Faculdade de Engenharia da Universidade do Porto


# Identificação e controlo de objectos em ambiente interior

José Carlos Costa Pinto Ribeiro

Dissertação realizada no âmbito do

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Major Telecomunicações

Orientador: Prof. Dr. Ricardo Morla

Co-orientador: Eng. Filipe Sousa

Junho de 2008

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2007/2008

A Dissertação intitulada

**"Identificação e Controlo de Objectos em Ambientes Interior"**

foi aprovada em provas realizadas 10/Julho/2008

**o júri**

Presidente Professor Doutor Pedro Henrique Henriques Guedes de Oliveira
Professor Catedrático da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Carlos Manuel Robalo Lisboa Bento
Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Professor Doutor Ricardo Santos Morla
Professor Auxiliar Convidado 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 - José Carlos Costa Pinto Ribeiro

Faculdade de Engenharia da Universidade do Porto

# Resumo

O ambiente que nos rodeia é constituído por um número cada vez maior de dispositivos electrónicos. Apesar destes serem cada vez mais inteligentes e autónomos, não dispensam por completo o controlo do utilizador. À medida que os dispositivos móveis se tornam mais poderosos e aumentam as suas capacidades de comunicação, torna-se cada vez maior a integração de diversas funcionalidades num só equipamento. Estes equipamentos tendem a ser uma extensão natural das capacidades do ser humano, tornando-o capaz de realizar novas ou as mesmas tarefas de um modo mais simples e mais eficaz.

As funcionalidades presentes nesses dispositivos são, assim, cada vez mais numerosas e complexas, o que obriga à criação de novos tipos de interacção homem-máquina, mais simples e intuitivos e que dispensem a utilização de *hardware* específico para o seu controlo. Como tal, propõe-se a criação de uma arquitectura de controlo através da Realidade Aumentada que permita expandir os limites das capacidades oferecidas pelos actuais interfaces, dispensando a proliferação de *displays* e evitando, assim, os custos associados à sua produção e ao consumo de energia. Uma arquitectura deste tipo deverá assentar em protocolos abertos e que funcionem de modo descentralizado, permitindo o seu funcionamento mesmo em redes repletas de dispositivos, como irão ser os ambientes habitacionais futuros.

Neste trabalho é proposta uma arquitectura e implementado um sistema baseado em Realidade Aumentada capaz de satisfazer essas necessidades. Usando esse sistema, foi possível comparar dois métodos de identificação de objectos.

# Abstract

*The number of electronic devices in our surrounding environment is increasing. Although these devices are becoming more inteligent and autonomous, they still need user intervention. The increasing power and the comunication capabilities included in mobile devices makes them capable of doing more tasks. These devices are becoming a natural extension to the human being, making him capable of completing tasks easier.*

*The number of features included in these devices and their complexity is increasing, which makes necessary the creation of new types of human-computer interaction. This means systems with simpler and more intuitive interfaces that don't require a specific hardware to control them.*

*The integration of Augmented Reality in control systems can expand the capabilities offered by the common interfaces, not requiring a dedicated display, which means energy saving and less production cost. Mobile devices are becoming more powerful, with better communication capabilities, which makes possible the integration of more features in a single device. A system architecture of this kind should rely on open protocols and work in a distributed way, avoiding bottlenecks, so it could preserve its capabilities even when applied to large networks.*

*In this work, it's proposed an architecture and a system based on Augmented Reality capable of fulfill those needs was implemented. Using this system, two different identification methods were compared.*

# Índice


1. **Introdução** **1**

1.1 Descrição do problema ..... 1
1.2 Objectivos ..... 2
1.3 Estratégia ..... 2
1.4 Estrutura ..... 2

2. **Estado da Arte** **5**

2.1 UPnP ..... 5
2.1.1 Descrição ..... 7
2.1.2 Controlo ..... 7
2.1.3 Notificações de eventos ..... 7
2.1.4 Apresentação ..... 7
2.1.5 Arquitectura UPnP AV ..... 8
2.2 Zigbee ..... 8
2.3 Realidade Aumentada ..... 10
2.3.1 Localização ..... 11
2.3.2 Marcas Identificadoras ..... 11
2.3.3 ARToolKit ..... 12
2.3.4 ARToolkitPlus ..... 12
2.4 Desenvolvimento de software - .NET Framework ..... 14
2.4.1 XAML ..... 15
2.4.2 Code-behind ..... 16
2.5 Wiimote ..... 16
2.5.1 WiimoteLib ..... 17
2.6 Porta USB ..... 18
2.7 Trabalho relacionado ..... 19
2.8 Conclusões ..... 20

3. **Arquitectura** **21**

3.1 Requisitos ... 21
3.2 Visão geral da solução ... 22
3.3 Identificação ... 23
3.4 Localização ... 24

3.4.1 Disparidade binocular ... 25
3.5 Controlo ... 28
3.6 Conclusão ... 28

4. **Implementação** **31**

4.1 Plataforma de desenvolvimento ... 31
4.2 O dispositivo controlável ... 31

4.2.1 O hardware ... 32

4.2.2 O software ... 38
4.3 O dispositivo de controlo ... 40
4.4 Interface gráfico ... 40

5. **Avaliação do trabalho** **45**

5.1 Cenários de teste ... 45

5.1.1 Validação do hardware utilizado ... 46

5.1.2 Leitura dos códigos infravermelhos ... 46

5.1.3 ARToolkitPlus ... 47
5.2 Resultados ... 48

5.2.1 O hardware ... 48

5.2.2 Leitura por infravermelhos ... 49

5.2.3 Leitura através da ARToolkitPlus ... 50
5.3 Avaliação ... 51

6. **Conclusões e trabalho futuro** **53**

6.1 Revisão do trabalho desenvolvido ... 53
6.2 Contribuições relevantes ... 54
6.3 Trabalho futuro ... 54

# Lista de figuras

**FIGURA 2.1** – PILHA DO *UNIVERSAL PLUG AND PLAY*. [1] ..... 6

**FIGURA 2.2** – A REALIDADE MISTURADA. ..... 10

**FIGURA 2.3** – BINARIZAÇÃO DA IMAGEM E DETECÇÃO DE CONTORNOS [4]..... 13

**FIGURA 2.4** – CÁLCULO DAS COORDENADAS DOS VÉRTICES DAS MARCAS ..... 13

**FIGURA 2.5** – ALGORITMO DE LEITURA DO CÓDIGO BINÁRIO [4]..... 14

**FIGURA 2.6** – TIPO DE MARCAS SUPORTADOS PELA ARTOOKITPLUS [4] ..... 14

**FIGURA 2.7** – COMPILAÇÃO DE CÓDIGO COM A .NET FRAMEWORK..... 16

**FIGURA 2.8** – ACELERÓMETRO DO WIIMOTE (À ESQ.) E WIIMOTE (À DIR.)..... 17

**FIGURA 2.9** – MÉTODO USADO PELO WIIMOTE PARA A DETERMINAÇÃO DA POSIÇÃO. ..... 17

**FIGURA 2.10** – CONSTITUIÇÃO INTERNA DO WIIMOTE..... 18

**FIGURA 3.1** – O DISPOSITIVO CONTROLADOR E OS DISPOSITIVOS CONTROLÁVEIS ..... 23

**FIGURA 3.2** – DISPARIDADE BINOCULAR..... 26

**FIGURA 3.3** – CAMPO DE VISÃO DA CÂMERA DE INFRAVERMELHOS..... 26

**FIGURA 3.4** – VARIAÇÃO DO ÂNGULO DE ABERTURA COM A DISTÂNCIA FÍSICA ENTRE OS LEDS E A DISTÂNCIA ENTRE O DISPOSITIVO E A CÂMERA. ..... 27

**FIGURA 3.5** – ÂNGULO DE CORRECÇÃO DAS COORDENADAS ENTRE AS DUAS CÂMERAS. ..... 28

**FIGURA 4.1 -** NE555 NA CONFIGURAÇÃO EM MODO ASTÁVEL [11] ..... 33

**FIGURA 4.2** – VARIAÇÃO DA TENSÃO $V_{BE}$ COM A CORRENTE NO COLECTOR. [11]..... 34

**FIGURA 4.3** – VARIAÇÃO DO PARÂMETRO $h_{FE}$ COM A CORRENTE NO COLECTOR. [11]..... 35

**FIGURA 4.4** – TENSÃO NO PINO DE SAÍDA DO NE555. [11]..... 35

**FIGURA 4.5** – ESQUEMA ELÉCTRICO DO DISPOSITIVO CONTROLÁVEL. ..... 36

**FIGURA 4.6** - DIMENSÕES DO MÓDULO DA FTDI. [11] ..... 36

**FIGURA 4.7** – CIRCUITO PROJECTADO VISTO DE TOPO. AS SETAS INDICAM OS LEDS IV. ..... 36

**FIGURA 4.8** – A PLACA DE CIRCUITO IMPRESSO PROJECTADA (À ESQ.) E CONCEBIDA (À DIR.). . 37

**FIGURA 4.9** – AS VÁRIAS VISTAS DO CIRCUITO CONSTRUÍDO. ..... 37

**FIGURA 4.10** – TRAMA ENVIADA PELO DISPOSITIVO IDENTIFICADOR ATRAVÉS DE IV. ..... 39

**FIGURA 4.11** – VISÃO GERAL DA SOLUÇÃO. ..... 40

**FIGURA 4.12** – SOBREPOSIÇÃO DO INTERFACE RECEBIDO, NO INTERFACE DO DISPOSITIVO DE CONTROLO. ..... 42

**FIGURA 4.13** – INTEGRAÇÃO DA ARTOOKITPLUS NA APLICAÇÃO. À ESQUERDA COM FUNDO TRANSPARENTE E Á DIREITA COM FUNDO OPACO. ..... 43

**FIGURA 5.1** – CENÁRIO DE TESTE DA PROBABILIDADE DE ERRO COM A VARIAÇÃO DO ÂNGULO DE VISÃO DO IDENTIFICADOR DE INFRAVERMELHOS. ..... 47

**FIGURA 5.2** – CENÁRIO DE TESTE PARA A OBTENÇÃO DA PROBABILIDADE DE ERRO ASSOCIADA AO ÂNGULO COM QUE A MARCA DE PAPEL É VISTA. ..... 47

**FIGURA 5.3** - TEMPO DE CONDUÇÃO DO LED (4 MS/DIV; 1 V/DIV). ..... 48

**FIGURA 5.4** - FORMA DE ONDA OBTIDA NO CÁTODO DO LED, ENQUANTO É ENVIADO UM CÓDIGO. (10MS/DIV; 1V/DIV) ..... 48

**FIGURA 5.5** – NÚMERO DE LEITURAS CORRECTAS E ERRADAS PARA DIFERENTES TEMPOS DE BIT. ..... 49

**FIGURA 5.6** – PROBABILIDADE DE LEITURA CORRECTA EM FUNÇÃO DA DISTÂNCIA. ..... 50

**FIGURA 5.7** – PROBABILIDADE DE LEITURA CORRECTA VARIANDO O ÂNGULO. ..... 50

**FIGURA 5.8** – COMPARAÇÃO DA PERCENTAGEM DE PROCESSAMENTO EXIGIDO. ..... 51

# Lista de tabelas

**TABELA 2.1** – DÉBITOS SUPORTADOS PELA PORTA USB ..... 19

**TABELA 2.2** – FUNÇÃO DOS PINOS DA PORTA USB ..... 19

**TABELA 5.1** – COMPARAÇÃO DAS MARCAS USADAS PELA ARTOOLKIT E PELA SOLUÇÃO DE IV. ..... 52

**TABELA 5.2** – COMPARAÇÃO DA DEPENDÊNCIA EM RELAÇÃO À DISTÂNCIA E AO ÂNGULO NA LEITURA CORRECTA DOS DOIS TIPO DE IDENTIFICADORES. ..... 52

**TABELA 5.3** – COMPARAÇÃO DAS NECESSIDADES DO DISPOSITIVO DE CONTROLO CONSOANTE O MÉTODO DE IDENTIFICAÇÃO. ..... 52

# Abreviaturas e Símbolos

| | |
|---|---|
| API | *Application Programming Interface* |
| AODV | *Ad hoc On-Demand Distance Vector* |
| BCH | Bose e CHaudhuri |
| CIL | *Common Intermediate Language* |
| CLR | *Common Language Runtime* |
| CRC | *Cyclic Redundancy Check* |
| CSMA-CA | *Carrier Sense Multiple Access - Collision Avoidance* |
| DHCP | *Dynamic Host Configuration Protocol* |
| DNS | *Domain Name System* |
| DVD | *Digital Versatile Disc* ou *Digital Video Disc* |
| FIFO | *First In First Out* |
| FFD | *Full Function Device* |
| FTDI | Future Technology Devices International |
| GENA | *General Event Notification Architecture* |
| GUID | *Globally Unique IDentifier* |
| HMD | *Head-Mounted Display* |
| HTTP | *HyperText Transfer Protocol* |
| I2C | *Inter-Integrated Circuit* |
| ID | IDentificador |
| IEEE | *Institute of Electrical and Electronics Engineers* |
| IP | *Internet Protocol* |
| IrDA | *Infrared Data Association* |
| IV | InfraVermelho |
| LED | *Light-Emitting Diode* |
| PC | *Personal Computer* |
| PDA | *Personal Digital Assistant* |
| PLC | *Power Line Communications* |
| RA | Realidade Aumentada |

| | |
|---|---|
| RFD | *Reduced Function Device* |
| SOA | *Service Oriented Architecture* |
| SOAP | *Simple Object Access Protocol* |
| SSDP | *Simple Service Discovery Protocol* |
| TCP | *Transmission Control Protocol* |
| UPnP | *Universal Plug and Play* |
| UPnP AV | *Universal Plug and Play Audio and Video* |
| URL | *Uniform Resource Locator* |
| USB | *Universal Serial Bus* |
| XAML | *eXtensible Application Markup Language* |
| XML | *eXtensible Markup Language* |
| WCF | *Windows Communication Foundation* |
| WPF | *Windows Presentation Foundation* |

# Capítulo 1

# Introdução

Este trabalho surge no seguimento de um trabalho de fim de curso, no qual se desenvolveu um sistema de controlo habitacional. Nesse sistema, os dispositivos espalhados pela habitação recorriam a redes sem fios Zigbee para comunicarem, e eram passíveis de serem controlados através de uma rede IP. O trabalho consistiu, portanto na criação de uma *bridge* Zigbee/UPnP. Como interface de controlo, usavam-se páginas *web*, onde se poderia encontrar a lista de todos os dispositivos existentes na rede habitacional e controlá-los. A evolução desse trabalho, passa pela simplificação do método de selecção e controlo do dispositivo que se pretende controlar.

## 1.1 Descrição do problema

No trabalho desenvolvido, o utilizador tinha acesso à lista de todos os dispositivos espalhados pertencentes à sua rede doméstica e necessitava de pesquisar nessa lista, o dispositivo que desejava controlar. Tornava-se necessário estudar métodos inovadores de controlo de dispositivos, que facilitessem e aumentassem a interactividade do utilizador com os mesmos. Neste sentido, surgiu a ideia de se criar um dispositivo de controlo universal que fosse capaz de comunicar com estes dispositivos simplesmente direccionando o comando para o dispositivo a controlar.

## 1.2 Objectivos

Um dos objectivos deste trabalho é melhorar a interacção homem-máquina, através da criação de métodos de controlo inovadores. Com isto, pretende-se, igualmente, uma diminuição do número de consolas de controlo através de uma uniformização dos métodos de controlo.

## 1.3 Estratégia

A estratégia utilizada passou pelo estudo de várias tecnologias de rede e comunicação e pelo estudo de diferentes métodos de identificação de objectos. Com base nesses conhecimentos adquiridos, foi criada uma arquitectura capaz de colmatar as falhas dos actuais sistemas. A Realidade Aumentada surgiu, assim, como uma alternativa promissora, uma vez que esta permite a inserção de objectos virtuais numa imagem real. Deste modo, a realidade é complementada com os respectivos interfaces de controlo.

Para a identificação e localização dos dispositivos foram comparados dois métodos diferentes. O primeiro recorre a marcas planares que refletem a luz visível com um determinado padrão e o segundo usa luz infravermelha para a emissão desses códigos.

## 1.4 Estrutura

Este texto está estruturado em seis capítulos divididos por secções e subsecções.

No primeiro capítulo é feita uma breve introdução do trabalho, mostrando o contexto onde este se insere. Após a enumeração dos seus objectivos, é relatada a estratégia adoptada para o cumprimento dos mesmos.

O segundo capítulo é reservado ao Estado da Arte. Nele são revistos alguns conceitos que levarão a uma melhor compreensão das arquitecturas, tecnologias e *frameworks* incluídas na arquitectura. É também descrito o modo de funcionamento de algum *hardware* necessário na concepção de um protótipo demonstrativo dessa arquitectura e é apresentado algum trabalho relacionado.

