

UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
INGENIERO EN INFORMÁTICA

PROYECTO FIN DE CARRERA

**ESTUDIO Y DESARROLLO DE
INTERFACES HOMBRE-MÁQUINA
BASADOS EN SENSORES
INALÁMBRICOS**

AUTOR: ENRIQUE ALCOR MARTÍN

MADRID, Septiembre 2008

Autorizada la entrega del proyecto del alumno/a:

Enrique Alcor Martín

EL DIRECTOR DEL PROYECTO

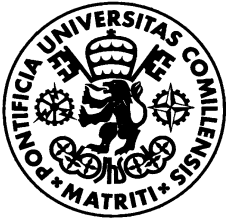
David Contreras Bárcena

Fdo.: Fecha: / /

Vº Bº del Coordinador de Proyectos

David Contreras Bárcena

Fdo.: Fecha: / /



UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
INGENIERO EN INFORMÁTICA

PROYECTO FIN DE CARRERA

**ESTUDIO Y DESARROLLO DE
INTERFACES HOMBRE-MÁQUINA
BASADOS EN SENSORES
INALÁMBRICOS**

AUTOR: ENRIQUE ALCOR MARTÍN

DIRECTOR: DAVID CONTRERAS BÁRCENA

MADRID, Septiembre 2008

AGRADECIMIENTOS

Gracias a mis padres y a toda mi familia, ya que si no fuera por ellos estas líneas no existirían, y por darme todo su apoyo durante la carrera y el proyecto.

Agradecer también a mi director y coordinador del proyecto, David, toda su ayuda durante el ciclo del proyecto y por empujarme a sacar lo mejor de mí mismo, así como a Santiago y Amanda su apoyo en el campo de las matemáticas y del inglés respectivamente.

Y por último a Moni, por soportarme, tanto durante la elaboración de este proyecto como el resto del tiempo.

RESUMEN

Desde su lanzamiento en Noviembre de 2006, la videoconsola *Wii* de *Nintendo* ha distribuido a día de hoy más de 24 millones de unidades en todo el mundo, incluso con un exceso de demanda que ha hecho en muchas ocasiones que fuera difícil encontrar unidades de la misma.

Dicha videoconsola, lejos de sostenerse únicamente en la pulsación de botones para el control de sus aplicaciones, está basada también en los movimientos del usuario, lo que permite una nueva forma de interactuar con el sistema, peculiaridad gracias a la cual ha conseguido tal éxito mundial. El mando/controlador responsable de detectar los movimientos del usuario tiene por nombre *Wiimote* y gracias a su conectividad *Bluetooth*, es posible utilizarlo también como dispositivo de entrada para aplicaciones de PC.

Por otra parte, resulta evidente que la tecnología cada vez tiene más presencia en el ámbito del hogar. Si antes era raro que alguien tuviese un ordenador en casa, ahora es habitual encontrarse varios. La incomodidad de utilizar dispositivos de entrada típicos como el teclado y el ratón en ciertas zonas del hogar como puede ser el salón, requiere ser solventada utilizando nuevas maneras de manejar aplicaciones que den el control de la imagen, el sonido e incluso del hogar al completo mediante dispositivos de domótica.

Este Proyecto Fin de Carrera (PFC) tiene por objeto el estudio de los actuales desarrollos para *Wiimote* en PC y el desarrollo de nuevas aplicaciones que aprovechen el movimiento como forma de interacción. Dichas aplicaciones hacen uso de los siguientes elementos:

- *Wiimote*
- Comunicación *Bluetooth*
- Hardware de domótica con protocolo *X10*
- PC conectado a monitor/TV/proyector

Aunque todas las aplicaciones hacen uso de los movimientos del usuario conviene distinguir entre dos tipos de aplicación. Por una parte están las aplicaciones que responden ante los movimientos del propio *Wiimote*, es decir, el usuario realiza los movimientos con el dispositivo en mano. El segundo tipo recibe las coordenadas de la

posición del usuario (normalmente de uno de sus dedos) utilizando el *Wiimote* como dispositivo de seguimiento, ya que en su interior dispone de una cámara de infrarrojos, y opera en base a ellas. Esto se consigue dotando al usuario de algún tipo de emisión de luz infrarroja hacia el *Wiimote*.

