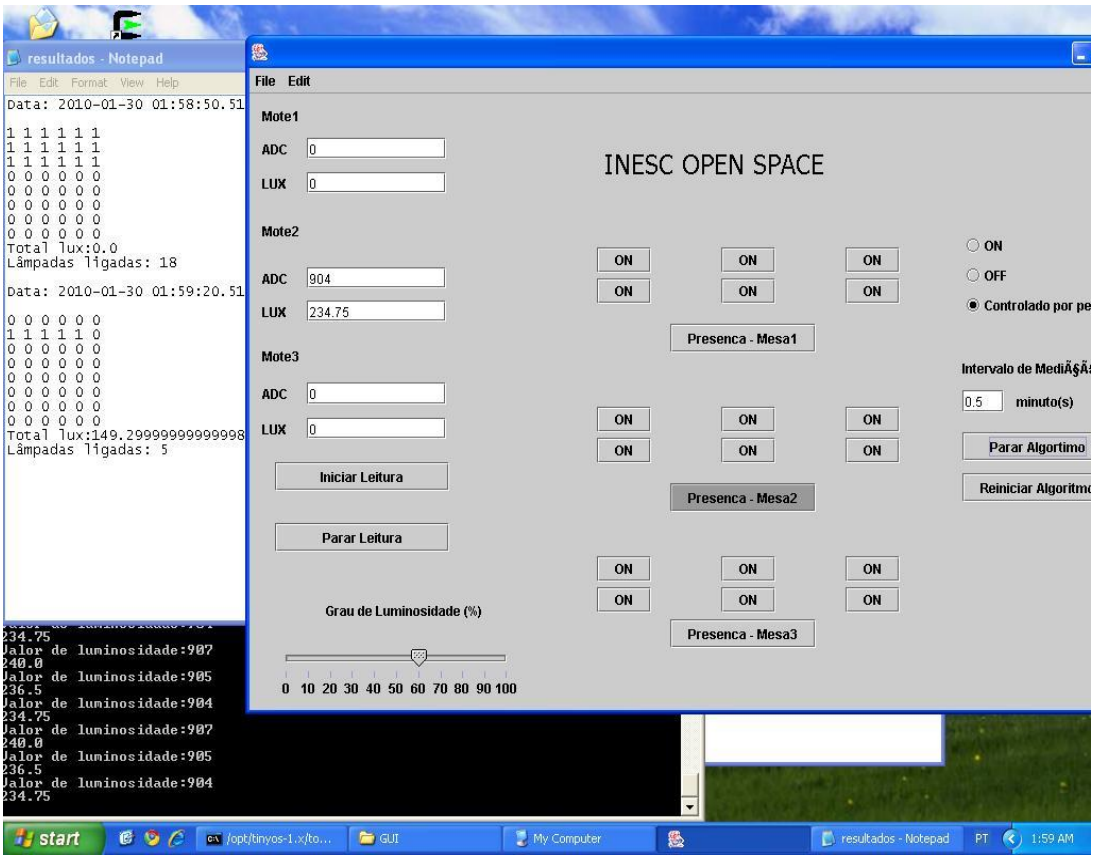


Figura 4.2 – *Screenshot* da aplicação a executar em tempo real

O teste realizado tem como objectivo provar o funcionamento do algoritmo em tempo real, como ilustra a Figura 4.1 e 4.2. Na Figura 4.1 é ilustrado o valor de intensidade luminosa em tempo real recorrendo ao luxímetro, na Figura 4.2 é ilustrado o valor lido na interface de

gestão e controlo depois de feita a conversão em tempo real do valor recebido do mote. No instante ilustrado as lâmpadas encontram-se todas ligadas. A pequena diferença de valores deve-se ao erro associado à curva de calibração do sensor. No canto superior esquerdo, está o ficheiro com a matriz de saída, que se pode observar que no primeiro instante (1:58:50) a matriz apresenta a saída para 241,2 lux. No instante seguinte (30 segundos depois), o administrador alterou a percentagem de intensidade luminosa para 60 % (144.72 lux) e verifica-se no ficheiro que o algoritmo altera a matriz de saída para o valor pretendido, pois o valor lido pelos motes está bastante acima do valor pretendido. Essa matriz de saída é criada através dos valores da matriz de influência para aquele sensor, através do qual o algoritmo identifica a posição das lâmpadas com maior contribuição de intensidade luminosa para o valor pretendido, colocando no ficheiro de saída o valor "1" para a posição das lâmpadas a serem ligadas.

O aspecto da interface está diferente de o que foi ilustrado anteriormente, pois o computador de testes usava uma versão do JRE (*Java Runtime Environment)* diferente da usada na construção da interface.

4.3. Eficiência do algoritmo

Para testar a eficiência do algoritmo, e uma vez que não foi implementado o sistema de comutação das lâmpadas, optou-se por usar valores simulados de luminosidade e de ocupação das mesas. Assim se comprova como o algoritmo melhoraria a eficiência energética.

Serão apresentados alguns cenários de testes feitos, com a variação de diferentes parâmetros, como a percentagem de luz requerida e a distribuição das presenças. Ainda será descrito um teste com a aplicação da matriz de obrigatoriedade.

De seguida é ilustrado um diagrama do algoritmo desenvolvido:

Figura 4.3 – Diagrama de entrada e saída do algoritmo do sistema

4.3.1. Comparação com a situação actual

Como foi visto anteriormente, a zona de testes é o *open space* do 3º andar do edíficio do INESC Porto; é composto por 8 mesas, onde seriam colocados 8 sensores (situados no centro das mesmas). Por cima de cada mesa, encontram-se 6 lâmpadas.

Normalmente, as lâmpadas encontram-se ligadas durante todo o dia, dentro do horário laboral, o que perfaz um total de 48 lâmpadas ligadas. Uma vez que cada lâmpada consume 36 W, o consumo total horário é de 1,728 kWh.

Para testar o desempenho do algoritmo, foi simulado um cenário em que o algoritmo é posto em funcionamento desde o inicio da manhã até ao final do dia de trabalho. Assumiu-se que o valor de luminosidade nunca era o pretendido e assim em todos os períodos era necessário fazer ajustes. Assumiram-se também os seguintes parâmetros:

- 6h30 – 9h30 → Ocupação = **3 mesas**
- 9h30 – 12h00 → Ocupação = **8 mesas**
- 12h00 – 14h00 → Ocupação = **4/5 mesas**
- 14h00 – 18h00 → Ocupação = **8 mesas**
- 18h00 – 20h00 → Ocupação = **3 mesas**
- Intensidade luminosa = **100%** e **70%**
- Período de Ajuste: **30 segundos** (correspondendo a meia hora no tempo real para acelerar as simulações)

Os resultados obtidos são apresentados no Anexo A1 e A2, poderá ser observado com mais detalhe o funcionamento do algoritmo. Depois de feitos os testes, e conhecido o número de lâmpadas activas em cada período, é possível observar através dos gráficos o ganho na poupança de energia comparado com a situação actual:

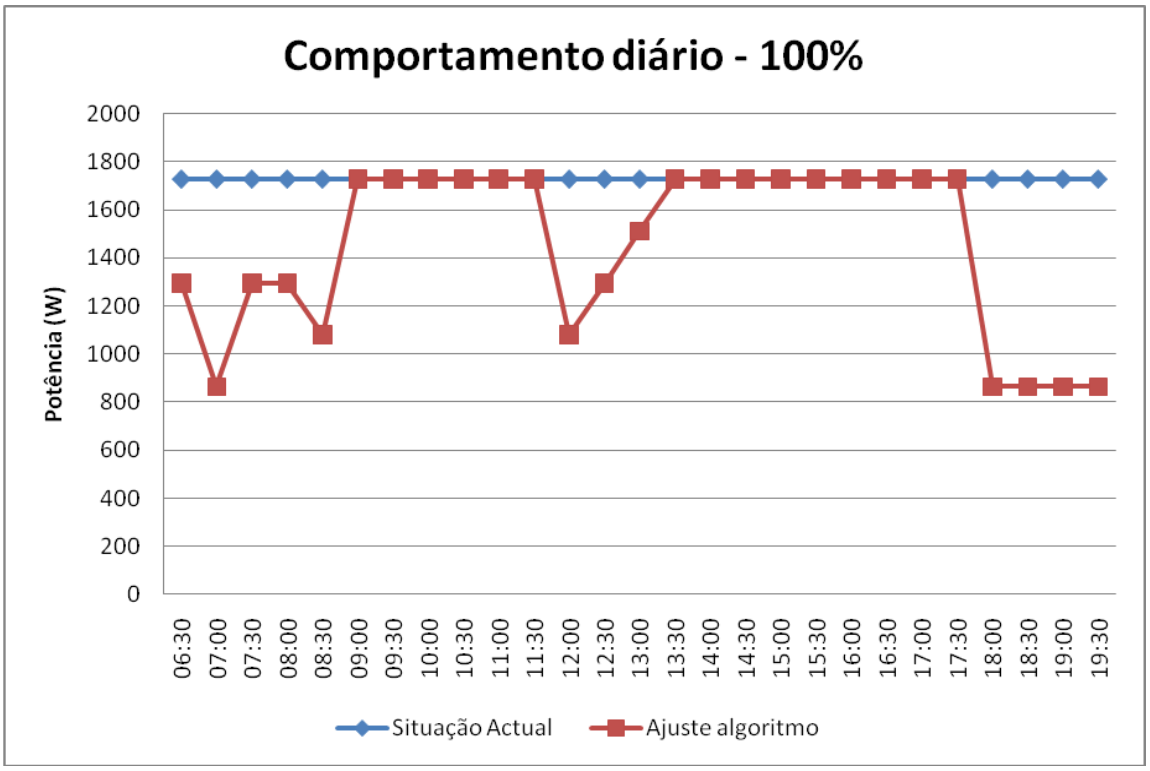


Figura 4.4 - Gráfico comparativo entre a situação actual e uma simulação do algoritmo em funcionamento com intensidade luminosa a 100% (241,2/228,3 lux)

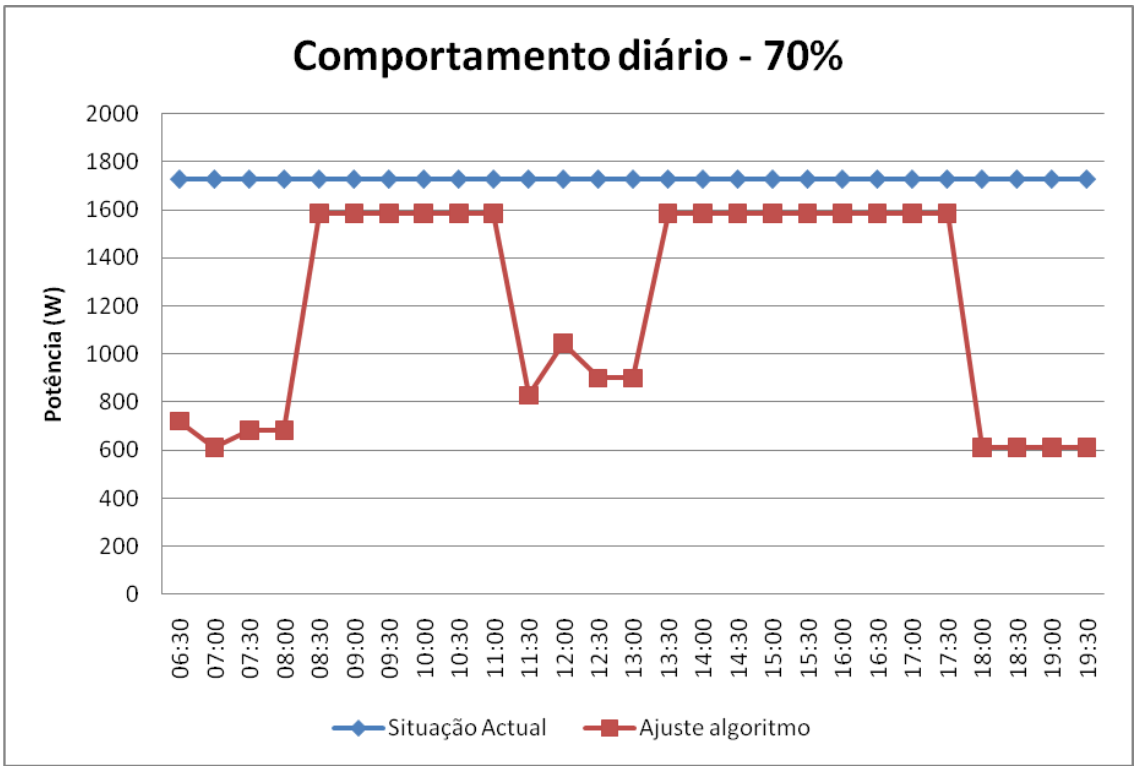


Figura 4.5 - Gráfico comparativo entre a situação actual e uma simulação do algoritmo em funcionamento com intensidade luminosa a 70% (168.84/159.81 lux)

Para a criação dos gráficos, foi multiplicado o número de lâmpadas ligadas pela potência de cada uma. Através do gráfico observa-se que nos horários em que eventualmente pode haver menor ocupação das mesas, ocorre a maior poupança de energia; não obstante, no cenário de ocupação total (9h -13h:30 e 14h – 17:30), para uma intensidade luminosa de 70%, também se observa uma poupança de energia, não muito significativa (menos 4 lâmpadas ligadas), em relação ao cenário actual.

Observa-se também que, em certas situações para um mesmo número de mesas ocupadas o consumo energético varia. Isso deve-se ao facto da proximidade entre os utilizadores, que reduz o número de lâmpadas acesas, como vai poder ser visto num teste posterior.

4.3.2. Variação das ocupações das mesas

Neste conjunto de testes será alterada, em diferentes cenários, a percentagem de ocupação do espaço (número de mesas ocupadas em relação ao total. Irão ser feitos testes de presenças para intensidade de 80%, e 50% de intensidade. O cenário de intensidade luminosa máxima foi excluído pois é o cenário onde todas as contribuições da matriz de influência são contabilizadas, incluindo as quase desprezáveis, pelo que se optou por reduzir para 80%, pois obtinha-se uma considerável melhoria da eficiência energética.

4.3.2.1. Cenário 1

Intensidade luminosa: **80% (192.96/182.64 lux)**
Mesas Ocupadas: **3**

1ºcaso -> Vector presenças = [11100000]:

```
Data: 2010-01-30 00:08:43.03

1 1 1 1 1 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 1 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:586.5
Lâmpadas ligadas: 19
```

2ºcaso -> Vector presenças = [10010010]:

```
Data: 2010-01-30 00:09:04.548

1 1 1 1 1 0
1 0 1 0 0 0
0 1 0 0 0 0
1 1 1 1 1 1
0 0 1 0 1 0
0 1 0 0 0 0
1 1 1 1 1 1
0 0 1 0 1 0
Total lux:586.5
Lâmpadas ligadas: 25
```

Através dos resultados obtidos, pode-se verificar que para um mesmo valor de ocupação (mesmo número de mesas), o número de lâmpadas ligadas pode ser diferente, obtendo-se eficiências energéticas diferentes. Isto acontece porque se estiverem 3 utilizadores em mesas adjacentes, as matrizes de influência dos sensores das mesas interceptam-se, não sendo necessário ligar uma nova lâmpada, recorrendo-se à do "vizinho". Neste caso poupam-se 6 lâmpadas.

4.3.2.2. Cenário 2

Intensidade luminosa: **80%(192.96/182.64 lux)**

Mesas ocupadas: **5**

1ºcaso -> Vector presenças = [11111000]

```
Data: 2010-01-30 00:12:40.423

1 1 1 1 1 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 1 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:982.9000000000001
Lâmpadas ligadas: 31
```

2ºcaso -> Vector presenças = [10110101]

```
Data: 2010-01-30 00:13:09.43

1 1 1 1 1 0
1 1 1 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 1 1 0 1 0
1 1 1 1 1 1
1 0 1 0 1 0
1 1 1 1 1 0
Total lux:974.8
Lâmpadas ligadas: 37
```

Neste caso também se pode verificar que com a proximidade dos ocupantes das mesas, menor será o número de lâmpadas a ligar. Outro factor interessante é que neste caso, para um menor número de lâmpadas ligadas obtém-se um valor menor de intensidade luminosa total. Esse total é a soma das contribuições das matrizes, o que apenas indica que no segundo caso foram necessárias mais lâmpadas com contribuição baixa nos sensores.