No capítulo 3 são referidos os requisitos da arquitectura, descrevendo o tipo de mensagens trocadas e os seus intervenientes. São também referidos alguns problemas levantados e a forma como os mesmos foram ultrapassados.

O capítulo seguinte é dedicado à forma como foi realizada a implementação da arquitectura descrita no terceiro capítulo. Ao longo das diversas secções é mostrado como se implementa cada um dos módulos necessários, quer em termos de *software* como de *hardware*.

No quinto capítulo faz-se uma avaliação do trabalho tendo por base os requisitos a que este se propõe. Para isso, são inicialmente definidos os cenários de teste e, a seguir, são mostrados os resultados retirados desses mesmos testes. No final, os resultados são avaliados, comparado-os com os obtidos por outras tecnologias.

O último capítulo é reservado às conclusões retiradas na concepção deste trabalho. O texto termina com uma referência a algumas contribuições fornecidas por este trabalho e como este deverá evoluir.

# Capítulo 2

# Estado da Arte

Este capítulo está divido em oito secções que percorrem as tecnologias estudadas neste trabalho, fornecendo ao leitor alguns elementos importantes para uma melhor compreensão das mesmas. É também feita uma caracterização dos diversos tipos de tecnologia que misturam a realidade com elementos virtuais. O capítulo termina com o estudo de algum equipamento necessário à sua implementação.

## 2.1 UPnP

O *Universal Plug and Play* é um conjunto de protocolos de rede de computadores lançado pelo UPnP Forum. As metas deste sistema distribuído são a conexão directa entre elementos de uma rede (*peer to peer*) e a simplificação da implementação de redes em casas ou escritórios. Estas são algumas das características desta tecnologia:

- Independência da tecnologia de transporte e do dispositivo. O UPnP pode ser utilizado em tecnologias como PLC, *Ethernet*, IR (IrDA), RF (*Wi-Fi*, *Bluetooth*), ou *FireWire*. Em vez de *drivers* são usados protocolos *standard*. O UPnP usa o TCP/IP e outros protocolos Internet, encaixando nas tecnologias já existentes;
- Possibilidade de acesso a um interface de controlo que permite a interacção do utilizador com o dispositivo através de um *web browser*;
- Independência do Sistema Operativo e da linguagem de programação, uma vez que não são especificadas regras para a API das aplicações de controlo;
- Extensibilidade. Os dispositivos UPnP podem ter serviços à medida do fabricante encaixado nos moldes da arquitectura base.

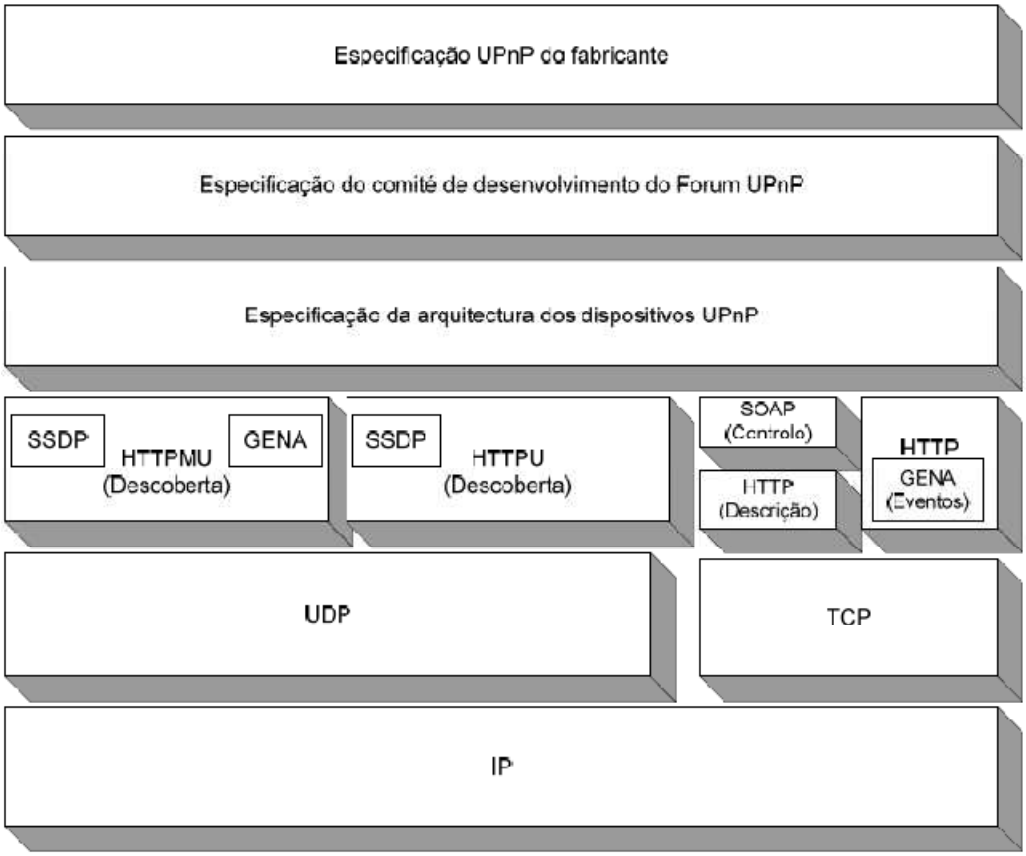


**Figura 2.1** – Pilha do *Universal Plug and Play*. [1]

Esta arquitectura suporta a descoberta automática de dispositivos e seus serviços, sem necessidade de configuração, permitindo a um dispositivo:

- Juntar-se dinamicamente a uma rede;
- Obter um endereço IP;
- Anunciar o seu nome;
- Anunciar as suas capacidades quando lhe é pedido;
- Detectar a presença e as capacidades de outros dispositivos na rede;
- Abandonar a rede, reportando esse acontecimento.

Os serviços de DHCP e DNS podem ser usados caso estejam presentes na rede.

Quando um dispositivo é adicionado a uma rede, o protocolo de descoberta do UPnP permite que o mesmo se anuncie aos *Control Points*. A informação contida neste anúncio inclui a descrição do próprio dispositivo e dos serviços que ele suporta. Este protocolo de descoberta é baseado no *Simple Service Discovery Protocol* (SSDP).

### 2.1.1 Descrição

Para que um *Control Point* possa saber mais acerca do dispositivo, este tem que descarregar as informações pormenorizadas do mesmo. Estas informações estão acessíveis a partir de um URL contido na mensagem de anúncio. Todas estas descrições usam a linguagem XML e podem incluir dados sobre o fabricante, o modelo e o número de série do produto.

Esta descrição pode também incluir listas de dispositivos embutidos e os seus serviços, assim como URLs para controlo, notificação e apresentação. Por cada serviço são descritos os respectivos comandos e os seus argumentos e ainda as variáveis de estado associadas a cada comando, descritas por tipo e valores limite e o processo de notificação de alterações nas mesmas.

### 2.1.2 Controlo

Depois de um *Control Point* ter recolhido a descrição de um dispositivo, este está pronto a enviar comandos aos serviços desse dispositivo. Para isso, são enviadas mensagens para o URL de controlo também expressas em XML, usando o encapsulamento SOAP. Em resposta a essas mensagens de controlo, o serviço envia as correspondentes mudanças nas variáveis de estado.

### 2.1.3 Notificações de eventos

Como já foi referido, o estado do serviço é dado pelas suas variáveis de estado.

O UPnP permite a um dispositivo subscrever notificações de alterações num serviço. Assim, quando o valor de uma variável de estado é alterado, esses dispositivos que subescreverem a notificação, são informados dos novos valores das variáveis. Estas mensagens de notificação contêm os nomes de uma ou várias variáveis de estado e os seu valores actuais. A linguagem usada para reportar estas alterações é XML e são formatadas segundo a arquitectura GENA.

### 2.1.4 Apresentação

A apresentação é uma página web, cujo URL é indicado na mensagem de anúncio do dispositivo, que permite ao utilizador saber o estado de um dispositivo e alterá-lo. O nível de controlo depende somente das funcionalidades implementadas na interface e que, obviamente, sejam suportadas pelo dispositivo.

Como o suporte da página é feito por um *browser*, este não pode ser notificado das alterações ocorridas nas variáveis de estado. É necessário, portando, que seja o cliente a inquirir o dispositivo sobre as alterações ocorridas nas suas variáveis de estado.

### 2.1.5 Arquitectura UPnP AV

Geralmente, em cenários UPnP, um *Control Point* faz a gestão de um ou mais dispositivos UPnP por forma a ser atingido um certo objectivo. Apesar do *Control Point* efectuar a gestão de vários dispositivos, todas as interacções ocorrem isoladamente entre o *Control Point* e cada dispositivo, para se atingir uma finalidade.

A maior parte dos cenários UPnP AV envolve a transferência de conteúdos multimédia (vídeos, músicas, imagens,etc.) de um dispositivo para outro, sendo que um *AV Control Point* interage com dois ou mais dispositivos que actuam como fonte ou receptor de informação.

Apesar das acções destes dispositivos serem controladas e sincronizadas pelo *Control Point*, entre estes existem interacções que envolvem protocolos de comunicação não UPnP. O *Control Point* inicializa e configura ambos os dispositivos de forma a que o conteúdo multimédia seja transferido, mas esta posterior transferência não o envolve, pelo que, após os comandos serem efectuados a ambos os dispositivos, a interacção do *Control Point* já não é necessária para que o comportamento desejado seja atingido. Num cenário UPnP AV estão envolvidas três entidades: *Control Point*, *Media Server*, *Media Renderer*.

Sendo estas entidades independentes entre si, e existentes isoladamente em dispositivos tais como um comando remoto (*Control Point*), um leitor de DVD (*Media Server*) ou uma televisão (*Media Renderer*), existem configurações que combinam estas entidades num mesmo dispositivo. Por exemplo, uma televisão - *Media Renderer* – geralmente tem um sintonizador embutido, que capta um dado canal de dados e pode encaminhar esses mesmos dados para o ecrã, actuando como um *Media Server*. Nestes casos, os componentes individuais podem interagir utilizando os protocolos UPnP normalizados (SOAP sobre HTTP) ou através de mecanismos privados de comunicação. De qualquer forma as funcionalidades de cada entidade lógica mantêm-se inalteradas, embora, neste último caso, os dispositivos não tenham capacidade de comunicar com outros dispositivos que não implementem esses mecanismos.

## 2.2 Zigbee

Das novas tecnologias sem fios para redes de baixo débito e baixo consumo que estão agora em expansão, o Zigbee é sem dúvida, uma das mais interessantes e completas. Ela foi criada com o objectivo de assentar em dispositivos simples e de baixo custo que possam ser espalhados pelas nossas casas tornando-as mais inteligentes.

Ao contrário de outras tecnologias que se baseiam em mecanismos de *broadcast* pelos nós da rede, este usa AODV para a descoberta da rota, baixando o tráfego na rede.

As principais características do ZigBee são:

- *Duty-cycle* baixo (< 0,1%);
- Capacidade do dispositivo de permanecer inactivo durante um longo período de tempo, permitindo uma esperança de vida de vários meses, ou mesmo anos;
- Elevada densidade de nós por rede, sendo assim indicada para redes de sensores;Espaço de endereçamento para 18,450,000,000,000,000,000 dispositivos (endereço de 64 bits IEEE);
- Opção de *time slot* garantido para aplicações que necessitam de uma baixa latência;
- Acesso ao canal do tipo CSMA-CA, que permite um débito elevado e uma latência baixa para dispositivos com um baixo *duty-cycle* como sensores ou controladores;
- Suporte de topologias estáticas e dinâmicas, tais como estrela, árvore e emalhada.

As comunicações por ZigBee são feitas em faixas de frequências não-licençadas (ISM), assim sendo, estão disponíveis as faixas de 2.4Ghz (mundialmente), 915Mhz (na América) e 868Mhz (na Europa). A taxa de transferência máxima é de 250kbps na frequência de 2.4Ghz, operando com 16 canais, 40kbps na frequência de 915Mhz operando com 10 canais e 20kbps na frequência de 868Mhz operando com 1 canal.

Os dispositivos Zigbee podem ser divididos consoante a sua função em:

- *Full Function Device* (FFD) - pode desempenhar qualquer papel numa rede ZigBee, podendo desempenhar a função de coordenador de uma rede, *router* ou *end device*. Tratam-se de dispositivos de construção mais complexa, com boas capacidades de processamento e de memória;
- *Reduced Function Device* (RFD) – é limitado a uma configuração com topologia em estrela, não podendo actuar como um coordenador da rede. Pode comunicar com um coordenador ou com um *router*, não podendo trocar informação directamente com outro RFD. São dispositivos de construção mais simples, de baixo custo.

Devemos observar que, em topologias com configuração estrela, uma rede ZigBee requer pelo menos um dispositivo FFD actuando como coordenador da rede e os demais dispositivos podem ser do tipo RFD para reduzir o custo do sistema. Para topologias ponto-a-ponto (*peer-to-peer*) e em árvore, todos os dispositivos devem ser FFD.

Existem três diferentes tipos de equipamentos ZigBee, com diferentes funções:

- ZigBee *Coordinator*: Há apenas um coordenador em cada rede, que é o dispositivo que tem o maior número de funções. O coordenador é capaz de criar uma rede, tornar-se a raíz da rede e ser o único dispositivo com autonomia de comutar dados entre redes. Para além disso ele faz o armazenamento das informações da rede.
- ZigBee *Router*: São os dispositivos que fornecem informações aos outros dispositivos da rede, ou seja, faz o roteamento das informações como um *routers* comum de *Wi-Fi*.
- ZigBee *End Device*: Tem apenas a função de trocar informações com um coordenador ou *routers*. Uma vantagem é ser um dispositivo com pouca memória pois não precisa de rotear informações, e consequentemente, o seu custo é menor.

## 2.3 Realidade Aumentada

Alguns sistemas actuais são difíceis de gerir ou controlar devido ao seu interface. À medida que estes se tornam mais complexos, torna-se difícil a integração de toda a informação disponível num interface intuitivo.

A Realidade Aumentada é um passo inovador nesse sentido. Em vez de se usarem vários *displays* para se representar os vários tipos de informação, esta é sobreposta, através de elementos virtuais, à realidade. Torna-se assim possível expandir o universo das aplicações para junto dos utilizadores, onde quer que estes estejam criando interfaces mais ricas, poderosas e intuitivas.

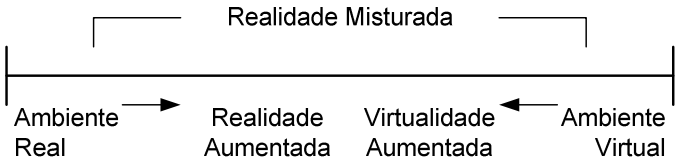


**Figura 2.2** – A Realidade Misturada.

O exemplo mais simples de RA seria a sobreposição de um objecto 2D numa cena 3D, porém, é comum vermos esta técnica aplicada a objectos virtuais tridimensionais que se fundem com a imagem real, parecendo fazer parte dela.

Os elementos sobrepostos, não são visíveis a olho nú, daí ser necessária utilização de algum tipo de *display*, como um monitor de computador. A evolução da tecnologia, levará a que seja possível integrar em dispositivos de dimensões reduzidas e com elevado poder de

processamento e que possibilitarão criar HMDs baseados em óculos transparentes, com câmeras integradas e que possibilitarão a utilização natural destas tecnologias que nos proporcionarão experiências totalmente imersivas em ambientes semi-virtuais.

Esta técnica pode trazer inúmeras vantagens no campo da computação móvel, uma vez que não obriga o utilizador a deslocar-se a pontos de controlo. O problema principal desta técnica continua a residir a dificuldade da criação de HMD ergonómicos, com capacidades de comunicação, processamento e autonomia que levem à aceitação da sociedade. O dispositivo ideal, deverá aproximar-se de uns óculos transparentes, em que as lentes apenas adicionam à imagem real os objectos virtuais. Neste momento, os sistemas de Realidade Aumentada são baseados nos equipamentos portáteis disponíveis, desde computadores a telemóveis, não sendo, portanto, os mais indicados para uma utilização massiva.

### 2.3.1 Localização

A localização de objectos e pessoas é cada vez mais importante nos nossos dias. A integração da localização em tempo real nos sistemas actuais aumenta a qualidade e a quantidade dos serviços que podem ser disponibilizados aos utilizadores.

Várias tecnologias têm vindo a ser estudadas, e várias já são alvo de utilização diária pelo público em geral, como o GPS. Os procedimentos básicos de localização baseiam-se normalmente em triangulações de distâncias a objectos cuja localização é conhecida. Para o cálculo da distância podem-se usar vários métodos, como o tempo de propagação, o tempo de chegada, o tempo de ida e volta ou a atenuação de um sinal.