También se pueden separar las aplicaciones por otro criterio más general:

- Aplicaciones de segundo plano: Estas aplicaciones están situadas en una capa intermedia entre el sistema operativo y la aplicación que se desea utilizar, que puede ser por ejemplo un navegador web, un gestor de medios o incluso un juego. Lo que hacen es, en base a los movimientos y gestos del usuario, mover el cursor del ratón por la pantalla y simular la pulsación de teclas, tanto de ratón como de teclado, por lo que, previamente configuradas, sirven para manejar cualquier aplicación prescindiendo parcial o totalmente de otros dispositivos. Entre estas aplicaciones se pueden encontrar tanto algunas que responden según los movimientos del *Wiimote*, y otras que operan bajo los movimientos del usuario.
- Aplicación con interfaz de usuario: Se ha desarrollado también una aplicación con interfaz de usuario propia para el manejo de los dispositivos domóticos del hogar en base a los propios movimientos y gestos del usuario. Dicha interfaz está 100% enfocada a este tipo de interacción.

Gracias a este amplio abanico de aplicaciones, se pretende demostrar que no es necesario utilizar aplicaciones específicas para manejarlas mediante movimiento, si bien pueden adaptarse de manera que sea más sencilla su utilización. Para ayudar posibles futuros desarrollos, se ha creado un anexo (II) en forma de manual para aprender a integrar el uso del *Wiimote* en aplicaciones programadas en lenguaje *C#*.

Debido a que la forma de interactuar con la mayoría de las aplicaciones realizadas en este proyecto, no requieren extensos ni bruscos movimientos, menos incluso que el manejar un teclado y un ratón, pueden ser perfectamente aprovechadas por personas de movilidad reducida (o *PMR*) para facilitar, tanto su vida personal y familiar, como una mejor y más fácil incorporación al mundo laboral.

ABSTRACT

Since its launch in November 2006, *Nintendo's* latest videogame console, *Wii*, has sold more than 24 million units worldwide and the demand has been higher than the supply.

Wii is not only based on button pressing to control the applications and games, it is also based on user movements. This has created a new form of interacting with a system and this peculiarity is the main point of its international success. The remote control that detects the user's movements is called *Wiimote*, and because it communicates using *Bluetooth* technology, it is also possible to use it as a PC input device.

On the other hand, it is obvious that each passing day technology has a bigger presence in the home environment. Years ago it was strange to find a computer at home, but today it is usual to have more than one. Using typical computer input devices, such as the keyboard or mouse, in the living room may not be comfortable. So, a need to develop new interfaces that can give the image control, sound and whole house automation exists.

This project's main objective was to review all actual PC *Wiimote* applications and to create new applications that integrate movement as an interface. The new applications make use of the following elements:

- *Wiimote*
- *Bluetooth* communication
- Home automation hardware using *X10* protocol
- Monitor/TV/projector connected to a PC

All applications make use of user movements, but there is a need to distinguish between two types. First, we have applications that respond to *Wiimote* movements, whereby the user makes movements with this device in their hand. Second, the application receives the user's position (typically a finger) using the *Wiimote* as a tracking device by means of an IR camera that monitors the user's position. In the case of the IR camera, the user must emit IR light so the *Wiimote* can track it; however, we want to classify our new applications using more general criteria:

- **Background applications:** These applications are located in a mid layer between the OS and the application, and correspond to a web browser, a media center, or even a game. Based upon the user's movements and gestures, the cursor is moved on the screen and simulates the use of the keyboard. Previously configured, they are capable of partially or totally working with any application, without using any other device.
- **User interface application:** I have also developed an application that has its own user interface managing and controlling home automation devices through user movements and gestures. The user interface is designed to contemplate gestures and movements, but the user can also use the keyboard and/or mouse.

With all these various applications, we would like to demonstrate that it is not necessary to use specific applications in order to control them using movement, but that we can also adapt them make them easier to use. A manual about how to develop C# applications using the *Wii mote* was created in order to help future developments.