4.3.2.3. Cenário 3

Intensidade luminosa: **80%(192.96/182.64 lux)**

Mesas Ocupadas: **8**

1ºcaso -> Vector presenças = [11111111]

```
Data: 2010-01-30 00:16:07.031

1 1 1 1 1 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 0
Total lux:1569.4000000000003
Lâmpadas ligadas: 46
```

Neste cenário já se aproxima do cenário extremo, em que a ocupação é máxima e o valor de intensidade também próximo do máximo possível. Apenas são poupadas 2 lâmpadas.

4.3.2.4. Cenário 4

Intensidade luminosa: **40%(96.48/91.32 lux)**

Mesas Ocupadas: **2**

1ºcaso -> Vector presenças = [11000000]

```
Data: 2010-01-28 02:37:46.125

1 1 1 0 0 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:215.79999999999998
Lâmpadas ligadas: 6
```

2ºcaso -> Vector presenças = [01000100]

```
Data: 2010-01-28 02:37:56.004

0 0 0 0 0 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:207.2
Lâmpadas ligadas: 6
```

Neste cenário já não se aplica a teoria da proximidade, pois como as matrizes já não se interceptam, o número de lâmpadas a ligar é o mesmo, como se pode observar em ambos os casos.

4.3.2.5. Cenário 5

Intensidade luminosa: **40%(96.48/91.32 lux)**

Mesas Ocupadas: **5**

1ºcaso -> Vector presenças = [11111000]

```
Data: 2010-01-28 02:38:19.358

1 1 1 0 0 0
0 1 1 1 0 0
0 1 1 1 0 0
0 1 1 1 0 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:526.6
```

```
Lâmpadas ligadas: 15
```

2ºcaso -> Vector presenças = [10110101]

```
Data: 2010-01-28 02:38:34.681

1 1 1 0 0 0
0 0 0 0 0 0
0 1 1 1 0 0
0 1 1 1 0 0
0 0 0 0 0 0
0 1 1 1 0 0
0 0 0 0 0 0
1 1 1 0 0 0
Total lux:535.1999999999999
Lâmpadas ligadas: 15
```

Mais um cenário que tanto distribuindo as presenças, como juntando em mesas adjacentes, o número de lâmpadas a ligar é o mesmo.

4.3.2.6. Cenário 6

Intensidade luminosa: **40%(96.48/91.32 lux)**

Mesas Ocupadas: **8**

1ºcaso -> Vector presenças = [11111111]

```
Data: 2010-01-28 02:39:06.371

1 1 1 0 0 0
0 1 1 1 0 0
0 1 1 1 0 0
0 1 1 1 0 0
0 1 1 1 0 0
0 1 1 1 0 0
0 1 1 1 0 0
1 1 1 0 0 0
Total lux:846.0
Lâmpadas ligadas: 24
```

Neste cenário comprova-se que para um cenário onde não seja necessário um valor de intensidade alto nas mesas, podem ser poupadas 24 lâmpadas (864 W) relativamente à situação actual.

4.3.3. Variação da percentagem de intensidade luminosa

Neste conjunto de testes, é variada a percentagem de intensidade luminosa pretendida pelo administrador, para um valor de ocupação fixo.

4.3.3.1. Cenário 1

Mesas Ocupadas: **6**

Vector de presenças: **[10110111]**

1ºcaso -> Intensidade luminosa: 80% (192.96/182.64 lux)

```
Data: 2010-01-30 00:23:43.868

1 1 1 1 1 0
1 1 1 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 1 1 0 1 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 0
Total lux:1173.0000000000002
Lâmpadas ligadas: 40
```

2ºcaso -> Intensidade luminosa: 60% (144.72/136.98 lux)

```
Data: 2010-01-30 00:24:07.861

1 1 1 1 0 0
0 0 0 0 0 0
1 1 1 1 1 0
1 1 1 1 1 0
0 0 0 0 0 0
1 1 1 1 1 0
1 1 1 1 1 0
1 1 1 1 0 0
Total lux:881.6
Lâmpadas ligadas: 28
```

3ºcaso -> Intensidade luminosa: 30% (72.76/68.49 lux)

```
Data: 2010-01-30 00:24:42.806

1 1 0 0 0 0
0 0 0 0 0 0
0 0 1 1 0 0
0 0 1 1 0 0
0 0 0 0 0 0
0 0 1 1 0 0
0 0 1 1 0 0
1 1 0 0 0 0
Total lux:467.20000000000005
Lâmpadas ligadas: 12
```

Para este cenário, constata-se que à medida que a intensidade luminosa vai baixando, a poupança energética será maior, como seria de esperar.

4.3.3.2. Cenário 2

Mesas Ocupadas: **8**

Vector de presenças: **[11111111]**

1ºcaso -> Intensidade luminosa: 80% (192.96/182.64 lux)

```
Data: 2010-01-30 00:26:30.005

1 1 1 1 1 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 0
Total lux:1569.4000000000003
Lâmpadas ligadas: 46
```

2ºcaso -> Intensidade luminosa: 60% (144.72/136.98 lux)

```
Data: 2010-01-30 00:27:31.973

1 1 1 1 0 0
1 1 1 1 1 0
1 1 1 1 1 0
1 1 1 1 1 0
1 1 1 1 1 0
1 1 1 1 1 0
1 1 1 1 1 0
1 1 1 1 0 0
Total lux:1180.2000000000003
Lâmpadas ligadas: 38
```

3ºcaso -> Intensidade luminosa: 30% (72.76/68.49 lux)

```
Data: 2010-01-30 00:28:37.332

1 1 0 0 0 0
0 0 1 1 0 0
0 0 1 1 0 0
0 0 1 1 0 0
0 0 1 1 0 0
0 0 1 1 0 0
0 0 1 1 0 0
1 1 0 0 0 0
Total lux:620.4000000000001
Lâmpadas ligadas: 16
```

Neste caso, como as mesas estão completamente ocupadas, o algoritmo para percentagens de intensidade luminosa alta não permite poupar muitas lâmpadas: pelo contrário, para percentagens baixas já são poupadas bastantes.

4.3.4. Aplicação da matriz de obrigatoriedade

Esta matriz foi pensada para os casos em que, por determinação do administrador, uma ou mais lâmpadas têm de estar ligadas. Antes de decidir quais lâmpadas a ligar, o algoritmo terá de ter em conta as lâmpadas que têm de estar obrigatoriamente ligadas, e se for o caso, incluir a contribuição dessa mesma lâmpada no cálculo da matriz de saída final. De forma a comprovar isso, foram feitos os seguintes testes:

4.3.4.1. Cenário 1

Intensidade luminosa: **80%(192.96/182.64 lux)**
Vector de presenças: **[10000000]**

1ºcaso -> Matriz de Obrigatoriedade obriga a ligar as 6 lâmpadas da mesa 1

```
Data: 2010-01-30 00:39:11.645

1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:187.7
Lâmpadas ligadas: 7
```

2ºcaso -> Ajuste feito pelo algoritmo

```
Data: 2010-01-30 00:39:40.172

1 1 1 1 1 0
1 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:190.1
Lâmpadas ligadas: 7
```

Neste cenário como se pode ver pelos resultados obtidos, em ambos os casos se obtém o valor de luminosidade pretendido (80% = 182.64 lux). No entanto, no primeiro caso o algoritmo verifica que o total de luminosidade (187.7) é suficiente para satisfazer a condição e não utiliza a matriz de influências para ajustar; pelo contrário, no 2º caso, o algoritmo actua normalmente pois não foi introduzida nenhuma restrição pela matriz de obrigatoriedade e consegue mais intensidade com o mesmo número de lâmpadas.