Para a Realidade Aumentada, apenas interessa o posicionamento do objecto em relação ao seu campo de visão. Isto faz com que a implementação das tradicionais técnicas de localização com este fim sejam demasiado complexas.

Como as ondas electromagnéticas são capazes de atravessar obstáculos, a utilização destas pode trazer a vantagem de eliminar o problema da oclusão das marcas, uma vez que deste modo, a identificação não ficaria sujeita à visualização do objecto. Porém, usando frequências dessa gama, a localização dos sensores seria muito mais complexa, uma vez que não se poderiam usar câmeras convencionais para o posicionamento dos feixes, uma vez que estas são sensíveis apenas a frequências próximas do visível.

### 2.3.2 Marcas Identificadoras

Os métodos de detecção de objectos podem-se dividir em dois grupos: os que usam marcas, e os que não as usam. Os métodos que não usam essas marcas são muito pesados em termos de

processamento, requerem aprendizagem, são mais sujeitos a falhas e não distinguem objectos idênticos, porém têm a vantagem de não necessitarem de acréscimo de informação, bastando a visualização do objecto para que este seja identificado [2].

As marcas usadas pelos sistemas de Realidade Aumentada, podem ser padrões quadrados a preto e branco, LEDs, as mãos do utilizador, etc. Estas soluções baseiam-se, na emissão de luz directa ou reflectida e que é depois detectada por uma câmera. As marcas baseadas em emissão de luz requerem um fonte de energia, porém, podem ser detectadas mesmo que a luz ambiente seja quase nula. As marcas baseadas apenas na reflexão, não necessitam de fonte de energia, mas o sistema de visão, em ambientes com pouca luz, pode ser o obrigado que iluminar as marcas, dando origem a maiores perdas originadas tanto na propagação como na reflexão dos feixes.

O tipo de marca utilizada pelo sistema altera-o profundamente, especialmente no que diz respeito ao *hardware* necessário.

### 2.3.3 ARToolKit

ARToolKit é biblioteca de *software* escrita em C/C++ voltada para o desenvolvimento de aplicações de Realidade Aumentada que fornece técnicas de Visão Computacional para o cálculo da posição e da orientação da câmera em relação aos marcadores em tempo real, de forma a que os objectos virtuais possam ser sobrepostos na imagem real.

A ferramenta ARToolKit permite a construção de sistemas do tipo *see-through* de Realidade Aumentada, tanto com visão óptica directa como na baseada em vídeo. Nestes sistemas, os objectos virtuais são adicionados ao mundo real independente do dispositivo de visualização que está a ser utilizado.

Actualmente, o ARToolKit corre nas plataformas, Linux, Windows 95/98/NT/2000/XP/Vista e Mac OS X. Existem versões separadas para cada uma destas plataformas, as quais conservam as funcionalidades, mas o desempenho pode variar. [3]

### 2.3.4 ARToolkitPlus

Existem várias bibliotecas derivadas da original ARToolkit. Uma delas é a ARToolkitPlus [2], criada para ser usada num projecto e, posteriormente tornada disponível ao público. Entre os vários melhoramentos a que o código foi sujeito, destacam-se algumas novas funcionalidades, tais como, o suporte de imagens em tons de cinzento de 8-bit a pensar nos dispositivos com recursos mais limitados, como PDAs e a capacidade de reconhecimento de marcas simples que não requerem treino.

O processo de detecção das marcas inicia-se com a captura da imagem real e a consequente conversão para cinzentos. Em seguida, a imagem é saturada de modo a ficar binarizada - os tons mais claros que um certo limite ficam brancos, e mais escuros ficam pretos. A fase seguinte consiste em detectar contornos (com a utilização de um filtro passa-alto). Na Figura 2.4 (à direita) são visíveis os 3 contornos a considerar. Toda a restante informação é descartada.

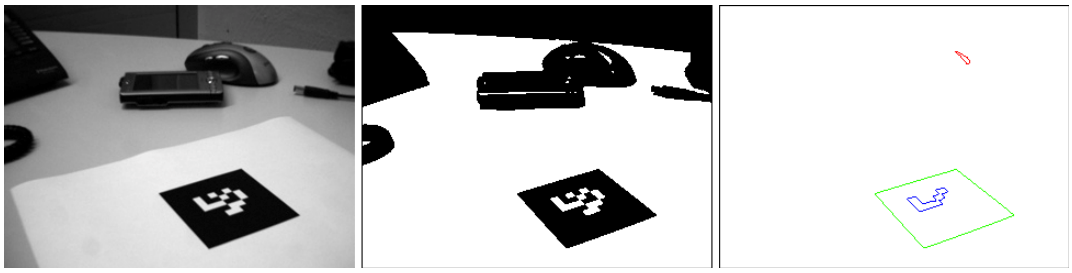

**Figura 2.3** – Binarização da imagem e detecção de contornos [4]

Em seguida, todos os contornos fechados são testados. Pode acontecer que as linhas podem não ser rectas perfeitas, uma vez que as marcas poderão não ser perfeitamente planares, por isso o algoritmo de pesquisa não é rígido, deixando margem para pequenas imperfeições.

Depois dos rectângulos serem detectados, o ARToolkitPlus calcula o centro de massa dos pontos correspondentes aos vértices e traça uma recta desde o vértice situado a uma distância maior de um ponto arbitrário até ao centro (Figura 2.4 ao centro). Os pontos situados mais à esquerda e à direita são os 2º e 3º pontos. O comprimento da linha é testado para os outros 2 pontos detectados, se a diferença de distância entre estes e aquela compreendida entre os pontos $c_1$ e $c_2$ for menor que um valor limite, os pontos são considerados um rectângulo válido.

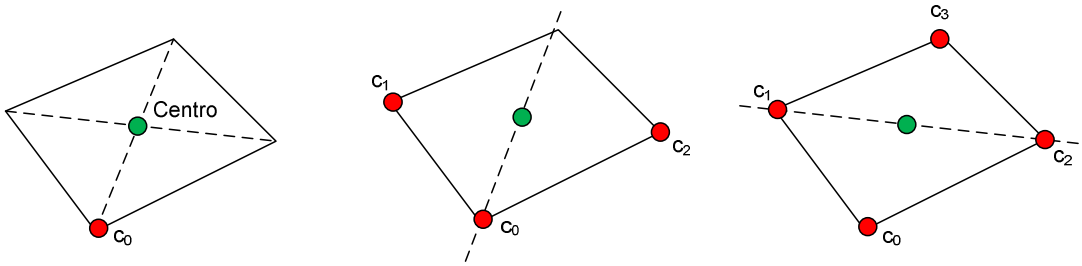


**Figura 2.4** – Cálculo das coordenadas dos vértices das marcas

Depois de descobertos, os índices dos pontos são postos por ordem, no sentido dos ponteiros do relógio. A fase seguinte é detectar se a região delimitada por estes contornos contém uma marca válida. Para isso, é necessário recuperar a informação da imagem no interior do contorno. Em seguida, é criado um mapa de bits, baseado no nível de cinzento dos pontos contidos naquela região.

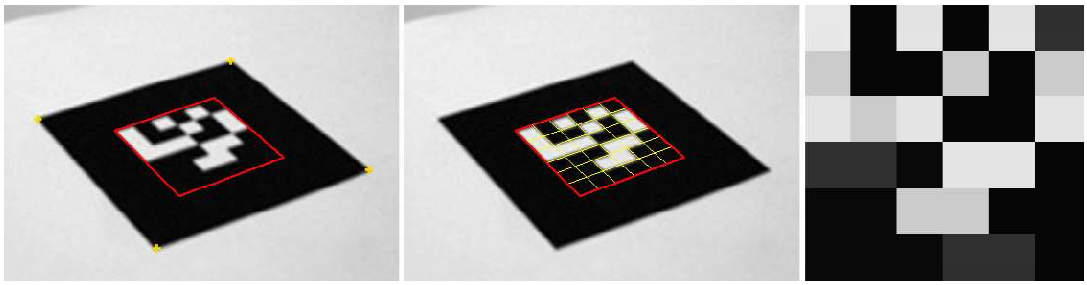

**Figura 2.5** – Algoritmo de leitura do código binário [4]

A detecção do código varia com o tipo de marca usada. Existem vários tipos de marcas, a destacar as que usam uma tabela e que, portanto, necessitam de treino prévio, e as que reproduzem directamente o código binário associado. As marcas deste último tipo podem ser idênticas às usadas pela ARToolkit (usam redundância dos dados x4) ou BCH – modo que usa códigos CRC para a validação dos mesmos, conseguindo para uma matriz de 6x6 aproveitar 12 bits, em vez dos 9 da outra versão. Deste modo, é possível obter 4096 marcas diferentes, enquanto que as tradicionais só poderiam codificar até 512.

Na versão mais recente da biblioteca ARToolkitPlus foi adicionado ainda mais um tipo de marca conhecido por DataMatrix. Trata-se de um código igualmente bidimensional e que pode conter até 2 kBytes de informação.

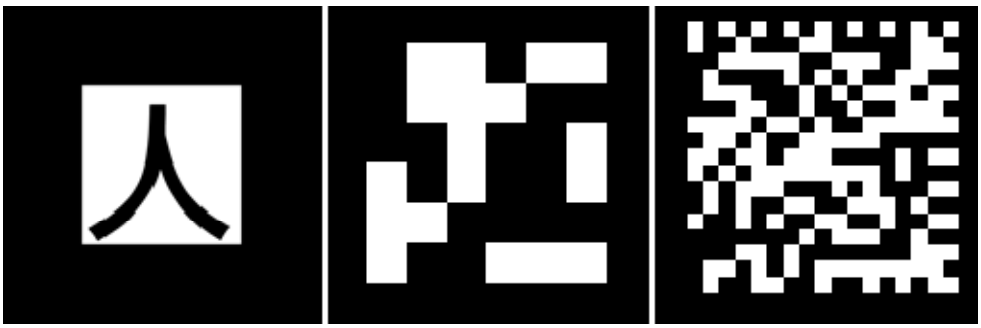

**Figura 2.6** – Tipo de marcas suportados pela ARTookitPlus [4]

## 2.4 Desenvolvimento de software - .NET Framework

As mais recentes versões das *frameworks* .NET – a 3.0 e a 3.5 – suportadas pelo Windows XP SP2 e incluídas no mais recente sistema operativo da Microsoft – o Windows Vista – trazem novas funcionalidades interessantes para os programadores. O WPF e o WCF são disso exemplo, introduzindo bastantes novidades, possibilitando a criação de aplicações mais ricas e reduzindo o tempo de concepção das mesmas.

O WPF é um conjunto de classes responsáveis pela parte gráfica das aplicações, que integra funcionalidades 3D em *managed code*, facilitando, assim, a tarefa de criar interfaces mais ricas. Para a descrição das interfaces foi criada uma nova linguagem.

### 2.4.1 XAML

XAML é uma linguagem baseada em XML e com ela pode-se descrever os elementos gráficos, os eventos associados e fazer *data binding*. Uma das grandes vantagens do uso desta linguagem é a possibilidade de serializar a informação, permitindo passar o interface gráfico para XML e vice-versa. O XAML, separa, portanto, a informação de apresentação da lógica de negócio do programa. Esta linguagem está também a ser usada para fins para os quais não foi inicialmente concebida como a descrição de documentos (XML *Paper Specification*), a versão simplificada da mesma, de modo a ser suportada em *web browsers* (Silverlight) e a descrição de *workflows* usada pela componente WF da plataforma .NET. [5]

O XAML suporta vários tipos de elementos, podendo estes serem:

- *Control Elements* – os elementos habituais como Checkbox, Button, Scrollbar, Listbox, etc.;
- *Panel Elements* – responsáveis pelo *layout* do interface como o Canvas e o Grid;
- *Shape Elements* – formas geométricas simples como rectângulos e elipses;
- *Geometric Elements* – formas 2D mais complexas como arcos ou formas irregulares;
- *Document Elements* – gerem a apresentação dos textos, como parágrafos e estilos;
- *Root Elements* – elemento raiz obrigatório e que contém todos os outros.

Para a descrição de uma propriedade estão definidas duas abordagens. A mais simples, em que os atributos do nó correspondem à propriedade:

```
<Button Name="button1" Opacity="0.6" Background="Black" Foreground="White"
Margin="29,46.14,0,0" Height="26.915 Width="119.195">Turn ON</Button>
```

Ou, então, mais complexas, definidas como <*elemento.propriedade*>

```
<Image Name="videoImage">
        <Image.RenderTransform>
                <ScaleTransform CenterY="245" ScaleY="-1"/>
        </Image.RenderTransform>
</Image>
```

O código que processará os eventos é referido como um atributo do elemento. O código seguinte, activará a função *processClick*, definido no *code-behind*:

```
<Button Name="button1" Click="processClick">Turn ON</Button>.
```

A função *processClick* receberá dois argumentos, o objecto responsável pelo evento, e um descritor do evento. O tipo de descritor depende do mesmo do tipo de evento em causa.

### 2.4.2 *Code-behind*

O *code-behind* pode ser escrito numa das várias linguagens suportadas pela .NET Framework, ou seja, qualquer linguagem que possa ser compilada para CIL, uma linguagem de nível intermédio e que é compilada pelo CLR para código nativo durante a execução do programa.

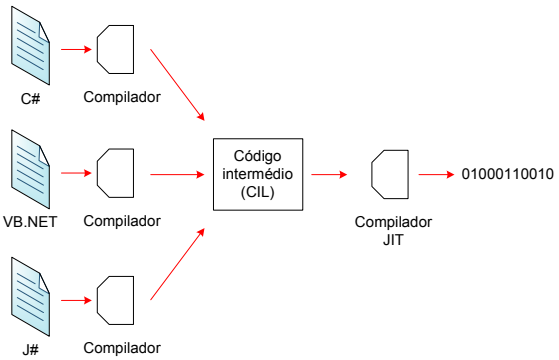


**Figura 2.7** – Compilação de código com a .NET Framework

## 2.5 Wiimote

Wiimote é o nome por que é conhecido o comando na mais recente consola da Nintendo – a Wii (abreviatura do inglês *Wii Remote*), é um dipositivo bastante interessante do ponto de vista tecnológico. Este comando está equipado com um acelerómetro de três eixos – o ADXL330 da Analog Devices. Através dos dados enviados pelo acelerómetro, é possível saber a aceleração decomposta nas três componentes, sendo possível calcular a posição estática do comando (decompondo a aceleração da gravidade nos 3 eixos) ou tentado perceber qual o movimento que o utilizador está a imprimir ao aparelho, medindo a variação dos valor obtidos.

A posição é calculada, em alguns sistemas, recorrendo à aceleração, contudo, é necessário adicionar integradores ao sistema. O problema da inclusão de integradores é o facto destes acumularem os erros, e, se este for sistemático, o desvio do valor calculado em relação ao valor correcto tenderá para o infinito. Como tal, e a pensar nos jogos que necessitam de atingir alvos no ecrã, ou seja, necessitam de grande precisão, a Nintendo adicionou ao Wiimote uma câmera CMOS que lê os feixes de luz infravermelha provenientes da chamada Sensorbar.

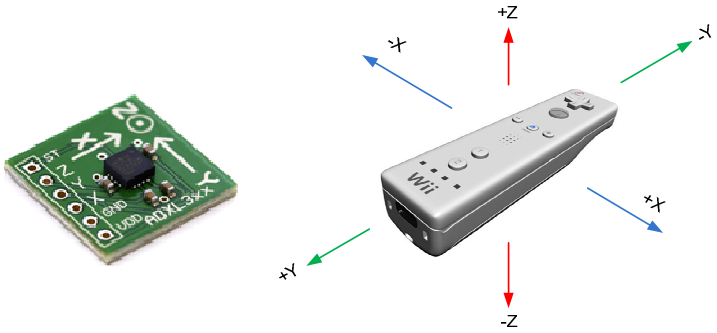


**Figura 2.8** – Acelerómetro do Wiimote (à esq.) e Wiimote (à dir.)

A Sensorbar, é uma barra com 5 LEDs emissores de luz infravermelha em cada extremo. A câmera, que é capaz de obter imagens 1024x768 a 200 Hz, lê a posição relativa destes feixes em relação ao seu campo de visão e envia estes dados para um barramento I2C. Estes dados são depois enviados por Bluetooth a uma frequência de 100 Hz para a consola, juntamente com a informação dos acelerómetros e dos botões. Os valores da posição calculados são, assim, apenas dependentes da posição da Sensorbar, não havendo acomulação de erros.

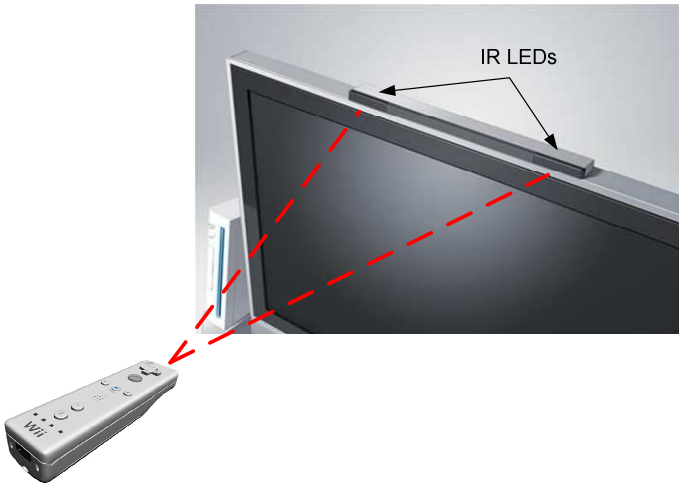


**Figura 2.9** – Método usado pelo Wiimote para a determinação da posição.

### 2.5.1 WiimoteLib

Para aceder aos dados emitidos pelo Wiimote, existem algumas bibliotecas que permitem o acesso aos dados provenientes do comando de uma forma mais simples. Uma das mais conhecidas é a WiimoteLib [6]. Esta biblioteca de código aberto, fornece uma API simples para a recepção e envio de mensagens entre o Wiimote e um PC.

Inicialmente é necessário emparelhar o comando com o computador. Para isso é necessário pressionar os botões 1 e 2 simultaneamente, para que este fique com o seu interface rádio em

modo Discoverable. O comando sinaliza a entrada neste modo, ligando os dois LEDs centrais de modo intermitente. De seguida, basta seleccioná-lo na lista de dispositivos fornecida pelo gestor de Bluetooth do computador e seleccionar a opção sem palavra-passe. O Wiimote será, assim, adicionado ao sistema como um dipositivo de interacção humana.

Depois de adicionado ao sistema, esta biblioteca permite a troca de mensagens com o comando. Durante a sua fase de conexão, esta começa por pesquisar o GUID da classe dos dispositivos de interacção humana, e, de seguida, pesquisa, nessa lista, aquele que possui o *Vendor ID* e o *Product ID* correspondentes ao Wiimote. Quando encontra, esta cria um FileStream para comunicação com o comando.

A informação trocada com HIDs é sempre organizada em relatórios, que não são mais que pequenos conjuntos de informação. Neste caso, estes relatórios têm 22 *bytes* de comprimento, contendo as informações dos sensores, e, do comando opcional que é possível acoplar ao Wiimote. Esta biblioteca, recebe essas informações e organiza-as numa estrutura de dados, e passa-os para o programa principal, gerando eventos.

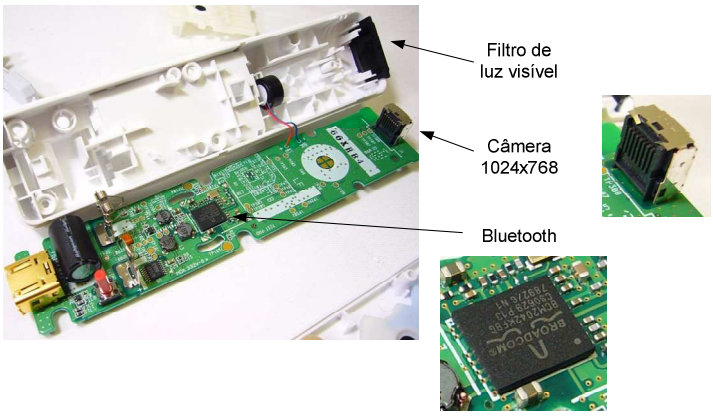


**Figura 2.10** – Constituição interna do Wiimote.

## 2.6 Porta USB

Hoje em dia, a porta USB é largamente utilizada para ligar periféricos a um computador. Esta porta possui um débito máximo que pode atingir os 480 Mbit/s e é capaz de fornecer uma corrente máxima de 500 mA [7]. Estas características tornam a porta USB como uma boa opção para ligar a maioria dos periféricos.

**Tabela 2.1** – Débitos suportados pela porta USB

| Modo | Suportado na versão 1.1 | Suportado na versão 2.0 | Débito (Mbit/s) |
|---|---|---|---|
| *Low Speed* | Sim | Sim | 1,5 |
| *Full Speed* | Sim | Sim | 12 |
| *High Speed* | Não | Sim | 480 |

Ao contrário da porta RS232, esta porta funciona como um barramento de dados, permitindo que vários dispositivos sejam ligados através de uma só porta. Para que isso seja possível, os dados que circulam no barramento são empacotados em descritores, o que significa que um dispositivo, para comunicar através desta porta, necessita, para além de uma grande velocidade de relógio capaz de ler os dados, de possuir rotinas de encapsulamento dos dados. Isto aumenta a complexidade da concepção de dispositivos que comuniquem directamente com a USB. Para facilitar esta tarefa, existem circuitos integrados específicos para estas funções, que trabalham internamente com frequências de relógio altas, capazes de receber os dados encapsulados à velocidade máxima, processar estes pacotes e enviar, por outro protocolo mais simples, por exemplo por RS232 ou RS245, com *baudrates* muito inferiores, para o controlador do dispositivo em questão, podendo este, neste caso, ser bastante mais simples e mais barato.

**Tabela 2.2** – Função dos pinos da porta USB

| Pino | Nome | Cor | Descrição |
|---|---|---|---|
| 1 | VCC | Vermelho | +5 V |
| 2 | D- | Branco | Dados (-) |
| 3 | D+ | Verde | Dados (+) |
| 4 | ID | - | Distingue o tipo de conector |
| 5 | GND | Sim | 480 |

## 2.7 Trabalho relacionado

Existe já algum trabalho no sentido de uniformizar o controlo através da Realidade Aumentada. A HP tem algum trabalho de pesquisa nesta área, através de um projecto denominado CoolTown [8]. Apesar deste ser anunciado em vários sites como estando em *open source*, a verdade é que a quantidade de informação acerca do mesmo é neste momento bastante

reduzida. Quando se consulta a página do projecto, é-se notificado que estará disponível em breve, não fornecendo assim qualquer informação relevante.

Um trabalho idêntico foi desenvolvido por uma equipa de engenheiros da Coreia do Sul. Toda a informação resume-se à existência de um *paper* [9] no *site* do IEEE, no qual se refere a construção de um dispositivo de controlo único recorrendo à Realidade Aumentada e ao UPnP. As marcas de papel, são naquele caso, ilumindadas com luz IV. É feita uma referência a uma futura implementação com *beacons* de luz infravermelha, mas não é explicitado nada acerca da arquitectura, o tipo de mensagens trocadas, bem como o método usado para a construção dinâmica do interface. Os resultados, resumem-se a algumas imagens sobre o interface.

Um outro trabalho relacionado é o recente Microsoft Surface [10]. Este sistema assemelha-se a uma simples mesa onde são colocados os diversos dispositivos que se pretendem controlar. A localização e identificação de dispositivos é baseada em técnicas que usam luz IV. Neste caso, os dispositivos são colocados em cima do Surface possuindo uma etiqueta reflectora. O Surface, irradia essa etiqueta com luz infravermelha e as câmeras colocadas por baixo captam os pontos reflectores. Com essa informação, é possível determinar qual o dispositivo colocado em cada local. O interface recorre à tecnologia WPF e é projectado no tampo da mesa.

Pode-se também considerar trabalho relacionado, as aplicações criadas por Johnny Chung Lee utilizando o Wiimote como câmera de infravermelhos e que usam a biblioteca de comunicações WiimoteLib para uma fácil concepção de sistemas que necessitam de localizar pontos brilhantes no campo de visão.

## 2.8 Conclusões

Este capítulo pretendeu cobrir os aspectos mais importantes das tecnologias incluídas neste trabalho, de modo a facilitar a compreensão dos assuntos aqui abordados. No capítulo seguinte veremos como estas tecnologias se integram na arquitectura do sistema apresentado.

# Capítulo 3

# Arquitectura

Este capítulo está dividido em seis secções que descrevem a arquitectura da solução apresentada. Começa-se por apresentar uma visão geral da mesma, e, em seguida analisa-se com mais pormenor, cada um componentes que a compõem. As secções seguintes mostram como foram concebidos os protótipos funcionais dos principais elementos da solução: o dispositivo de controlo e o controlável.

## 3.1 Requisitos

Para que seja possível o controlo de dispositivos pertencentes a uma rede doméstica, utilizando um controlo remoto único, é necessário conceber uma arquitectura que possibilite a comunicação entre os vários dispositivos de uma forma padronizada, permitindo a flexibilidade suficiente para se adaptar às especificidades de cada dispositivo. Por outro lado, a arquitectura deverá oferecer poderosas capacidades de visualização, de modo a se poderem construir interfaces de controlo de utilização mais simples e eficaz.

Para a sua validação é necessária a criação de um controlo universal que recorra à Realidade Aumentada para a representação dos intefaces de controlo, bem como a concepção de um protótipo funcional de um dispositivo controlável.

A solução deverá obter níveis aceitáveis de *performance*, ou seja, os tempos de identificação e apresentação do respectivo interface devem ser curtos e a distância máxima entre o utilizador e o dispositivo deve ser suficiente para que não seja necessária a aproximação do utilizador. Quanto ao protocolo de rede, este deverá funcionar de modo distribuído, de modo a evitar o congestionamento de nós da rede.

## 3.2 Visão geral da solução

Para que os dispositivos que se encontram espalhados pela casa sejam passíveis de controlo através da Realidade Aumentada foram criadas duas identidades: os dispositivos controladores e os controláveis. Os dispositivos controláveis são aqueles que requerem a recepção de comandos provenientes do utilizador e que, consequentemente, actuam de modo a cumprir essas ordens. Os dispositivos controladores têm como funções a apresentação dos interfaces de controlo oferecidas pelos dispositivos controláveis, e são responsáveis pelo envio dos comandos do utilizador para o respectivo dispositivo. Para cada tipo de dispositivo foi criado um protótipo funcional, de modo a ilustrar a solução apresentada.

O processo de identificação, localização e controlo proposto de um dispositivo é o seguinte:

- O dispositivo controlável anuncia-se na rede, enviando a sua descrição, a descrição dos serviços por si suportados e o seu interface;
- O utilizador, move-se, transportando com ele o dispositivo de controlo. Este vai mostrando o campo de visão da sua câmera e envia, periodicamente, luz IV.
- Quando um dispositivo controlável detecta essa radiação, significa que este se encontra no campo de visão de um *control point* e activa, também, a emissão de luz IV, enviando para o dispositivo de controlo o código 255;
- O *control point* é notificado (por GENA - UPnP) que existem, na rede, um dispositivo sem identificador atribuído. Este verifica a lista de identificadores já atribuídos e escolhe um livre, enviando o comando SetID (do serviço UPnP *IDManagerService*) a esse dispositivos, atribuindo-lhe um identificador único (compreendido entre 1 e 254).
- O dispositivo controlável recebe o novo código e passa a emiti-lo por infravermelhos. O *control point* recebe o código enviado, selecciona o respectivo interface gráfico e representa-o na posição calculada através das coordenadas do feixe, sobrepondo o interface de controlo à imagem real capturada;
- Os eventos que ocorrerem no interface e que, para os quais, foi explicitado que seriam alvo de notificação, são comunicados ao dispositivo por UPnP;
- O dispositivo controlável recebe a descrição do evento, reenvia-a aos *control points* que subescreveram notificação e processa a informação contida na descrição do evento, alterando as propriedades correspondentes;
- Quando uma propriedade é alterada, o novo valor é comunicado a todos os dispositivos de controlo. Quando as propriedades relativas à apresentação são alteradas, os *control points* recebem o novo interface e actualizam a sua apresentação.

**Figura 3.1** – O dispositivo controlador e os dispositivos controláveis

O dispositivo controlador (*control point*) descobre os dispositivos presentes na sua rede e os serviços por cada um suportado através de SSDP. Os dispositivos que sejam adicionados à rede são automaticamente adicionados à lista juntamente com um conjunto de informações que o descrevem. Já no caso da remoção, esta pode ser feita por anúncio por parte do dispositivo, ou se o tempo estabelecido para o novo anúncio desse dispositivo for ultrapassado, significando que ocorreu perda de conectividade com o mesmo. Neste caso, o dispositivo de controlo liberta todas as informações relativas a esse dispositivo.

## 3.3 Identificação

A identificação dos objectos é feita através da colocação de marcas emissoras de luz infravermelha. Estas marcas necessitam de possuir energia e controlo, porém, como se pretende a detecção apenas de dispositivos electrónicos (controláveis), a alimentação e o controlo destas podem facilmente ser incluídos no dispositivo.

Nesta solução, os dispositivos controláveis devem suportar os serviços UPnP de atribuição dinâmica de identificadores e o serviço de gestão da interface e processamento de eventos.

O IDManagerService é um serviço UPnP responsável pela compressão da gama de endereços únicos (GUID) num espaço de endereçamento muito mais reduzido. Isto é conseguido através da alocação dinâmica dos identificadores, passando a ser necessário apenas um número de combinações que identifique univocamente os dispositivos visíveis por qualquer um dos *control points* presentes na rede. Por outras palavras, usando 8 bits para o identificador, é possível ter uma rede com um número ilimitado de dispositivos (inactivos), dos quais 254 (os identificadores 0 e 255 estão reservados) estão simultaneamente no campo de visão de um ou vários dispositivos de controlo. Esta forma dinâmica de atribuição de identificadores faz com

que seja necessário adicionar um LED de luz infravermelha aos dispositivos controladores, e o respectivo receptor aos dispositivos controláveis. O dispositivo de controlo emite periodicamente luz IV. Quando este está direccionado para um dispositivo de controlável, este detecta a radiação e passa o seu ID do valor 0 (estado de repouso) para 255 (espera atribuição de ID). O *control point*, que conhece todos os identificadores atribuídos, atribui, a esse dispositivo um código disponível que identificará univocamente o mesmo.

Os receptores de infravermelhos colocados nos dispositivos podem também ser úteis em termos de eficiência energética do identificador, uma vez que o mesmo pode desligar a emissão de luz até que receba um feixe no receptor. Se o número de dispositivos da rede não exceder o espaço de endereçamento conseguido pelo número de *bits* usado para o identificador, este *hardware* pode ser retirado, sendo apenas necessário atribuir um identificador único a cada dispositivo. Como, durante a fase de descoberta, o *control point* subescreve as notificações vindas deste serviço de identificação, qualquer alteração no código identificador é automaticamente comunicado a todos os *control points* presentes na rede.

Uma vez atribuído o código identificador ao dispositivo, este envia o código binário associado, a uma cadência conhecida, e que será lido pela câmera de infravermelhos. O Wiimote, usado como câmera de infravermelhos nesta solução, envia as coordenadas dos pontos brilhantes ao computador (os LEDs IV, uma vez que possui um filtro), e o *software* calcula quanto tempo cada LED IV esteve aceso ou apagado. Com estes tempos é possível extrair o código binário que identifica o dispositivo.

## 3.4 Localização

A localização precisa dos objectos no espaço é uma condição essencial da Realidade Aumentada.

O primeiro passo da localização é a detecção das coordenadas do feixe proveniente do dispositivo, relativas ao campo de visão da câmera de infravermelhos. A câmera utilizada para a captura da radiação infravermelha é um comando da Wii, o Wiimote. Este comando é também responsável cálculo das coordenadas dos feixes, devolvendo ao ponto de controlo as coordenadas de quatro pontos brilhantes dentro da gama dos infravermelhos.

Outro parâmetro fundamental para se obter a localização é a distância entre o dispositivo de controlo e o controlável. Para se obter a distância, é necessário que o dispositivo a ser controlado possua dois pontos de referência, neste caso, dois LEDs. A distância pode assim ser calculada, conhecendo a distância real dos LEDs e a distância correspondente na imagem. A

distância real a que os LEDs estão colocados pode ser dada por um serviço UPnP, ou ser previamente configurada.

Em termos de requisitos de processamento e capacidade de comunicação, um dispositivo que integre esta solução deve apenas ter capacidades de suportar a arquitectura UPnP – a qual requer capacidades de processar XML e comunicar por IP. Os dispositivos descritos nos perfis da Zigbee Aliance, estão, portanto, fora deste grupo. Para que estes possam integrar esta solução, estes têm de comunicar com uma *bridge*, como a desenvolvida no semestre anterior e que é reponsável pela criação do dispositivo virtual compatível com UPnP. Como a alocação do identificador, neste sitema, é dinâmica, esta pode ser feita por várias alternativas: os dispositivos podem incluir um conjunto de novos comandos que lhe permitam trocar estas mensagens; o identificador pode ser dado pelo endereço de rede de 16 *bits* do dispositivo Zigbee, no caso de se usar uma só rede; usar o endereço IEEE 64 *bits*. Nenhuma destas alternativas é óptima, uma vez que a primeira, a mais eficiente, necessita que os dispositivos passem a incluir comandos que não estão definidos na norma e as restantes aumentam o tempo de detecção do identificador. Uma outra solução é usar uma tabela fixa de endereços, obrigando a configuração prévia do número identificador para esse dispositivo, reservando-o, diminuindo o espaço de endereçamento disponível para os restantes.