4.3.4.2. Cenário 2

Intensidade luminosa: **80% (192.96/182.64 lux)**

Vector de presença: **[00100000]**

1ºcaso -> Matriz de Obrigatoriedade obriga a ligar as 6 lâmpadas da mesa 2

```
Data: 2010-01-30 00:45:49.284

0 0 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:186.1
Lâmpadas ligadas: 11
```

2ºcaso -> Ajuste feito pelo algoritmo

```
Data: 2010-01-30 00:45:53.213

0 0 0 0 0 0
0 0 0 0 0 0
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:179.79999999999998
Lâmpadas ligadas: 7
```

Neste cenário, o objectivo era provar a inteligência do algoritmo, pois no primeiro caso as 6 lâmpadas da mesa adjacente onde se encontra o utilizador são aproveitadas através da sua contribuição no sensor da mesa 3, deixando de ser necessário ligar mais uma lâmpada na mesa 3 e outra na mesa 4, pois a intensidade luminosa pretendida já foi atingida. Em comparação com o ajuste sem restrições pelo algoritmo, são poupadas 2 lâmpadas aproveitando a luminosidade da mesa ao lado.

4.3.5. Considerações finais

Depois de testados os diversos cenários, pode-se concluir que o algoritmo é eficiente, como foi visto no primeiro teste, pois reagiu às alterações do espaço convenientemente,

melhorando a eficiência energética do espaço. Observa-se a importância da matriz de influências e a posição dos sensores na escolha das lâmpadas a ligar. Por exemplo, nas mesas encostadas à parede, visto que a posição do sensor na mesa não é exactamente no meio, o algoritmo ao contrário das restantes mesas, começa por ligar as lâmpadas de uma das pontas. Isto porque foi definido na matriz de influências (Figura 3.6) que essas lâmpadas possuíam maior contribuição. Assim também se explica que quando a intensidade e a ocupação está a 80%, são ligadas todas as lâmpadas excepto uma em cada ponta, pois o algoritmo considera que para alcançar o valor pretendido existem lâmpadas com maior contribuição.

Nos restantes cenários, ficou provado que o algoritmo responde às variações nos parâmetros introduzidos, nomeadamente a intensidade de luz e as presenças nas mesas. Para o caso da matriz de obrigatoriedade, verifica-se a inteligência do algoritmo ao aproveitar as lâmpadas da mesa vizinha para não ligar lâmpadas adicionais.

Também se concluí que para cenários de intensidade luminosa máxima e ocupação alta, a eficiência energética é baixa, pois são aproveitadas lâmpadas da matriz de influências com contribuições muito baixas.

Capítulo 5

Conclusões e trabalho futuro

Neste capítulo, serão apresentadas as conclusões relativas ao trabalho desenvolvido face aos objectivos propostos, bem como propostas para possíveis melhoramentos futuros do sistema.

5.1. Conclusões

Foi desenvolvido um sistema inteligente de iluminação, recorrendo a uma rede de sensores sem fios e identificado os factores essenciais na criação desse mesmo sistema. Foi demonstrado que o algoritmo promove a poupança de energia, pois apenas liga as lâmpadas necessárias para uma determinada percentagem de luz e ocupação do espaço. Concluí-se que o sistema inteligente melhora a eficiência no cenário de teste - *open space* do 3º piso do INESC Porto, comparativamente com o estado actual e apresenta boas perspectivas de funcionamento noutros cenários.

Concluí-se que os factores essenciais na construção de um sistema inteligente são: a elaboração da calibração do sensor, pois é através da mesma que é feita a conversão dos valores dos sensores para lux, onde facilmente se cometem erros, pois a curva característica do sensor é criada com base numa aproximação a rectas. Outro factor essencial é a elaboração das matrizes de influência, pois é através das mesmas que o algoritmo toma decisões de qual lâmpada a ligar, sendo a base do sistema.

Ao desenvolver este sistema, para além de se poupar energia, também se estende o tempo de vida das lâmpadas.

5.2. Trabalho futuro

Face ao estado actual de desenvolvimento do sistema e aos resultados obtidos, existem vários melhoramentos que poderão ser introduzidos em trabalhos futuros. Em primeiro lugar, a implementação da interface responsável pelo controlo das lâmpadas, pois assim seria possível observar o sistema completamente implementado e efectuar testes mais rigorosos em condições reais. Implementado o sistema de comutação das lâmpadas e os sensores de presença, poderia também ser efectuado outro tipo de testes, nomeadamente o grau de satisfação do utilizador com a iluminação, revelando assim melhor como resolver o compromisso entre a satisfação do utilizador e a poupança energética.

Outro factor a considerar seria a introdução da luz solar no sistema, e verificar a influência que pode ter ao longo de um dia: considerando a sua contribuição no sistema, poderia ser poupada bastante energia em certas horas do dia, desligando algumas lâmpadas e aproveitando a luz incidente do exterior. Para além disso, como o *open space* do INESC possui estores, poderia ser desenvolvido um sistema de comutação dos estores nas horas de maior exposição solar.

Na interface gráfica, poderia ser introduzido um editor de zona de testes, não limitando o sistema ao perfil do INESC, tornando-o mais versátil. Esse editor daria ao administrador a possibilidade de introduzir o número e tipo de lâmpadas, tal como a sua posição.

Outro melhoramento possível para tornar o sistema versátil, seria explorar o modelo cliente-servidor numa configuração em rede, em que cada utilizador teria a sua conta e o seu perfil de iluminação que poderia ajustar dinamicamente com base nas tarefas que está a executar (e.g. ler artigos, programar ou outras tarefas, que exigem mais ou menos intensidade luminosa).

Anexo A

Resultados

Neste anexo será apresentado o resultado da simulação do comportamento diário em horário de trabalho do algoritmo.

A1 – Teste Intensidade luminosa – 100 %

Data: 2010-01-29 23:09:06.989

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:698.4
Lâmpadas ligadas: 36

Data: 2010-01-29 23:09:36.99

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:710.7
Lâmpadas ligadas: 24

Data: 2010-01-29 23:10:06.99

1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
Total lux:710.7
Lâmpadas ligadas: 36

Data: 2010-01-29 23:10:36.991

1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
Total lux:710.7
Lâmpadas ligadas: 36

Data: 2010-01-29 23:11:06.996

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:710.7
Lâmpadas ligadas: 30

Data: 2010-01-29 23:11:36.994

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:12:06.994

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:12:36.993

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:13:06.994

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:13:36.995

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:14:06.996

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:14:36.996

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:951.7000000000002
Lâmpadas ligadas: 30

Data: 2010-01-29 23:15:06.997

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:1192.7000000000003
Lâmpadas ligadas: 36

Data: 2010-01-29 23:15:36.998

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:939.4
Lâmpadas ligadas: 42

Data: 2010-01-29 23:16:06.998

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:939.4
Lâmpadas ligadas: 48

Data: 2010-01-29 23:16:36.999

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:17:07.0

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:17:37.001

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:18:07.001

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:18:37.002

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:19:07.003

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:19:37.005

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:20:07.005

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
Total lux:1903.4000000000005
Lâmpadas ligadas: 48

Data: 2010-01-29 23:20:37.005

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:710.7
Lâmpadas ligadas: 24

Data: 2010-01-29 23:21:07.006

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:710.7
Lâmpadas ligadas: 24

Data: 2010-01-29 23:21:37.006

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:710.7
Lâmpadas ligadas: 24

Data: 2010-01-29 23:22:07.007

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:710.7
Lâmpadas ligadas: 24

A2 – Teste intensidade luminosa a 70%

Data: 2010-01-29 23:26:06.269

1 1 1 1 0 0
1 0 1 0 0 0
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
1 1 1 1 1 1
0 0 1 0 0 0
Total lux:535.7
Lâmpadas ligadas: 20

Data: 2010-01-29 23:26:36.269

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:535.7
Lâmpadas ligadas: 17

Data: 2010-01-29 23:27:06.269

1 1 1 1 0 0
1 0 1 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:535.7
Lâmpadas ligadas: 19

Data: 2010-01-29 23:27:36.274

0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
Total lux:539.4
Lâmpadas ligadas: 19

Data: 2010-01-29 23:28:06.276

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:28:36.272

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:29:06.272

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:29:36.273

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:30:06.273

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:30:36.274

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:31:06.275

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:715.5000000000001
Lâmpadas ligadas: 23

Data: 2010-01-29 23:31:36.276

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:895.3000000000002
Lâmpadas ligadas: 29

Data: 2010-01-29 23:32:06.276

0 0 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:719.2
Lâmpadas ligadas: 25

Data: 2010-01-29 23:32:36.277

0 0 0 0 0 0
0 0 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
1 0 1 0 0 0
1 1 1 1 0 0
Total lux:715.4999999999999
Lâmpadas ligadas: 25

Data: 2010-01-29 23:33:06.278

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:33:36.278

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:34:06.279

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002

Lâmpadas ligadas: 44

Data: 2010-01-29 23:34:36.28

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:35:06.282

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:35:36.282

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:36:06.282

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:36:36.285

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 0 0
Total lux:1431.0000000000002
Lâmpadas ligadas: 44

Data: 2010-01-29 23:37:06.284

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
Total lux:535.7
Lâmpadas ligadas: 17

Data: 2010-01-29 23:37:36.285

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:535.7
Lâmpadas ligadas: 17

Data: 2010-01-29 23:38:06.286

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
Total lux:535.7
Lâmpadas ligadas: 17

Data: 2010-01-29 23:38:36.287

1 1 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

0 0 0 0 0 0
0 0 0 0 0 0
Total lux:535.7
Lâmpadas ligadas: 17

Anexo B

Código Desenvolvido

Neste anexo irá ser apresentado o código JAVA desenvolvido para o desenvolvimento do algoritmo de decisão e a comunicação entre o JAVA e os motes, respectivamente Algoritmo.java e CommInterface.java. Também é apresentado o código para os motes, OscilloscopeRF e TOSbase.Só apresento este extracto de código, pois penso ser o mais relevante.