### 3.4.1 Disparidade binocular

O erro de disparidade binocular existe devido ao facto de se utilizarem duas câmeras. O campo de visão da câmera sem filtro de infravermelhos é, evidentemente, diferente da câmera de infravermelhos. Esta diferença deve, portanto, ser compensada, para que a sobreposição dos objectos virtuais seja a correcta. Os cálculos relativos à posição das marcas são calculados em relação ao ângulo formado pela direcção em que o objecto é visto e o centro da imagem. Para calcular esse ângulo é necessário aplicar um factor de conversão da resolução da câmera de píxeis para graus.

O erro associado à visão estereoscópica, varia com a distância, motivo pelo qual se têm que utilizar dois LEDs. O ângulo medido do centro da imagem até ao ponto médio entre os dois LEDs, $\alpha$, é, para o eixo horizontal, dado por

$$\alpha_x = \left(\frac{x_1 + x_2}{2} - \frac{R_x}{2}\right) \times \frac{\varphi_x}{R_x} \quad , \tag{3.1}$$

em que $x_1$ e $x_2$ são as coordenadas horizontais dos pontos brilhantes, $R_x$ é a resolução horizontal da câmera em píxeis e $\varphi_x$ o ângulo de abertura da mesma, nessa direcção.

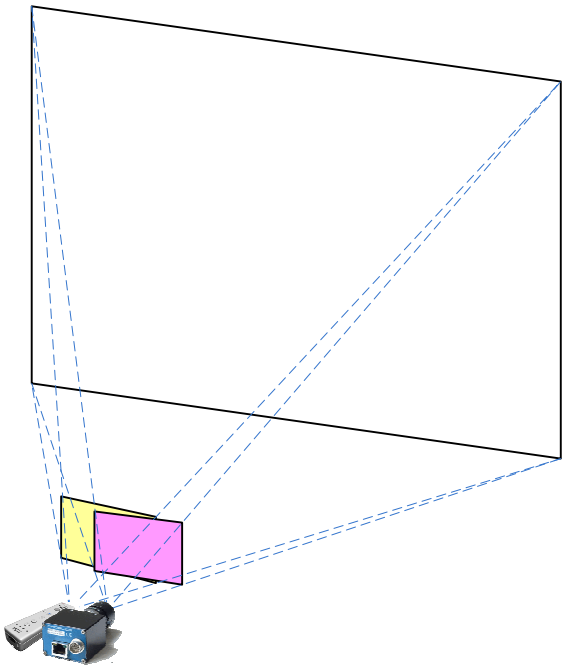

**Figura 3.2** – Disparidade binocular

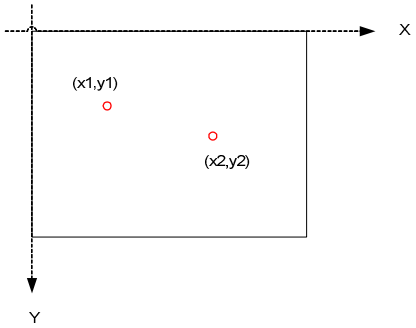


**Figura 3.3** – Campo de visão da câmera de infravermelhos.

A distância ao objecto é, portanto, dada por

$$D = \frac{\frac{d}{2}}{\tan\left(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \times \frac{\varphi/_2}{R}\right)} \quad , \qquad (3.2)$$

em que $x_1$, $x_2$, $y_1$ e $y_2$ são as coordenadas dos pontos, $R$ a resolução, $\varphi$ o ângulo de abertura da câmera e $d$ a distância física entre os LEDs.

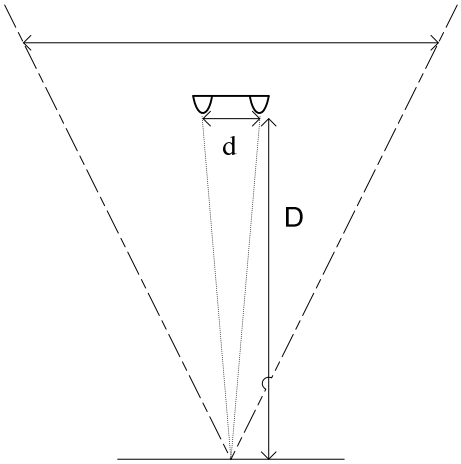


**Figura 3.4** – Variação do ângulo de abertura com a distância física entre os LEDs e a distância entre o dispositivo e a câmera.

Como a câmera pode não ter uma resolução angular igual nas duas direcções, o factor de conversão de píxeis para graus deve ser expresso de forma independente. Reescrevendo, fica finalmente

$$D = \frac{d}{2 \times \tan\left(\sqrt{(x_1 - x_2)^2 \times \left(\frac{\varphi_x}{2R_x}\right)^2 + (y_1 - y_2)^2 \times \left(\frac{\varphi_y}{2R_y}\right)^2}\right)} \quad , \qquad (3.3)$$

sendo $x_1$, $x_2$, $y_1$ e $y_2$ são as coordenadas dos pontos, $R_x$ a resolução horizontal e $R_y$ a resolução vertical da câmera, $\varphi_x$ e $\varphi_y$ o ângulo de abertura da câmera de infravermelhos nas respectivas direcções e $d$ a distância física entre os LEDs.

Sabendo a distância, o ângulo que corrige a disparidade binocular $\beta$, como $\delta \ll D$ é

$$\beta = \sin^{-1}\frac{\delta}{D} \quad , \qquad (3.3)$$

em que $\delta$ é a distância física entre as câmeras e $D$ a distância das câmeras ao objecto.

Caso haja divergência angular das duas câmeras ($\epsilon$), esta deve ser tida em conta, bastando somar ao ângulo calculado, o ângulo feito pelas duas câmeras $\epsilon$, ficando $\theta = \alpha + \beta + \epsilon$ para a horizontal e para a vertical. Os limites para a leitura da marca são dados pelo ângulo de radiação dos LEDs, ou seja, são pouco dependentes da tecnologia.

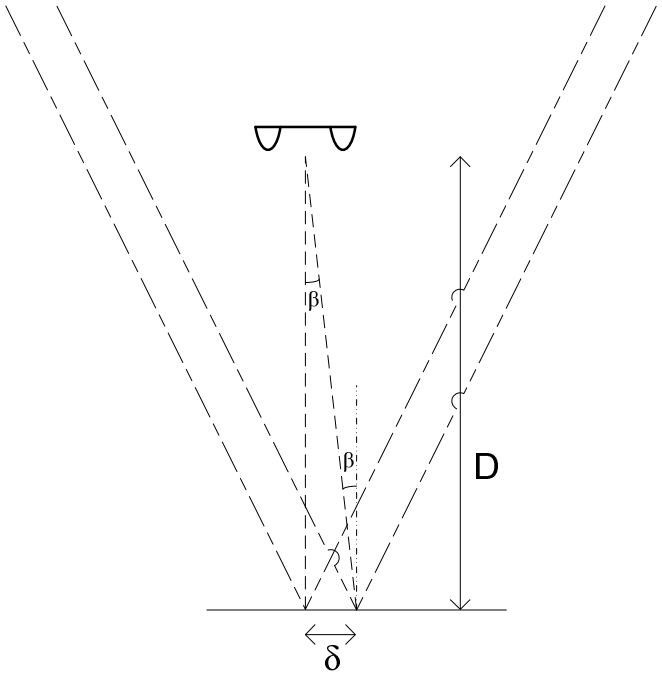


**Figura 3.5** – Ângulo de correcção das coordenadas entre as duas câmeras.

## 3.5 Controlo

A informação trocada pelos diversos dispositivos na rede é independente do sistema operativo usado por cada um, uma vez que toda a informação é serializada e enviada em formato XML. Como a informação sobre um comando é sempre descrita pelo evento associado, apenas o dispositivo necessita de conhecer como traduzir essas descrições em acções. Por exemplo, se o elemento Slider associado a um interface de uma lâmpada passar a ter o seu atributo Value definido em 55, o *control point* notifica a lâmpada desse evento e, o dispositivo de controlo da lâmpada, depois de processar a mensagem recebida, fará actuar a saída de modo a que o valor de intensidade luminosa seja regulada para 55% do seu valor máximo. Isto significa que o dispositivo é o único responsável pelo interface que apresenta e pelo tratamento dos comandos que lhe são enviados.

## 3.6 Conclusão

A solução encontrada tem por base o trabalho anteriormente realizado, onde se usava o UPnP como tecnologia de interligação dos dispositivos e o estudo dos trabalhos referidos na secção 2.7. Isto possibilitou a criação de uma arquitectura suficientemente poderosa capaz de ser aplicável a qualquer tipo de dispositivo com capacidades próprias ou indirectas (através de

uma *bridge*) de comunicação IP. No capítulo seguinte veremos como esta pode ser implementada.

# Capítulo 4

# Implementação

Ao longo das quatro secções deste capítulo, demonstra-se como foi implementada a validação da arquitectura apresentada no capítulo anterior e é, também, apresentado o *hardware* necessário à sua implementação.

## 4.1 Plataforma de desenvolvimento

A escolha da plataforma de desenvolvimento recaiu sobre o pacote Visual Studio 2008 da Microsoft devido à facilidade de programação e integração de qualquer linguagem de programação do pacote as ferramentas da Intel para o *Universal Plug and Play* e da utilização do *wrapper* para C# da biblioteca de Realidade Aumentada. O pacote de ferramentas *Plug and Play* da Intel contém, entre outras, uma ferramenta que permite criar as bases para o controlo da *stack* UPnP em linguagem C#. Esta foi então a linguagem utilizada na implementação dos diversos módulos.

Para o desenvolvimento das interfaces foi usada a recente linguagem XAML, disponível nas *framework* .NET 3.0 e 3.5 da Microsoft. As vantagens do uso desta linguagem serão relatadas na secção 4.2.2.

## 4.2 O dispositivo controlável

Como protótipo de um possível dispositivo integrado nesta rede, foi concebido um circuito electrónico capaz de enviar o código binário de identificação e trocar informações através do protocolo definido na arquitectuta UPnP.

### 4.2.1 O hardware

Sendo necessário alimentar os LEDs e receber o código do computador, optou-se por usar a porta USB como interface, uma vez que o débito é claramente suficiente, está disponível em todos os novos computadores e permite ao mesmo tempo alimentar o dispositivo, dispensando, portanto, a fonte de alimentação.

A tensão de alimentação proveniente da porta USB é de 5V, tensão ideal para o funcionamento de grande parte dos circuitos integrados existentes e suficiente para alimentar os LEDs.

Para se conseguir construir um dispositivo que produzisse um sinal dependente do código identificador e com uma dada cadência, havia três alternativas: utilizar um microprocessador suficientemente rápido para conseguir ler directamente os dados enviados pela porta USB; utilizar um dispositivo que convertesse os dados da USB para, por exemplo, um formato série de mais baixo débito como o largamente usado RS232; utilizar apenas o dispositivo conversor e controlar a cadência a partir do computador. A alternativa escolhida recaiu nesta última, pela facilidade de programação – apenas um código – e pelo tamanho físico do próprio dispositivo.

O protótipo de um possível dispositivo é, portanto, formado pelo computador – responsável pela comunicação IP, pelo processamento das mensagens e pelo controlo do código enviado ao *control point* – e pelo emissor do código binário associado ao dispositivo.

O emissor é constituído por dois LED de infravermelhos, um circuito de regulação do *duty-cycle* e controlado por um módulo de avaliação de um dos famosos dispositivos de conversão USB em série – o FT232R. Este circuito integrado, produzido pela FTDI, é capaz de traduzir os dados recebidos pela porta USB num formato série com um determinado débito, podendo ser controlado, a partir de um computador, através de uma porta de comunicações virtual. O problema do uso deste modo é que não é possível retirar os *bits* de Start e Stop. Contudo, este circuito integrado pode ser usado em diversos modos. A idéia original consistia em usar o módulo em modo série, mas nesse modo são enviados *bits* Start e Stop do protocolo série, o que interferiam no protocolo pretendido. Também foi ponderada a utilização da versão FIFO, o FT245. Estes circuitos já não inserem os *bits* de controlo, mas, após análise com o auxílio de um osciloscípio, verificou-se que o tempo de estabelecimento do estado dos *bits* não obedece à frequência de relógio, invalidando esta opção.

Utilizando o Windows Vista, e sabendo que não se trata de um sistema operativo de tempo-real, o tempo entre operações de escrita na porta USB é de 2 ms (controlado pelo *driver* da porta), o que significa que, para tempos desta ordem de grandeza, é possível controlar o tempo apenas tendo em conta o número de escritas efectuado. Por exemplo, se se pretendesse gerar

uma onda quadrada de 50 Hz, bastaria realizar 5 operações de escrita para a porta USB no estado 0 e 5 operações de escrita com o estado 1. Este foi o método usado para contornar os problemas descritos anteriormente.

Como a folha de características dos LEDs utilizados não está disponível, os valores considerados foram os valores típicos. Empiricamente, sabe-se que o alcance do feixe proveniente dos LEDs é dependente da intensidade da corrente que por eles passa, por isso, o circuito possui um NE555 responsável por regular o período activo dos LEDs, possibilitando a o dimensionamento da corrente para próximo dos valores de pico, mas reduzindo o tempo de condução para alguns micro-segundos (à semelhança do que é feito neste tipo de comunicações). Deste modo, aumenta-se o alcance dos feixes, reduz-se significativamente a potência dissipada nos LEDs prevenindo que estes se danifiquem, e, consequentemente, reduz-se o consumo.

Como são necessários 2 LEDs para se calcular a distância, apenas um dos LEDs é controlável, ficando o outro permanentemente aceso. Isto faz com que o *control point* seja capaz de ter uma referência mesmo quando o nível lógico do *bit* a transmitir seja OFF (e o LED não esteja a emitir), tornando a frequência de actualização apenas dependente das capacidades da câmera de infravermelhos. Apenas a frequência de actualização da distância fica dependente da emissão em simultâneo dos dois LEDs. A codificação do tipo Manchester poderia resolver este problema, uma vez que o LED, usando esta codificação, não ficaria apagado durante um período de tempo superior a um tempo de *bit* e traria as vantagens de se poder eliminar erros de relógio do emissor (a informação da frequência pode ser retirada pelo tempo de transição de estados) e reduziria o código inicializador (bastaria permanecer no estado activo durante um tempo de *bit*), porém, gastaria o dobro das transições para enviar o mesmo número de bits de informação válida.

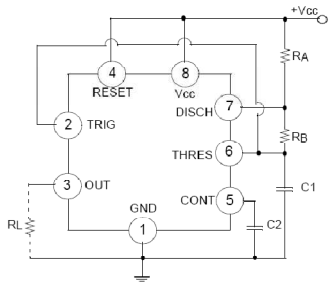


**Figura 4.1** - NE555 na configuração em modo astável [11]

Da folha de características [11] do NE555, pode ver-se que a frequência da onda gerada pelo circuito configurado como um multivibrador astável é dada por

$$f = \frac{1.44}{(R_A + 2R_B)C_1}\,. \tag{4.1}$$

Tendo por base os valores usados nas comunicações por infravermelhos, como por exemplo o RC5, a frequência teria que ser na ordem das dezenas de kHz e o período activo dos LEDs, na ordem dos µs.

Fazendo $C_1 = 22\text{nF}$ e f = 40 kHz, vem que $R_A + 2R_B = \frac{1.44}{f \times 22 \times 10^{-9}} = 1.6\ k\Omega$

Para essa frequência, o período é 25µs. Para que o período activo seja de alguns micro-segundos considerou-se o *duty-cycle* inferior 20%, portanto, $R_A = 4R_B$ logo $6R_B = 1.6\ k\Omega$, ou seja, $R_B = 267\ \Omega \cong 220\ \Omega$ e $R_A = 1600 - 2 \times 220 = 1160\ \Omega \cong 1.2\ k\Omega$. Neste caso

$$f = \frac{1.44}{(1.2k + 2 \times 220) \times 22n} \cong 40\ \text{kHz}\ , \tag{4.2}$$

com *duty-cycle* de $\frac{220}{1.2k + 220} \cong 15\%$.