B1 – Algoritmo.java

```
package net.tinyos.GUI;

import java.util.TimerTask;
import java.io.*;
import java.sql.Timestamp;
import java.lang.*;

/**
 *
 * @author Bruno Ferreira
 */
public class Algoritmo extends TimerTask  {

    public static int contador = 0,LAMPADAS = 6, COLUNAS = 8, i=0, j=0, z=0, k =
0,col=0,lamp=0,lixo;
    public double totallux=0,target=0;
    public double indicescolunas[] = new double[18];
    public double indiceslampadas[] = new double[18];
    public static double[][] matrizInfluencia1 = new double[COLUNAS][LAMPADAS];
    public double[] vectorInfluencia1 = {39.4, 41, 31.8, 30, 14, 14,16.4, 6.1, 17.5, 5.4,
9.7, 3.4,0.5, 0.1, 0.5, 0.1, 0.5, 0.1};
    public static double[][] matrizInfluencia2 = new double[COLUNAS][LAMPADAS];
    public static double[][] matrizInfluencia3 = new double[COLUNAS][LAMPADAS];
```

```
        public static double[][] matrizInfluencia4 = new double[COLUNAS][LAMPADAS];
        public static double[][] matrizInfluencia5 = new double[COLUNAS][LAMPADAS];
        public static double[][] matrizInfluencia6 = new double[COLUNAS][LAMPADAS];
        public static double[][] matrizInfluencia7 = new double[COLUNAS][LAMPADAS];
        public static double[][] matrizInfluencia8 = new double[COLUNAS][LAMPADAS];
        public double[] vectorInfluencia2 ={4.2, 9.5, 4.1, 7.8, 3.7, 7.5 ,25.7, 27.0, 39.5, 37.1,
20, 19.3,6.5, 2.3, 11.2, 3.7, 8.9, 3.0};
        public double[] teste = new double[18];
        public static int[][] matrizsaida = new int[COLUNAS][LAMPADAS];
        public static int[][] matrizsaidabackup = new int[COLUNAS][LAMPADAS];
        private String lol;
        public static int[] presenca={0,0,0,0,0,0,0,0};

        public double[] lumino= new double[8];



        public void run() {


           limparMatriz(matrizsaida,COLUNAS,LAMPADAS);

           k=0;
           totallux=0;
           contador=0;

           for (i = 0; i < COLUNAS; i++)
                    for (j = 0; j < LAMPADAS; j++) {
                    if(i==0 || i==1 || i==2){
                    matrizInfluencia1[i][j]=vectorInfluencia1[k];
                    k++;
                    }
                    else
                       matrizInfluencia1[i][j]=0;
                    }
           k=0;
           for (i = 0; i < COLUNAS; i++)
                    for (j = 0; j < LAMPADAS; j++) {
                       if(i==0 || i==1 || i==2){
                    matrizInfluencia2[i][j]=vectorInfluencia2[k];
                    k++;
                       }
                       else
                          matrizInfluencia1[i][j]=0;
                    }
           k=0;

           for (i = 0; i < COLUNAS; i++)
              for (j = 0; j < LAMPADAS; j++) {
              if(i==1 || i==2 || i==3){
              matrizInfluencia3[i][j]=vectorInfluencia2[k];
              k++;
              }
              else
                matrizInfluencia3[i][j]=0;
              }
           k=0;
           for (i = 0; i < COLUNAS; i++)
              for (j = 0; j <LAMPADAS; j++) {
              if(i==2 || i==3 || i==4){
```

```
                matrizInfluencia4[i][j]=vectorInfluencia2[k];
                k++;
                }
                else
                   matrizInfluencia4[i][j]=0;
                }
             k=0;
             for (i = 0; i < COLUNAS; i++)
                for (j = 0; j < LAMPADAS; j++) {
                if(i==3 || i==4 || i==5){
                matrizInfluencia5[i][j]=vectorInfluencia2[k];
                k++;
                }
                else
                   matrizInfluencia5[i][j]=0;
                }
             k=0;
             for (i = 0; i < COLUNAS; i++)
                for (j = 0; j < LAMPADAS; j++) {
                if(i==4 || i==5 || i==6){
                matrizInfluencia6[i][j]=vectorInfluencia2[k];
                k++;
                }
                else
                   matrizInfluencia6[i][j]=0;
                }
             k=0;
             for (i = 0; i < COLUNAS; i++)
                for (j = 0; j < LAMPADAS; j++) {
                if(i==5 || i==6 || i==7){
                matrizInfluencia7[i][j]=vectorInfluencia2[k];
                k++;
                }
                else
                   matrizInfluencia7[i][j]=0;
                }
             k=0;
             for (i = 0; i < COLUNAS; i++)
                for (j = 0; j < LAMPADAS; j++) {
                if(i==5 || i==6 || i==7){
                matrizInfluencia8[i][j]=vectorInfluencia1[k];
                k++;
                }
                else
                   matrizInfluencia8[i][j]=0;
                }
             k=0;




             net.tinyos.GUI.MenuInicial.toolkit.beep();


             if(presenca[0]==1)
                if (lumino[0] >(net.tinyos.GUI.MenuInicial.target1 +10) || lumino[0]
<(net.tinyos.GUI.MenuInicial.target1 -10))
                   algoritmo(net.tinyos.GUI.MenuInicial.target1,matrizInfluencia1,1);
                else
                   copiarMatriz(matrizsaidabackup,matrizsaida);
```

```
          if(presenca[1]==1)
              if (lumino[1] >(net.tinyos.GUI.MenuInicial.target2 +10) || lumino[1]
<(net.tinyos.GUI.MenuInicial.target2 -10))
                  algoritmo(net.tinyos.GUI.MenuInicial.target2,matrizInfluencia2,2);
              else
                  copiarMatriz(matrizsaidabackup,matrizsaida);
          if(presenca[2]==1)
              if (lumino[2] >(net.tinyos.GUI.MenuInicial.target2 +10) || lumino[2]
<(net.tinyos.GUI.MenuInicial.target2 -10))
                  algoritmo(net.tinyos.GUI.MenuInicial.target2,matrizInfluencia3,3);
              else
                  copiarMatriz(matrizsaidabackup,matrizsaida);
           if(presenca[3]==1)
              if (lumino[3] >(net.tinyos.GUI.MenuInicial.target2 +10) || lumino[3]
<(net.tinyos.GUI.MenuInicial.target2 -10))
                  algoritmo(net.tinyos.GUI.MenuInicial.target2,matrizInfluencia4,4);
              else
                  copiarMatriz(matrizsaidabackup,matrizsaida);
           if(presenca[4]==1)
              if (lumino[4] >(net.tinyos.GUI.MenuInicial.target2 +10) || lumino[4]
<(net.tinyos.GUI.MenuInicial.target2 -10))
                  algoritmo(net.tinyos.GUI.MenuInicial.target2,matrizInfluencia5,5);
              else
                  copiarMatriz(matrizsaidabackup,matrizsaida);
           if(presenca[5]==1)
              if (lumino[5] >(net.tinyos.GUI.MenuInicial.target2 +10) || lumino[5]
<(net.tinyos.GUI.MenuInicial.target2 -10))
                  algoritmo(net.tinyos.GUI.MenuInicial.target2,matrizInfluencia6,6);
              else
                  copiarMatriz(matrizsaidabackup,matrizsaida);
           if(presenca[6]==1)
              if (lumino[6] >(net.tinyos.GUI.MenuInicial.target2 +10) || lumino[6]
<(net.tinyos.GUI.MenuInicial.target2 -10))
                  algoritmo(net.tinyos.GUI.MenuInicial.target2,matrizInfluencia7,7);
              else
                  copiarMatriz(matrizsaidabackup,matrizsaida);
           if(presenca[7]==1)
              if (lumino[7] >(net.tinyos.GUI.MenuInicial.target1 +10) || lumino[7]
<(net.tinyos.GUI.MenuInicial.target1 -10))
                  algoritmo(net.tinyos.GUI.MenuInicial.target1,matrizInfluencia8,8);
              else
                  copiarMatriz(matrizsaidabackup,matrizsaida);

          contadorDeLampadas(matrizsaida);
          escreverFicheiro(matrizsaida,"resultados.txt",totallux,contador);
          imprimirmatriz(matrizsaida);
          copiarMatriz(matrizsaida,matrizsaidabackup);

      }

      public void algoritmo(double targetx, double[][] matrizInfluencia,int poslampadas) {
          System.out.println("Algoritmo em execução!");
          ordenarInfluencias(matrizInfluencia, targetx, poslampadas);

      }

      public static void bubbleSort(double[] a, double[] b, double[] c) {
          for (i = 0; i < a.length - 1; i++) {
              for (j = 0; j < a.length - 1; j++) {
```

```
                if (a[j] < a[j + 1]) {
                    swap(a, j, j + 1);
                    swap(b, j, j + 1);
                    swap(c, j, j + 1);
                }
            }
        }

    }

    private static void swap(double[] a, int i, int j) {
        double temp = a[i];
        a[i] = a[j];
        a[j] = temp;
    }

    public void imprimirmatriz(int[][] a) {
        System.out.println();
        for (i = 0; i < COLUNAS; i++) {
            for (j = 0; j < LAMPADAS; j++) {
                System.out.print(a[i][j]);
            }
            System.out.println();
        }
    }

    public void ordenarInfluencias(double[][] matrizInfluencia, double target, int
poslampadas) {
        int y=0;
                for (i = 0; i < COLUNAS; i++) {
                    for (j = 0; j < LAMPADAS; j++) {
                        target = target - (matrizInfluencia[i][j] *
net.tinyos.GUI.MenuInicial.matrizOb[i][j]);
                        totallux=totallux+(matrizInfluencia[i][j]*
net.tinyos.GUI.MenuInicial.matrizOb[i][j]);
                        if (target <= 0) {
                            matrizsaida[i][j] = 1;
                            return;
                        } else if (net.tinyos.GUI.MenuInicial.matrizOb[i][j] == 1) {
                            matrizsaida[i][j] = 1;
                        }
                    }
                }

        for (i = 0; i < COLUNAS; i++) {
            for (j = 0; j < LAMPADAS; j++) {
                if(net.tinyos.GUI.MenuInicial.matrizOb[i][j]==1) // EXCLUIR LUZES LIGADAS DA
SUBTRACCAO POSTERIOR
                    lixo=0;
                else
                    if(matrizInfluencia[i][j]!=0){
                    teste[y] = matrizInfluencia[i][j];
                    indicescolunas[y] = i;
                    indiceslampadas[y] = j;
                    y++;
                    }
            }
        }

        bubbleSort(teste, indicescolunas, indiceslampadas);
```

```
            for (z = 0; z < 18; z++) {
               target = target - teste[z];

               col = (int) indicescolunas[z];
               lamp = (int) indiceslampadas[z];


               totallux=totallux+teste[z];

               if (target <= 0) {

                  if(poslampadas==8){
                            if(col==5){
                               matrizsaida[col+2][lamp] = 1;
                            }
                            else if(col==7){
                               matrizsaida[col-2][lamp] = 1;
                            }
                           else
                           matrizsaida[col][lamp] = 1;
                  }
                  else
                     matrizsaida[col][lamp] = 1;
                  return;
               }
                else {
                   if(poslampadas==8){
                            if(col==5){
                               matrizsaida[col+2][lamp] = 1;
                            }
                            else if(col==7){
                               matrizsaida[col-2][lamp] = 1;
                            }
                            else
                            matrizsaida[col][lamp] = 1;
                  }
                   else
                     matrizsaida[col][lamp] = 1;

               }
            }

         }

         public void escreverFicheiro(int[][] matrizsaida,String nomefich,double totallux,int
contador) {

            File f;
            Timestamp time = new Timestamp(System.currentTimeMillis());
            lol = time.toString();

            try {
               f = new File(nomefich);
               FileWriter fstream = new FileWriter(f,true);
               BufferedWriter out = new BufferedWriter(fstream);


               out.write("Data:" + " " + lol);
               out.newLine();
```

```
            out.newLine();
            for (i = 0; i < COLUNAS; i++) {
                for (j = 0; j < LAMPADAS; j++) {
                    out.write(matrizsaida[i][j] + " ");
                    if (j == 5) {
                      out.newLine();
                    }
                }
            }
            out.write("Total lux:"+Double.toString(totallux));
            out.newLine();
            out.write("Lâmpadas ligadas: "+Integer.toString(contador));
            out.newLine();
            out.newLine();
            //Close the output stream
            out.close();
        } catch (Exception e) {//Catch exception if any
            System.err.println("Error: " + e.getMessage());
        }


    }

    public static void limparMatriz(int[][] matrizsaida,int colunas,int lampadas){
        for (i = 0; i < colunas; i++) {
                for (j = 0; j < lampadas; j++)
                    matrizsaida[i][j]=0;

    }

}
    public void copiarMatriz(int[][] matriz1,int[][] matriz2)
    {
       for (i = 0; i < COLUNAS; i++) {
                for (j = 0; j < LAMPADAS; j++)
                    matriz2[i][j]=matriz1[i][j];
    }



    }

    public void contadorDeLampadas(int[][] matriz){
          for (i = 0; i < COLUNAS; i++) {
                for (j = 0; j < LAMPADAS; j++)
                    if(matriz[i][j]==1)
                        contador++;
    }
    }
}
```

B2 – CommInterface.java

```
package net.tinyos.bruno;

import java.io.*;
import net.tinyos.message.*;
import net.tinyos.GUI.*;
import net.tinyos.util.PrintStreamMessenger;
/**
 *
 * @author Bruno Ferreira
 */
public class CommInterface {

    public static int moteID,packetNum, channelID,i,media,copo;

    public static double lum1,lum2,lum3;

    public static int[] data,enviar;

    public static class MessageReceive implements MessageListener {

        public void messageReceived(int dest_addr, Message msg) {
            if (msg instanceof OscopeMsg) {
                oscopeReceived(dest_addr, (OscopeMsg) msg);
            } else {
                throw new RuntimeException("messageReceived: Got bad message type: " + msg);
            }
        }

        public static void oscopeReceived(int dest_addr, OscopeMsg omsg) {


            moteID = omsg.get_sourceMoteID();
            channelID = omsg.get_channel();
            packetNum = omsg.get_lastSampleNumber();
            data = omsg.get_data();

            copo=0;
            for(i=0;i<10;i++)
                copo+=data[i];

            media =copo/10;

            System.out.println("Valor de luminosidade:" + media);
        if (net.tinyos.bruno.CommInterface.moteID  == 1)
        {
            MenuInicial.TextMote1.setText(Integer.toString(media));
            lum1=MenuInicial.ADC_to_LUX(media);
            MenuInicial.TextMote2.setText(Double.toString(lum1));
        }
        if (net.tinyos.bruno.CommInterface.moteID  == 2)
        {
            MenuInicial.TextMote3.setText(Integer.toString(media));
            lum2=MenuInicial.ADC_to_LUX(media);
            MenuInicial.TextMote4.setText(Double.toString(lum2));
        }
        if (net.tinyos.bruno.CommInterface.moteID  == 3)
        {
```

```
        MenuInicial.TextMote5.setText(Integer.toString(media));
        lum3=MenuInicial.ADC_to_LUX(media);
        MenuInicial.TextMote6.setText(Double.toString(lum3));
      }

      }
    }

}
```

B3 - OscilloscopeRF

B3.1 – Oscilloscope.nc

```
configuration Oscilloscope { }
implementation
{
  components Main, OscilloscopeM
           , TimerC
           , LedsC
           , DemoSensorC as Sensor
           , GenericComm as Comm;

  Main.StdControl -> OscilloscopeM;
  Main.StdControl -> TimerC;

  OscilloscopeM.Timer -> TimerC.Timer[unique("Timer")];
  OscilloscopeM.Leds -> LedsC;
  OscilloscopeM.SensorControl -> Sensor;
  OscilloscopeM.ADC -> Sensor;
  OscilloscopeM.CommControl -> Comm;
  OscilloscopeM.DataMsg -> Comm.SendMsg[AM_OSCOPEMSG];
}
```

B3.2 . OscilloscopeM.nc

```
includes OscopeMsg;

module OscilloscopeM
{
  provides interface StdControl;
  uses {
    interface Timer;
    interface Leds;
    interface StdControl as SensorControl;
    interface ADC;
    interface StdControl as CommControl;
    interface SendMsg as DataMsg;
    interface ReceiveMsg as ResetCounterMsg;
  }
```

```
}
implementation
{
  uint8_t packetReadingNumber;
  uint16_t readingNumber;
  TOS_Msg msg[2];
  uint8_t currentMsg;

  /**
   * Used to initialize this component.
   */
  command result_t StdControl.init() {
    call Leds.init();
    call Leds.yellowOff(); call Leds.redOff(); call
Leds.greenOff();

    //turn on the sensors so that they can be read.
    call SensorControl.init();

    call CommControl.init();

    atomic {
      currentMsg = 0;
      packetReadingNumber = 0;
      readingNumber = 0;
    }

    dbg(DBG_BOOT, "OSCOPE initialized\n");
    return SUCCESS;
  }

  /**
   * Starts the SensorControl and CommControl components.
   * @return Always returns SUCCESS.
   */
  command result_t StdControl.start() {
    call SensorControl.start();
    call Timer.start(TIMER_REPEAT, 125);
    call CommControl.start();
    return SUCCESS;
  }

  /**
   * Stops the SensorControl and CommControl components.
   * @return Always returns SUCCESS.
   */
  command result_t StdControl.stop() {
    call SensorControl.stop();
    call Timer.stop();
    call CommControl.stop();
    return SUCCESS;
  }

  task void dataTask() {
    struct OscopeMsg *pack;
    atomic {
```

```
      pack = (struct OscopeMsg *)msg[currentMsg].data;
      packetReadingNumber = 0;
      pack->lastSampleNumber = readingNumber;
    }

    pack->channel = 1;
    pack->sourceMoteID = TOS_LOCAL_ADDRESS;

    /* Try to send the packet. Note that this will return
     * failure immediately if the packet could not be queued for
     * transmission.
     */
    if (call DataMsg.send(TOS_BCAST_ADDR, sizeof(struct
OscopeMsg),
                          &msg[currentMsg]))
      {
      atomic {
        currentMsg ^= 0x1;
      }
      call Leds.yellowToggle();
      }
  }

  /**
   * Signalled when data is ready from the ADC. Stuffs the
sensor
   * reading into the current packet, and sends off the packet
when
   * BUFFER_SIZE readings have been taken.
   * @return Always returns SUCCESS.
   */
  async event result_t ADC.dataReady(uint16_t data) {
    struct OscopeMsg *pack;
    atomic {
      pack = (struct OscopeMsg *)msg[currentMsg].data;
      pack->data[packetReadingNumber++] = data;
      readingNumber++;
      dbg(DBG_USR1, "data_event\n");
      if (packetReadingNumber == BUFFER_SIZE) {
     post dataTask();
      }
    }
    if (data > 0x0300)
      call Leds.redOn();
    else
      call Leds.redOff();

    return SUCCESS;
  }

  /**
   * Signalled when the previous packet has been sent.
   * @return Always returns SUCCESS.
   */
  event result_t DataMsg.sendDone(TOS_MsgPtr sent, result_t
success) {
```

```
    return SUCCESS;
  }

  /**
   * Signalled when the clock ticks.
   * @return The result of calling ADC.getData().
   */
  event result_t Timer.fired() {
    return call ADC.getData();
  }
```

B3.3 - OscopeMsg.h

```
enum {
  BUFFER_SIZE = 10
};

struct OscopeMsg
{
    uint16_t sourceMoteID;
    uint16_t lastSampleNumber;
    uint16_t channel;
    uint16_t data[BUFFER_SIZE];
};

enum {
  AM_OSCOPEMSG = 10,
};
```

B4 - TOSBase

B4.1 - TOSBase.nc

```
configuration TOSBase {
}
implementation {
  components Main, TOSBaseM, RadioCRCPacket as Comm, FramerM,
UART, LedsC;

  Main.StdControl -> TOSBaseM;

  TOSBaseM.UARTControl -> FramerM;
  TOSBaseM.UARTSend -> FramerM;
  TOSBaseM.UARTReceive -> FramerM;
  TOSBaseM.UARTTokenReceive -> FramerM;

  TOSBaseM.RadioControl -> Comm;
  TOSBaseM.RadioSend -> Comm;
  TOSBaseM.RadioReceive -> Comm;

  TOSBaseM.Leds -> LedsC;
```

```
  FramerM.ByteControl -> UART;
  FramerM.ByteComm -> UART;
}
```