Para as correntes nos LEDs infravermelhos, foi escolhida uma corrente de pico superior a 100mA. Exigindo uma corrente dessa grandeza à saída do NE555, pode verificar-se, consultando a folha de características do circuito, a tensão de saída para o estado *low* é superior a 0.75 V (valor obtido a 50 mA). Considerando $\beta \gg 1$, como

$$i_{LED1} = i_c \cong i_e \cong 100\ mA = \frac{5 - 0.75}{R} \quad , \tag{4.3}$$

vem que $R < 43\ \Omega \cong 39\ \Omega$. Nessas condições, a corrente no LED é

$$i_{LED1} < \frac{5 - 0.75}{39} = 109\ mA. \tag{4.4}$$

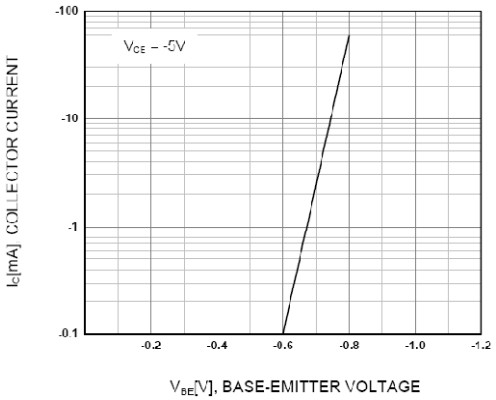


**Figura 4.2** – Variação da tensão $V_{BE}$ com a corrente no colector. [11]

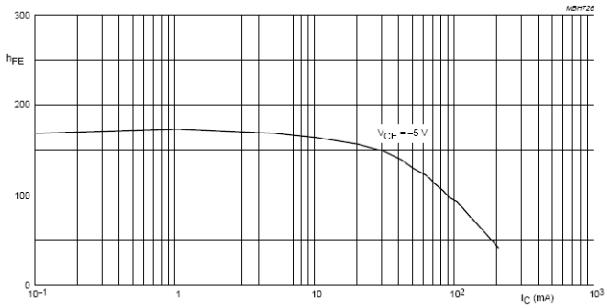


**Figura 4.3** – Variação do parâmetro $h_{FE}$ com a corrente no colector. [11]

Consultando as folhas de características dos transistores utilizados, o BC558 e BC547, pode-se considerar $V_{BE} \cong -0.8$ V e β≅100, como

$$i_{LED2} = i_C = \beta \times i_B \quad \text{vem que} \tag{4.5}$$

$$i_B = \frac{100}{100} = 1 \text{ mA.} \tag{4.6}$$

Como a base do transístor BC547 está ligado ao módulo do FT232R, consultado a folha de características, verifica-se que está abaixo do seu limite de 24 mA.

| Parameter | Description | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| Voh | Output Voltage High | 3.2 | 4.1 | 4.9 | V | I source = 2mA |
| Vol | Output Voltage Low | 0.3 | 0.4 | 0.6 | V | I sink = 2mA |
| Vin | Input Switching Threshold | 1.3 | 1.6 | 1.9 | V | ** |
| VHys | Input Switching Hysteresis | 50 | 55 | 60 | mV | ** |

**Figura 4.4** – Tensão no pino de saída do NE555. [11]

O LED controlável, estará a emitir quando a base do transístor NPN estiver a 1 e a saída no NE555 a 0. É necessário ter em conta, mais uma vez a tensão de saída do pino 3 quando os LEDs conduzem, que será superior a 0.75V. Durante a condução, é exigida à porta do módulo uma corrente de cerca de 1mA. Consultando a folha de características, verifica-se que a tensão de saída típica é de 4.1V para 2mA. Neste caso, tensão deverá aproximar-se um pouco mais da tensão de alimentação, e, por isso,

$$R_B > \frac{4.1-0.75-0.8}{1\times10^{-3}} = 2.55\ k\Omega \quad . \tag{4.7}$$

No caso limite, pode acontecer que a tensão à saída do módulo se aproxime de 5V, enquanto a tensão de saída do NE555 será, de certo, superior a 0.75V, portanto, no pior caso

$$R_B < \frac{5-0.75-0.8}{1\times10^{-3}} = 3.45\ k\Omega \tag{4.8}$$

Optou-se por garantir que a corrente de pico do LED não é excedida, optando-se, por isso, pelo valor mais alto do intervalo, ou seja, $R_B = 3.3\ k\Omega$

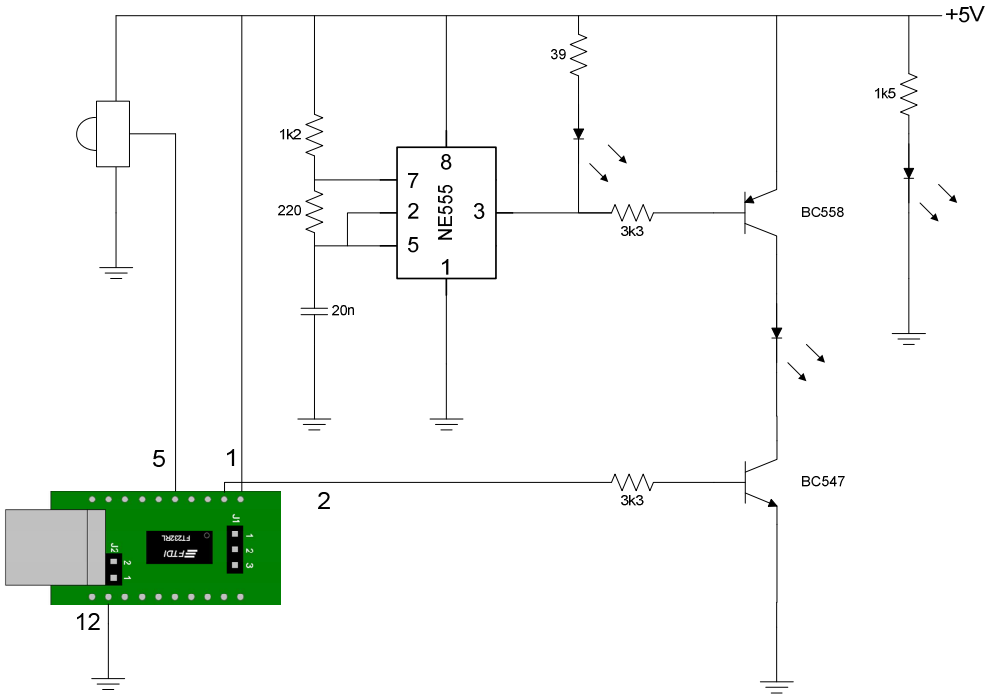


**Figura 4.5** – Esquema eléctrico do dispositivo controlável.

Por inspecção, a resistência que dissipa mais potência é a que se encontra em série com o LED que está sempre activo. A potência média dissipada por essa resistência de 39 Ω é 18% (período activo) da potência dissipada quando o LED acende, ou seja,

$$P = R \times i_{LED1}^2 \times dutycycle = 39 \times 0.109^2 \times 15\% \cong 70 \text{ mW} < 250 \text{ mW}, \qquad (4.9)$$

o que significa que se podem utilizar resistências de ¼ de Watt.

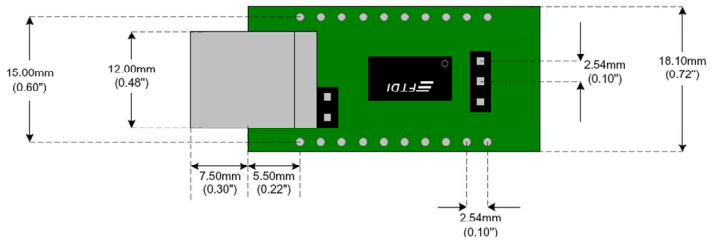


**Figura 4.6** - Dimensões do módulo da FTDI. [11]

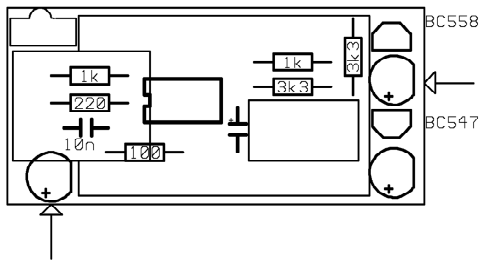


**Figura 4.7** – Circuito projectado visto de topo. As setas indicam os LEDs IV.

Quanto às dimensões físicas da placa de circuito impresso, estas foram optimizadas para se obter um tamanho reduzido. O circuito foi, portanto, colocado por baixo, como uma base, onde encaixa o módulo do circuito da FTDI. Este módulo controla a base do NPN e fornece ao circuito modulador a alimentação proveniente da porta USB, configurada através dos *jumpers* para 5V. Para minimizar o erro no cálculo da distância entre o utilizador e o dispositivo, a distância entre os LEDs infravermelhos foi maximizada. Para isso foram colocados na diagonal do circuito, tendo sido arredondada para 4 cm. Ao circuito, foi também adicionado um condensador de filtragem, para diminuir a influência das correntes de pico exigidas pelos LEDs ao barramento da porta USB. A corrente de pico, é, com esta montagem, a soma das correntes dos dois LEDs, sendo, portanto superior a 200 mA.

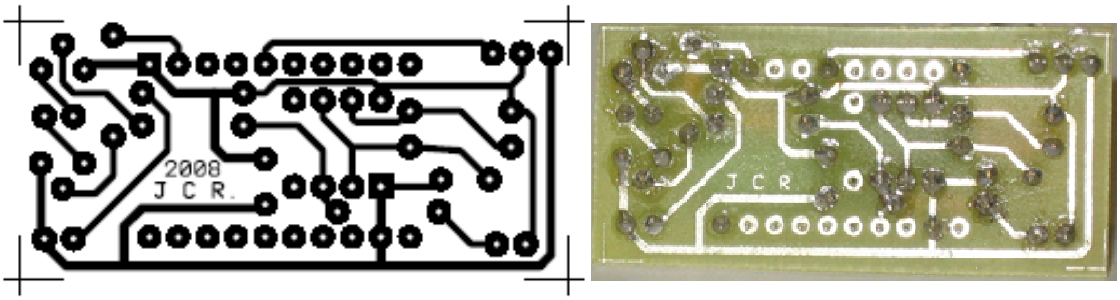


**Figura 4.8** – A placa de circuito impresso projectada (à esq.) e concebida (à dir.).

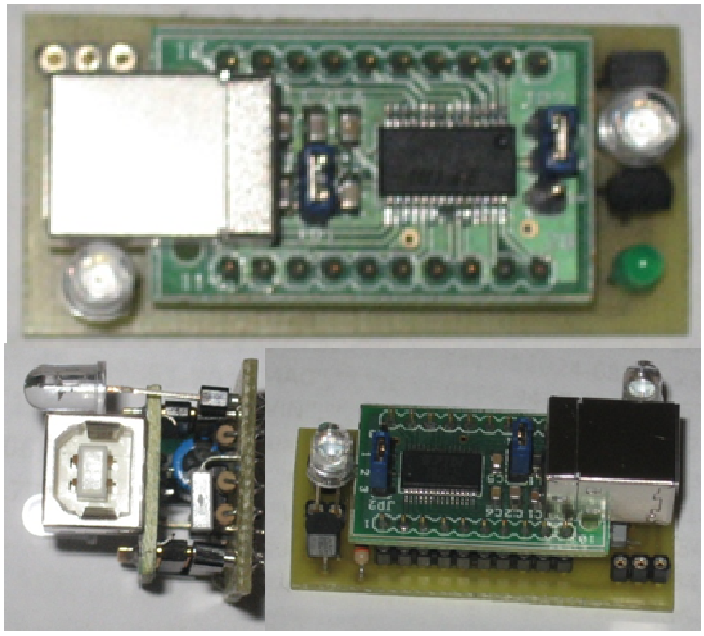

**Figura 4.9** – As várias vistas do circuito construído.

### 4.2.2 O software

Para a criação dos serviços UPnP foi utilizado o pacote de ferramentas disponibilizado pela Intel. Neste caso usou-se, em primeiro lugar, o Service Author, para a criação dos dois serviços que os dispositivos devem suportar. De seguida, foi usado o Device Builder. Esta aplicação é capaz de gerar o código-base para a implementação da *stack* UPnP em linguagem C#.

O dispositivo criado foi baseado no exemplo de uma lâmpada com controlo de intensidade (*dimmable light*), a qual já inclui os serviços SwitchPower para ligar e desligar e o serviço DimmingService que controla a intensidade luminosa da mesma.

A essa lâmpada foram adicionados os dois serviços necessários para a integração dessa nesta arquitectura: o serviço IDManagerService e o ProcessEvents.

O primeiro é responsável pela gestão do identificador. Os comandos disponíveis são o GetID (devolve o identificador actual) e o SetID (atribuição de novo identificador). Este último, altera a variável de estado ID, o que leva a que todos os nós que subescreveram este serviço (neste caso, todos os *control points* desta arquitectura) a serem notificados do novo código identificador deste dispositivo. Isto não seria possível fazer-se recorrendo a um *webbrowser*. Nesse caso, teria de ser o *control point* a pedir periodicamente ao dispositivo que lhe reportasse as suas variáveis de estado, desperdiçando recursos.

O serviço ProcessEvents, fornece aos *control points* a descrição em linguagem XAML do interface actual do dispositivo através da acção GetXAML, ou então, através da informação do último evento disponibilizado na variável de estado LastEvent.

A informação contida na informação de evento está descrita em XML e tem a seguinte estrutura:

```
<Event>
        <Changes ObjectName="nomeObjecto" Event="nomeEvento">
        ...
        </Changes>
        <XAMLChanges>
                <Canvas xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                Tag="ea676a60-e8c9-42af-a26e-2512c8bcfb81">
                ...
                </Canvas>
        </XAMLChanges>
</Event>
```

Daqui é possível extrair informação acerca de qualquer tipo de evento, uma vez que na construção do XML se usa reflexão, ou seja, todas as propriedades do evento são serializadas,

tornando o código extremamente flexível. Esta é a parte do documento recebido que o dispositivo necessita de interpretar, para saber que implicações este evento terá nas suas variáveis de estado.

Dentro do elemento XAMLChanges é descrita a interface. O elemento raiz deve especificar os espaços de nomes (*namespaces*) associados aos elementos utilizados. O atributo Tag do elemento raiz contém o GUID do *control point* que responsável pela última alteração. Este atributo é útil para que o *control point*, quando recebe um novo interface indicado por ele mesmo, não voltar a refazer o interface, poupando tempo de processamento.

Depois de o dispositivo receber e processar o XML, altera as suas variáveis de estado e reflete estas alterações em acções físicas, como o aumento de intensidade de ventilação ou aquecimento.

Quanto à gestão do código binário enviado, o *software* desenvolvido envia esse código encapsulado como apresentado na figura 4.10.

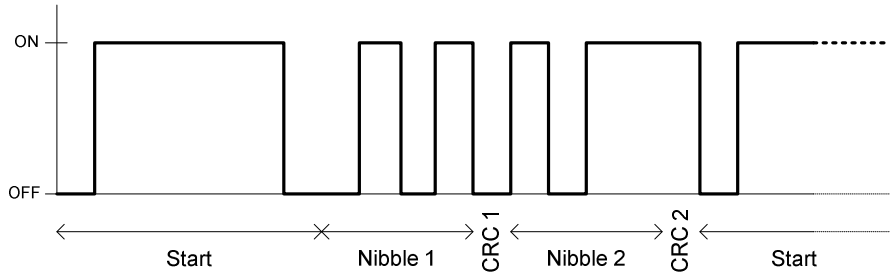


**Figura 4.10** – Trama enviada pelo dispositivo identificador através de IV.

Inicialmente o LED responsável pelo envio do código é desligado durante um tempo de *bit* e em seguida brilha durante um período superior a 5 *bits*, correspondendo a um início de trama. Para marcar o fim do código, o estado passa a OFF durante um tempo de *bit*. Em seguida são enviados dois *nibbles* (4 *bits*). O *nibble* mais significativo do código é enviado, seguido do CRC desses 4 *bits*. Este CRC é calculado pelo XOR destes 4 primeiros *bits*. Em seguida é enviado o segundo grupo de 4 *bits* do código e o CRC correspondente calculado pela mesma forma do primeiro. A inserção dos códigos CRC na trama tem duas funções importantes. A primeira é verificar se a paridade dos *bits* recebidos está correcta e a segunda função é limitar o número de *bits* consecutivos a 4, garantindo que uma trama recebida correctamente, não contém o código de início de trama no espaço reservado ao código. Repare-se que só se poderia obter 5 estados a 1, se um dos *nibbles* fosse 1 1 1 1, porém o CRC deste *nibble* daria 0, reduzindo o número máximo de 1's consecutivos para 4.

## 4.3 O dispositivo de controlo

O dispositivo de controlo é composto por um computador e duas câmeras, sendo uma delas uma *webcam* e a outra o Wiimote. O *software* de controlo integra a vizualização da imagem real, e sobrepõe os interfaces dos dispositivos que vão aparecendo no campo de visão quer da *webcam* (através das marcas suportadas pela ARToolkitPlus) quer do Wiimote (envio do código através de luz infravermelha).