B4.2 - TOSBaseM.nc

```
#ifndef TOSBASE_BLINK_ON_DROP
#define TOSBASE_BLINK_ON_DROP
#endif

module TOSBaseM {
  provides interface StdControl;
  uses {
    interface StdControl as UARTControl;
    interface BareSendMsg as UARTSend;
    interface ReceiveMsg as UARTReceive;
    interface TokenReceiveMsg as UARTTokenReceive;

    interface StdControl as RadioControl;
    interface BareSendMsg as RadioSend;
    interface ReceiveMsg as RadioReceive;

    interface Leds;
  }
}

implementation
{
  enum {
    UART_QUEUE_LEN = 12,
    RADIO_QUEUE_LEN = 12,
  };

  TOS_Msg    uartQueueBufs[UART_QUEUE_LEN];
  TOS_MsgPtr uartQueue[UART_QUEUE_LEN];
  uint8_t    uartIn, uartOut;
  bool       uartBusy, uartFull;

  TOS_Msg    radioQueueBufs[RADIO_QUEUE_LEN];
  TOS_MsgPtr radioQueue[RADIO_QUEUE_LEN];
  uint8_t    radioIn, radioOut;
  bool       radioBusy, radioFull;

  task void UARTSendTask();
  task void RadioSendTask();

  void failBlink();
  void dropBlink();

  command result_t StdControl.init() {
    result_t ok1, ok2, ok3;
    uint8_t i;

    for (i = 0; i < UART_QUEUE_LEN; i++) {
      uartQueue[i] = &uartQueueBufs[i];
```

```
    }
    uartIn = uartOut = 0;
    uartBusy = FALSE;
    uartFull = FALSE;

    for (i = 0; i < RADIO_QUEUE_LEN; i++) {
      radioQueue[i] = &radioQueueBufs[i];
    }
    radioIn = radioOut = 0;
    radioBusy = FALSE;
    radioFull = FALSE;

    ok1 = call UARTControl.init();
    ok2 = call RadioControl.init();
    ok3 = call Leds.init();

    dbg(DBG_BOOT, "TOSBase initialized\n");

    return rcombine3(ok1, ok2, ok3);
  }

  command result_t StdControl.start() {
    result_t ok1, ok2;

    ok1 = call UARTControl.start();
    ok2 = call RadioControl.start();

    return rcombine(ok1, ok2);
  }

  command result_t StdControl.stop() {
    result_t ok1, ok2;

    ok1 = call UARTControl.stop();
    ok2 = call RadioControl.stop();

    return rcombine(ok1, ok2);
  }

  event TOS_MsgPtr RadioReceive.receive(TOS_MsgPtr Msg) {
    TOS_MsgPtr pBuf = Msg;

    dbg(DBG_USR1, "TOSBase received radio packet.\n");

    if ((!Msg->crc) || (Msg->group != TOS_AM_GROUP))
      return Msg;

    atomic {
      if (!uartFull) {
      pBuf = uartQueue[uartIn];
      uartQueue[uartIn] = Msg;

      if( ++uartIn >= UART_QUEUE_LEN ) uartIn = 0;

      if (uartIn == uartOut) {
        uartFull = TRUE;
```

```
      }

      if (!uartBusy) {
        if (post UARTSendTask()) {
          uartBusy = TRUE;
        }
      }
       } else {
      dropBlink();
       }
    }

    return pBuf;
  }

  task void UARTSendTask() {
    bool noWork = FALSE;

    dbg (DBG_USR1, "TOSBase forwarding Radio packet to UART\n");

    atomic {
      if (uartIn == uartOut && uartFull == FALSE) {
      uartBusy = FALSE;
      noWork = TRUE;
       }
    }
    if (noWork) {
      return;
    }

    if (call UARTSend.send(uartQueue[uartOut]) == SUCCESS) {
      call Leds.greenToggle();
    } else {
      failBlink();
      post UARTSendTask();
    }
  }

  event result_t UARTSend.sendDone(TOS_MsgPtr msg, result_t
success) {

    if (!success) {
      failBlink();
    } else {

      atomic {
      if (msg == uartQueue[uartOut]) {
        if( ++uartOut >= UART_QUEUE_LEN ) uartOut = 0;
        if (uartFull) {
                uartFull = FALSE;
        }
      }
       }
    }

    post UARTSendTask();
```

```
    return SUCCESS;
  }

  event TOS_MsgPtr UARTReceive.receive(TOS_MsgPtr Msg) {
    return Msg;
  }

  event TOS_MsgPtr UARTTokenReceive.receive(TOS_MsgPtr Msg,
uint8_t Token) {
    TOS_MsgPtr  pBuf = Msg;
    bool reflectToken = FALSE;

    dbg(DBG_USR1, "TOSBase received UART token packet.\n");

    atomic {
      if (!radioFull) {
      reflectToken = TRUE;
      pBuf = radioQueue[radioIn];
      radioQueue[radioIn] = Msg;
      if( ++radioIn >= RADIO_QUEUE_LEN ) radioIn = 0;
      if (radioIn == radioOut)
        radioFull = TRUE;

      if (!radioBusy) {
        if (post RadioSendTask()) {
          radioBusy = TRUE;
        }
      }
       } else {
      dropBlink();
       }
    }

    if (reflectToken) {
      call UARTTokenReceive.ReflectToken(Token);
    }

    return pBuf;
  }

  task void RadioSendTask() {

    bool noWork = FALSE;

    dbg (DBG_USR1, "TOSBase forwarding UART packet to Radio\n");

    atomic {
      if (radioIn == radioOut && radioFull == FALSE) {
      radioBusy = FALSE;
      noWork = TRUE;
       }
    }
    if (noWork)
      return;
```

```
    radioQueue[radioOut]->group = TOS_AM_GROUP;

    if (call RadioSend.send(radioQueue[radioOut]) == SUCCESS) {
      call Leds.redToggle();
    } else {
      failBlink();
      post RadioSendTask();
    }
  }

  event result_t RadioSend.sendDone(TOS_MsgPtr msg, result_t
success) {

    if (!success) {
      failBlink();
    } else {
      atomic {
      if (msg == radioQueue[radioOut]) {
        if( ++radioOut >= RADIO_QUEUE_LEN ) radioOut = 0;
        if (radioFull)
          radioFull = FALSE;
      }
       }
    }

    post RadioSendTask();
    return SUCCESS;
  }

  void dropBlink() {
#ifdef TOSBASE_BLINK_ON_DROP
    call Leds.yellowToggle();
#endif
  }

  void failBlink() {
#ifdef TOSBASE_BLINK_ON_FAIL
    call Leds.yellowToggle();
#endif
  }
}
```

Referências

[1] Artigo com dados da Frost & Sullivan, disponível em http://www.electricityforum.com/news/apr09/Globalelectricitydemandtoexplodeintwodecades.html , Acesso em Janeiro 2010.

[2] Artigo publicado por The NEED PROJECTO, disponível em http://www.need.org/needpdf/infobook_activities/SecInfo/ConsS.pdf, Acesso em Janeiro 2010.

[3] Control4, produtos de domótica, http://www.control4.com/home-control/green-living/, Acesso em Janeiro 2010.

[4] Mitsunori Miki, Emi Asayama, Tomoyuki Hiroyasu, "Intelligent Lighting System using Visible-Light Communication Technology", 2006.

[5] F.L. Lewis, "Wireless Sensor Networks", disponível em http://arri.uta.edu/acs/networks/WirelessSensorNetChap04.pdf, Acesso em Junho 2009.

[6] Jason Lester Hill,"*System Architecture for Wireless Sensor Networks*", Tese publicada na Universidade da California, 2003.

[7] Programa emulador de um sistema UNIX, disponível em http://www.cygwin.com, acesso em Janeiro 2010.

[8] Tutorial usado como referência para uso do *tinyos*, http://www.tinyos.net/tinyos-1.x/doc/tutorial/, Acesso em Janeiro 2010.

[9] Tech Note, "X-10 Communications Protocol and Power Line Interface PSC04 & PSC05", rev 2.4, disponível em http://www.x10pro.com/pro/pdf/technote.pdf, Acesso em Junho de 2009.

[10]Homeplug *whitepapers*, http://www.homeplug.org/tech/whitepapers/ , Acesso em Janeiro 2010.

[11]Patrick Kinney, *ZigBee Technology : Wireless Technology Controls that Simply Works,* 2003.

[12]Z-Wave, produtos e especificações, disponível em http://web1.zen-sys.com/modules/Products&Techonology/, Acesso Junho 2009.

[13]Meng-Shiuan Pan, Lun-Wu Yeh, Yen-Ann chen, Yu-Hsuan Lin, Yu-Chee Tseng, "A WSN-based Intelligent Light Control System Considering User Activities and Profiles", 2006.

[14]INSTEON produtos e especificações, disponível em http://www.insteon.net/, Acesso Junho 2009.

[15]SmartHome, produtos de Domótica, http://www.smarthome.com/_/index.aspx, Acesso em Junho 2009.

[16]Mitsunori Miki, Emi Asayama, Tomoyuki Hiroyasu, “Intelligent Lighting System using Visible-Light Communication Technology”, 2006.

[17]*CrossBow Technology Inc*, é uma empresa de fabrico de *hardware* para redes de sensores RF, disponível em http://www.xbow.com, acesso em Janeiro 2010.

[18]*X-10 PR – Security and Automation for Professional, Technote X10 communications protocol and power line interface PSC04 & PSC05”,* rev2.4.