Em termos de conectividade, o *software* é capaz de comunicar com os dispositivos por UPnP através de qualquer interface com suporte IP disponível na máquina. A *webcam* utilizada pode ser qualquer uma suportada pelo sistema operativo e a comunicação com o Wiimote é feita através de Bluettooth usando a WiimoteLib.

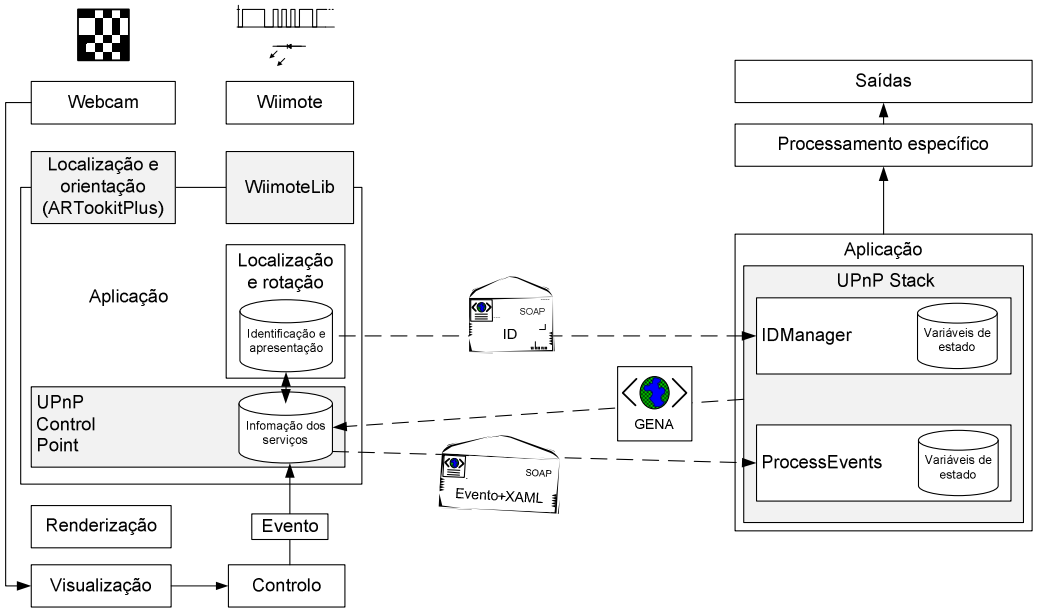


**Figura 4.11** – Visão geral da solução.

## 4.4 Interface gráfico

O controlo dos dispositivos, nesta solução, tem por base a flexibilidade. Esta solução separa o interface gráfico do código associado aos comandos.

O interface é descrito em XAML, podendo-se usar as cada vez mais numerosas ferramentas de edição. A regra é que o elemento raiz deste XAML seja do tipo Canvas, isto por se tratar de um elemento do tipo Panel Control (responsável pelo *layout*) e porque este será inserido numa janela, o que exclui o elemento Window como possível elemento raiz (o elemento Window tem que ser sempre o elemento raiz em WPF).

Este documento pode incluir informações gráficas adicionais como animações, mas não pode conter código associado. Esta é uma característica importante, uma vez que impede que um dispositivo corra no *control point* código nocivo. A estas informações gráficas apenas se podem adicionar os atributos correspondentes a eventos de um elemento, como por exemplo Click, a que corresponde o clique do rato no elemento. Estes eventos, informam os *control points* dos eventos que necessitam de ser comunicados ao dispositivo, sendo, os restantes, ignorados. Neste caso, o evento basta ser declarado, não nessessitando de especificar o nome da função responsável pelo tratamento do mesmo. Para que o *control point* seja notificado quando o valor de um Slider foi alterado basta apenas referir o evento ValueChanged:

```
<Slider Name="slider1" ValueChanged="" SmallChange="1" Maximum="100" />.
```

Outra vantagem do WPF é conseguir renderizar o interface gráfico sem que seja necessário haver um fundo opaco. Esta é uma enorme vantagem em relação aos *webbrowsers*, porque possibilita a criação de interfaces que se integrem melhor com a imagem real. O WPF possibilita, ainda, a visualização dos interfaces num ambiente tridimensional. Como na solução apenas são usados dois LEDs, não é apenas possível obter informação acerca da rotação ao longo do eixo que une o dispositivo à câmera de infravermelhos. Para que fosse possível comparar a solução das marcas usadas pela ARToolkitPlus com os LEDs infravermelhos e para demonstrar o poder gráfico do WPF, o *control point* desenvolvido integra, também, a biblioteca ARToolkitPlus, passando a haver dois tipos de marcas que se podem usar para que o interface seja apresentado. É obvio, que estas marcas, ao contrário das que utilizam LEDs, são fixas, não permitindo a alocação dinâmica de códigos. Estes devem ser, portanto, pré-configurados.

O interface da aplicação de controlo permite ajustar os desvios entre a *webcam* e o Wiimote, bastando para isso deslocar os Sliders e permite activar e desactivar a as funcionalidades da ARToolkitPlus, uma vez que estas são extremamente exigentes a nível de processamento e são, neste caso, opcionais.

O interface enviado pelos dispositivos, em formato XAML, necessita de ser processado. Para isso, existe a função XAMLWriter.Load que aceita como argumento um *stream* que contém o XAML do interface e devolve o respectivo objecto. Esta função não processa a associação de eventos e, por isso, é necessário, em primeiro lugar, remover todos os atributos referidos XAML recebido. A lista de eventos foi recentemente publicada em formato XSD, pela Microsoft. O programa, ao iniciar, faz uma *query* em LINQ a esse ficheiro de especificações e guarda em memória todos os eventos definidos na norma. Depois, quando recebe um XAML, retira esses atributos para poder usar a função XAMLWriter.Load. Em seguida, cada evento de

cada controlo do interface é associado à função de tratamento de eventos. Esta função é apenas uma, estando apenas definidos vários *overloads*, para que, dependendo do tipo de evento gerado, a função responsável por esse evento "exista". A função de tratamento dos eventos é, depois, usando reflexão, capaz de olhar para as propriedades do evento gerado e construir o XML que descreva adequadamente esse evento. O código torna-se, assim, muito flexível, suportando qualquer evento definido na norma, sem ser necessária a criação de funções específicas.

A renderização dos objectos virtuais é feita aplicando as capacidades do WPF. Como, usando 2 LEDs, apenas se pode retirar um ângulo de rotação, o interface é rodado dependendo do ângulo formado pelos LEDs. Em WPF, basta especificar o ângulo como um atributo de um elemento de transformação.


**Figura 4.12** – Sobreposição do interface recebido, no interface do dispositivo de controlo.

Quanto à integração da ARToolkitPlus, foi criado um interface demonstrativo com maiores capacidades de renderização do que o anterior, uma vez que é possível saber os ângulos de rotação nos 3 eixos (a solução anterior teria de utilizar 4 LEDs para obter o mesmo resultado). A renderização de objectos 3D não é simples, contudo, esta foi também implementada, por se tratar de uma característica bastante interessante, uma vez que torna-se possível representar o interface visto de qualquer direcção, justapondo-se à imagem real de um modo mais realístico. Para que isso fosse possível foi necessário criar um elemento Viewport3D e nos seus recursos especificar a malha de pontos tridimensionais que constituem o objecto. É necessária, também, a configuração do tipo de material que constitui o elemento e a caracterização das luzes de

iluminação da cena, para que as reflexões de luz sejam mais aproximadas à realidade. Por fim é necessário, ainda, indicar todos os parâmetros relativos à câmera, como a sua localização, direcção e ângulo de abertura. Felizmente, em WPF existe o elemento Viewport2DVisual3D que permite o mapeamento de elementos de interface gráficos 2D em 3D. Dentro deste pode, então, ser descrito o interface gráfico da forma normal, contendo todas as funcionalidades oferecidas pela WPF.

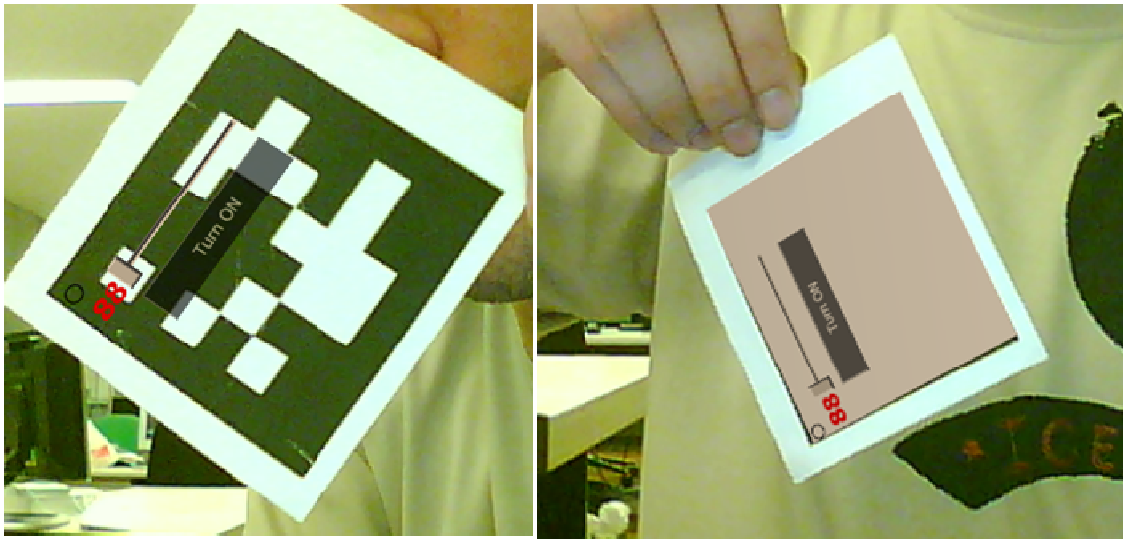


**Figura 4.13** – Integração da ARTookitPlus na aplicação. À esquerda com fundo transparente e á direita com fundo opaco.

# Capítulo 5

# Avaliação do trabalho

Neste capítulo são descritos os vários cenários de teste. Em seguida são apresentados os resultados relativos a esses testes. O capítulo termina com a avaliação dos mesmos, comparando os dois métodos de identificação de objectos.

## 5.1 Cenários de teste

Para a realização dos testes foi utilizado um computador portátil com as seguintes características: processador Intel Core2Duo T5250 @ 1.50 GHz, com 2GB de memória RAM e com o sistema operativo Windows Vista.

A *webcam* utilizada foi configurada para uma resolução de 640x480 e a *framerate* configurada para o seu máximo, 30 FPS. Note-se que a *framerate* especificada pode não ser a verificada realmente, uma vez que esta depende das condições de iluminação da sala, sendo controlada pelo controlador da mesma, afim de aumentar o tempo de exposição em imagens demasiado escuras. A sala onde os testes foram efectuados encontrava-se iluminada apenas com luz artificial proveniente do tecto da divisão, fornecida por lâmpadas de descarga. A câmera foi convenientemente ajustada para fazer face à cintilação provocada pela frequência de 50 Hz da rede.

O código binário escolhido para ser enviado pelo dispositivo controlável foi o 91 em todos os testes, enquanto para a marca de papel foi usado o código 481.

### 5.1.1 Validação do hardware utilizado

Os primeiros testes foram efectuados para validar o dimensionamento eléctrico do circuito desenvolvido para a emissão dos códigos de luz infravermelha. O dispositivo controlável foi ligado à porta USB do computador e correu-se a aplicação de controlo.

Para a visualização dos sinais foi utilizado um osciloscópio digital com largura de landa de 40MHz e uma frequência de amostragem máxima de 100MS/s. Os resultados foram retirados através da aplicação de suporte do osciloscópio.

### 5.1.2 Leitura dos códigos infravermelhos

Para o cálculo do erro de identificação através de infravermelhos foi utilizado o computador com um *dongle* Bluetooth para comunicar com o Wiimote. Foi adicionada informação de *debug* no instante do reconhecimento das marcas, indicando se a mesma tinha sido reconhecida com sucesso. Esta informação foi redireccionada para um ficheiro para posterior tratamento. Depois de adicionar o Wiimote ao sistema, as duas aplicações desenvolvidas (a de controlo e a controlável) foram postas a correr.

Na primeira fase, o identificador foi colocado sobre uma mesa, a 1 metro do utilizador que segurava o Wiimote apontado-o para a marca. Foram realizados testes com a duração de 60 segundos para diferentes tempos de *bit*, e os resultados da detecção foram guardados em ficheiros. A alteração do tempo de *bit* necessitou de pequenos ajustes no número de escritas na porta USB, uma vez que para além dos 2 ms do tempo de escrita é necessário ter em conta o tempo de processamento para que essa seja efectuada. Este ajuste foi efectuado com o auxílio do osciloscópio.

Na segunda fase de testes, fixou-se o tempo de *bit* em 40ms e variou-se a distância entre o utilizador e a marca. O utilizador, permaneceu estático durante os sessenta segundos de cada teste. Os resultados foram guardados, mais uma vez, em ficheiros para posterior tratamento.

Na terceira fase de testes, foi fixado o tempo de *bit* em 40ms e a distância do utilizador aos LEDs em 1 metro. Variou-se, então, o ângulo com que a marca era vista pelo Wiimote e guardaram-se os resultados obtidos durante um minuto para cada distância.

Para finalizar, foi observado e registado o processamento médio da aplicação de controlo, através do Monitor de Recursos do sistema operativo.

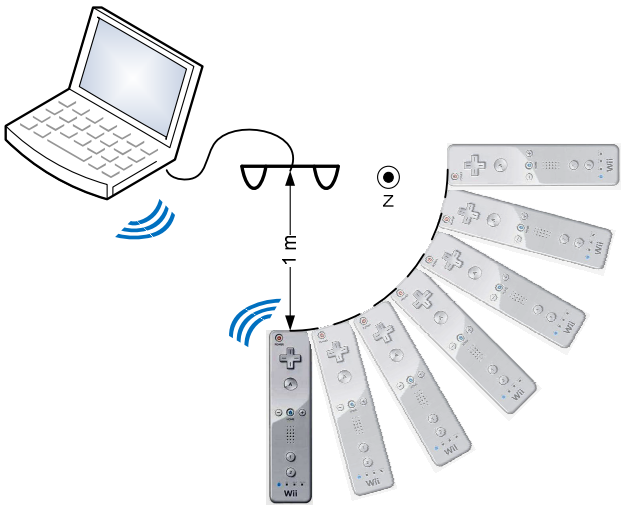


**Figura 5.1** – Cenário de teste da probabilidade de erro com a variação do ângulo de visão do identificador de infravermelhos.

### 5.1.3 ARToolkitPlus

Para o teste da solução de identificação de marcas já existente, foi utilizada a marca 481 com dimensões 7.9 cm x 7.9 cm, impressa a *laser* num papel branco.

Numa primeira fase, esta foi colocada a várias distâncias e foi determinado, através da adição de informação de depuração na aplicação criada, o instante em que se efectuou uma detecção. A sala onde o teste foi efectuado foi a mesma do cenário anterior, conservando, aparentemente, as mesmas condições de iluminação. A marca permaneceu estática durante cada minuto de teste.

Na segunda fase de testes, a marca foi colocada a uma distância de 1 metro da câmera e rodada, segundo o eixo vertical, com diferentes inclinações. Cada teste teve, mais uma vez, a duração de um minuto e os seus resultados foram guardados num ficheiro de texto.

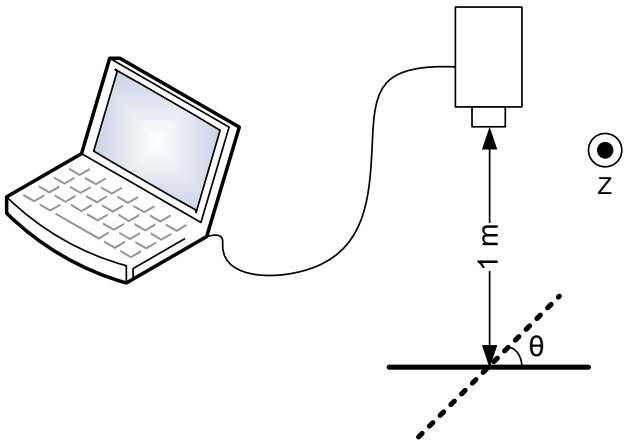


**Figura 5.2** – Cenário de teste para a obtenção da probabilidade de erro associada ao ângulo com que a marca de papel é vista.

## 5.2 Resultados

### 5.2.1 O hardware

Nos teste ao *hardware* foram analisadas diversas formas de onda que permitem calcular alguns parâmetros importantes. A frequência da onda foi calculada pelo osciloscópio, tendo dado 37.43 kHz. Isto significa um erro relativo de 6,4% em relação ao valor para o qual foi dimensionado. Verificou-se também que o período de condução do LED é de 5us correspondendo a um *duty-cycle* de 18.715%. Esta informação pode ser retirada a partir da Figura 5.3.

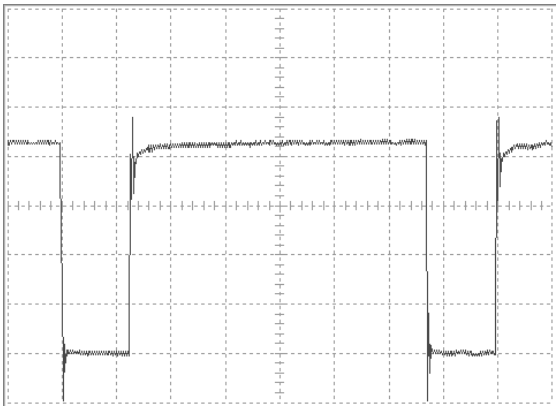

**Figura 5.3** - Tempo de condução do LED (4 ms/div; 1 V/div).

Quanto ao tempo de bit, após compensar o tempo de processamento, obteve-se, para um período de 40 ms a forma de onda representada na Figura 5.4.

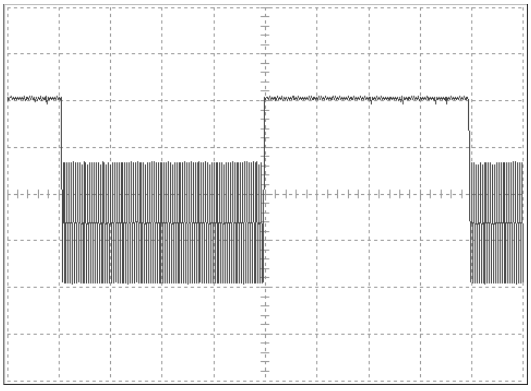

**Figura 5.4** - Forma de onda obtida no cátodo do LED, enquanto é enviado um código. (10ms/div; 1V/div)

Para além destes parâmetro, verificou-se que a tensão de saída do NE555 quando está no seu estado “0” é de cerca de 1,6 V (a 67 mA), valor muito superior ao especificado na folha de

características do circuito para 50 mA e que serviu de referência para o dimensionamento do circuito.

### 5.2.2 Leitura por infravermelhos

Na Figura 5.5 pode verificar-se que quanto maior é o tempo de *bit*, menor é o erro relativo associado. Apesar de, com um tempo de bit de 40 ms se enviarem menos tramas, como estas são mais vezes bem detectadas, o número de tramas correctamente lidas é superior ao obtido para 30 ms. Verifica-se que um tempo de 20 ms por bit é claramente insuficiente para se obter uma leitura adequada.

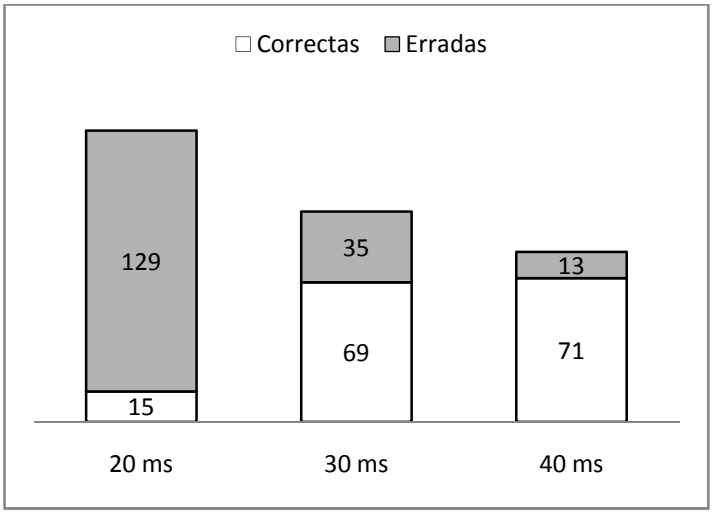


**Figura 5.5** – Número de leituras correctas e erradas para diferentes tempos de bit.

De seguida procedeu-se à fixação do tempo de bit em 40 ms. Os resultados estão expressos no gráfico da Figura 5.6, onde se mostra que a variação da taxa de erros, à medida que o utilizador se afasta da marca, não é significativa. Dos resultados pode ainda saber-se que, para este circuito, o seu alcance máximo é inferior a 5 metros.

Para a avaliação da capacidade de leitura do código identificador sem que os dispositivos se encontrem frente-a-frente, variou-se o ângulo com que o mesmo é visto, mantendo-se a distância em um metro. Do gráfico da Figura 5.7 pode-se verificar que esse ângulo afecta fortemente a capacidade de leitura, pelo que, a partir de 45º, esta torna-se impossível.

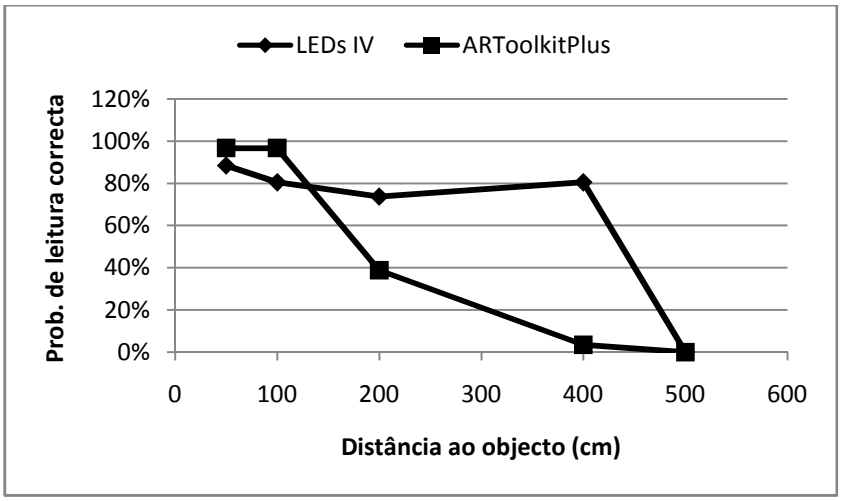


**Figura 5.6** – Probabilidade de leitura correcta em função da distância.

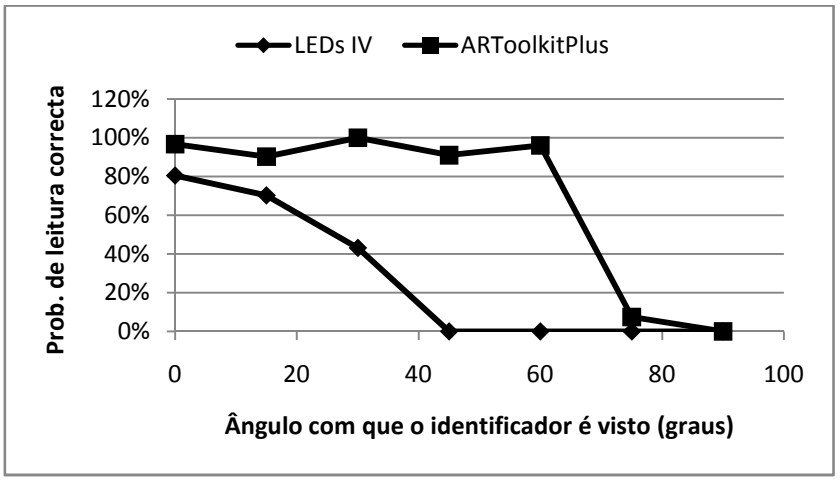


**Figura 5.7** – Probabilidade de leitura correcta variando o ângulo.

### 5.2.3 Leitura através da ARToolkitPlus

Em primeiro lugar é necessário esclarecer que, como o número de leituras máximo depende do *hardware*, foi considerado o máximo obtido em todos os testes realizados (valor obtido a uma distância de um metro e 30º de inclinação)..

Através da análise do gráfico da Figura 5.6, pode verificar-se que até uma distância de 1 metro, a identificação da marca é praticamente infalível. Para distâncias superiores, esta forma de identificação começa a revelar taxas de erro bastante elevadas. Aos quatro metros de distância ainda se torna possível a detecção esporádica da marca, sendo impossível detectá-la a 5 metros.

Quanto ao efeito do ângulo com que esta é vista pela câmera, os resultados mostram a grande capacidade de leitura do código identificador nessas condições. De notar, que com uma

inclinação de 60º, a leitura é feita correctamente em 96% dos casos. Com uma inclinação de 75º ainda é possível a detecção esporádica da marca, sendo, obviamente, impossível detectar com uma inclinação de 90º.

## 5.3 Avaliação

No dimensionamento do circuito emissor de luz infravermelha, devido à falta de informação da folha de características do NE555, foi considerado que o valor da tensão à saída do pino 3, durante o período de condução, seria 0,75 V. Pôde verificar-se que este valor sobe bastante com a corrente fornecida, o que levou a uma corrente menor nos LEDs. Este facto, afectou a distância máxima conseguida por esta solução. O circuito, deveria, assim, ser refeito, adicionando um transístor ao ramo do LED que está em série com a resistência. A tensão de saturação entre a base e o colector de um transístor bipolar como os utilizados para o outro LED, é bastante inferior, o que possibilitaria a passagem de uma corrente maior em ambos os LEDs. Com esta alteração espera-se que a distância máxima a que o feixe é detectado passe a ser superior.

Comparando os dois métodos de identificação pode verificar-se que a leitura correcta de luz infravermelha não é muito dependente da distância, como acontece com a ARToolitPlus, porém, estas podem ser lidas com inclinações elevadas.

Quanto ao processamento, este pode ser verificado na Figura 5.8. Como se pode constatar, o processamento exigido para a detecção de marcas através da imagem real requer elevado poder de processamento, enquanto na detecção por IV, as necessidades de processamento são desprezáveis.

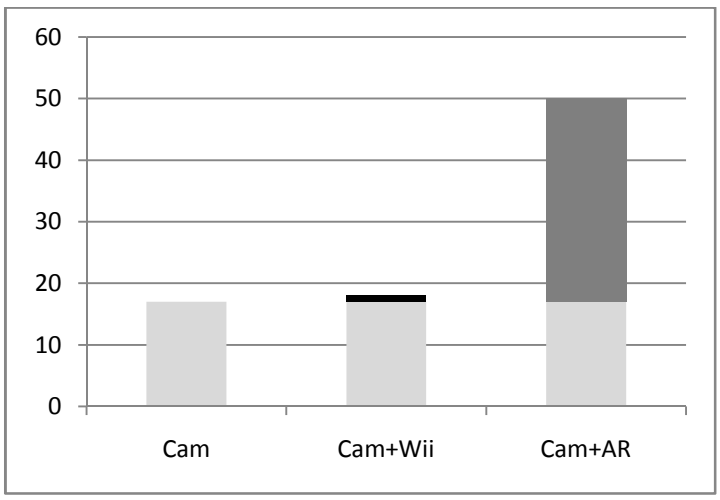


**Figura 5.8** – Comparação da percentagem de processamento exigido.

As tabelas 5.1 e 5.2, resumem as características das marcas utilizadas pelas duas tecnologias. Analisando-as podemos ver que as vantagens das marcas de papel é serem mais simples, não consumirem energia e fornecerem informação sobre o posicionamento em 3 eixos, mas, apesar de possuirem um espaço de endereçamento superior, não permitem alocação dinâmica de identificadores, requerendo, portanto, uma configuração prévia. As vantagens da utilização de identificadores por IV é que estes são passíveis de serem detectados na ausência total de luz ambiente. Esta é uma característica importante, quando se trata de controlar os dispositivos domésticos como os responsáveis pela iluminação.

Quanto ao dispositivo de controlo, verifica-se na Tabela 5.3 que as marcas de papel apenas necessitam de uma câmera de infravermelhos, mas, em contra-partida, é necessário um alto poder de processamento, o qual aumenta quando se usam câmeras com maior resolução.

**Tabela 5.1** – Comparação das marcas usadas pela ARToolkit e pela solução de IV.

| Tipo | Necessita controlo | Eixos | Potência em *standby* | Potência modo activo | Endereçamento |
|---|---|---|---|---|---|
| AR | Não | 3 | 0 | 0 | 4096 (fixos) |
| IV | Sim | 1 | ≅0 | ≅200 mW | 254 (dinâmicos) |

**Tabela 5.2** – Comparação da dependência em relação à distância e ao ângulo na leitura correcta dos dois tipo de identificadores.

| Tipo | Dependência da distância | Dependência do ângulo |
|---|---|---|
| AR | Elevada | Reduzida |
| IV | Reduzida | Elevada |

**Tabela 5.3** – Comparação das necessidades do dispositivo de controlo consoante o método de identificação.

| Tipo | Núm. de câmeras | Detecção sem luz | Processamento necessário | Dependência processamento/resolução |
|---|---|---|---|---|
| AR | 1 | Não | Elevado | Elevada |
| IV | 2 | Sim | Reduzido | Reduzida |

# Capítulo 6

# Conclusões e trabalho futuro

Este capítulo faz um resumo dos principais elacções retiradas na realização deste trabalho, sugerindo-se em que sentido este deve evoluir.

## 6.1 Revisão do trabalho desenvolvido

A Realidade Aumentada, terá concerteza, no futuro, um papel determinante na criação de interfaces mais simples e intuitivas. A vulgarização desta tecnologia permitirá métodos de controlo melhor adequados aos cada vez mais complexos dispositivos que invadem os nossos lares. A contenção de custos de produção e de energia, despertará um cada vez maior interesse na diminuição do número de *displays* necessários ao controlo ubíquo, o que levará ao alargamento deste tipo de técnicas.

A arquitectura proposta, é um passo em frente na concentração do controlo num só dispositivo, permitindo a flexibilidade suficiente para se adaptar a qualquer contexto. Apesar de requerer uma ligação IP, o UPnP é adequado ao controlo de dispositivos de uma forma genérica e descentralizada.

Quanto à utilização de marcas que recorrem ao envio de luz IV, estas obrigam a inclusão de duas câmeras no dispositivo de controlo e apenas são detectáveis quando o utilizador está quase de frente para as mesmas. Em comparação com as marcas de papel estas são mais adequadas para este fim, uma vez que as suas dimensões são muito menores para se obter o mesmo alcance, não necessitam de luz ambiente e o processamento necessário à sua detecção é, comparativamente, reduzido.

## 6.2 Contribuições relevantes

Este trabalho sugere um método de controlo de equipamentos electrónicos, nomeadamente através de uma arquitectura validada através da sua implementação, que permite, ao utilizador, dentro do contexto habitacional, identificar e controlar os dispositivos presentes no seu campo de visão.

## 6.3 Trabalho futuro

A evolução do trabalho aqui apresentado passará pela expansão do número de pontos de referência dos identificadores, possibilitando retirar informação acerca da rotação em torno dos três eixos. Isto fará com que se possa aprofundar a informação acerca da localização relativa entre o utilizador e os dispositivos e mesmo entre dispositivos, possibilitando uma oferta de serviços alargada e que, informando os sistemas de domótica dessas localizações relativas, os possa tornar mais eficientes.

# Referências


[1] Pilha protocolar UPnP. Disponível em http://technet.microsoft.com/en-us/library/bb457049(TechNet.10).aspx. Acesso em 24/Maio/2008.

[2] S. Cawood, M. Fiala, "*Augmented Reality – A Practical Guide*", Pragmatic Bookshelf, Jan. 2008, pp. 1-27.

[3] ARToolkit. Disponível em http://www.hitl.washington.edu/artoolkit/. Acesso em 8/Junho/2008.

[4] Handheld Augmented Reality. Disponível em http://www.icg.tu-graz.ac.at/Members/daniel/HandheldAR_Thesis. Acesso em 29/Junho/2008.

[5] Introdução ao XAML. Disponível em http://msdn.microsoft.com/en-us/library/ms752059.aspx. Acesso em 20/Junho/2008.

[6] Managed Library for Nintendo's Wiimote. Disponível em http://www.codeplex.com/WiimoteLib/Release/ProjectReleases.aspx?ReleaseId=7880. Acesso em 6/Maio/2008

[7] Porta USB. Disponível em http://en.wikipedia.org/wiki/Universal_Serial_Bus. Acesso 15/Junho/2008.

[8] J. Barton, T. The Cooltown User Experience. Disponível em http://www.hpl.hp.com/techreports/2001/HPL-2001-22.pdf. Acesso em 27/Junho/2008.

[9] A Unified Remote Console Based on Augmented Reality in a Home Network Environment. Disponível em http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4146123. Acesso em 15/Junho/2008.

[10] Microsoft Surface. Disponível em http://en.wikipedia.org/wiki/Microsoft_Surface. Acesso em 27/Junho/2008.

[11] Folha de características do NE555. Disponível em http://www.fairchildsemi.com/ds/LM/LM555.pdf. Acesso em 28/Junho/2008.