The interfaces developed in this project do not need rapid or precise movements, in fact they need fewer movements than those required by a standard mouse or a keyboard. Thus, they may be useful for disabled people in both their personal and professional life.

Índice

Capítulo 1. Descripción del proyecto	3
1.1. Motivación	4
1.2. Objetivos	5
1.3. Metodología	7
1.4. Planificación	8
1.5. Recursos utilizados	12
Capítulo 2. Tecnología utilizada.....	13
2.1. Hardware.....	14
2.1.1. Controlador Wiimote.....	14
2.1.2. Protocolo X10.....	20
2.1.3. Matriz de leds infrarrojos	26
2.1.4. Puntero de luz infrarroja.....	28
2.2. Software	30
2.2.1. Librerías para C# y Visual Basic.....	30
2.2.2. Otras librerías y drivers	33
2.2.3. GlovePIE	37
2.2.4. Desarrollos de Johnny Chung Lee.....	38
Capítulo 3. Desarrollo de Software para Wiimote	44
3.1. Creación de Scripts para GlovePIE.....	45
3.1.1. Mando a distancia convencional	46
3.1.2. Dispositivo apuntador.....	48
3.1.3. Desplazamiento por aceleración.....	51
3.2. Desarrollos basados en movimientos del usuario	54
3.2.1. Aplicación para el control del cursor (<i>WiiClick</i>).....	57
3.2.2. Aplicación para dibujo por movimiento (<i>WiiAirBoard</i>).....	68

3.2.3. Aplicación de reconocimiento de gestos (<i>WiiGestures</i>).....	71
3.2.4. Aplicación de domótica (<i>WiiHome</i>)	82
Capítulo 4. Presupuesto	91
Capítulo 5. Conclusiones y trabajos futuros	94
Capítulo 6. Bibliografía.....	97
Anexo I. Manual para conexión de Wiimote al PC	100
Anexo II. Manual de programación de Wiimote en C#	108

Capítulo 1

Descripción del

proyecto

1.1. Motivación

A pesar de la gran cantidad de consolas *Wii* distribuidas (más de 24 millones) por los hogares de todo el mundo y por tanto de su controlador *Wiimote*, aún existen pocos desarrollos para PC que aprovechen las posibilidades que ofrece. Muchos de los desarrollos existentes fueron creados como aplicaciones de demostración que no tienen una utilidad real. Es objetivo de este proyecto el crear nuevas aplicaciones innovadoras y actualmente inexistentes aprovechando las posibilidades del dispositivo. Debido a las limitaciones a la hora de manejar aplicaciones en el entorno del salón del hogar, resulta interesante aprovechar este nuevo método de interacción para eliminarlas.

La gran ventaja del controlador *Wiimote* frente a otros dispositivos similares, es que consigue aunar en una misma carcasa distintas tecnologías a un precio muy ajustado, gracias a su producción a gran escala. Tecnologías cuyo coste de adquisición, incluso en el caso de obtenerlas por separado, sería superior. Además, dado su carácter autónomo, no se requiere la adquisición de una consola *Wii* para su utilización conjunta con aplicaciones de PC, por lo que los costes en los que se incurren para la ejecución de las mismas son muy bajos o incluso nulos en caso de ya disponer de la consola junto a su correspondiente *Wiimote*.

Tal y como se detallará más adelante, existe la posibilidad de controlar aplicaciones no solo con el movimiento del propio *Wiimote* si no también con el de nuestros dedos, por lo que sería posible adaptar algunas de las aplicaciones a desarrollar para su manejo por personas de movilidad reducida (o PMR) que con simples movimientos de un dedo puedan controlar diversas aplicaciones.

Por último, cabe señalar que dado el carácter reciente de las aplicaciones desarrolladas hasta el momento, no existe un punto central de referencia para conocerlas todas. Es un objetivo el estudio y documentación de los mismos.

1.2. Objetivos

Dado que los desarrollos para *Wii mote* son escasos en número y muy recientes, es difícil aproximar con exactitud la duración de la parte de desarrollo del proyecto por lo que se han englobado los objetivos en primarios y secundarios según su prioridad.

En cuanto a objetivos primarios se han fijado los siguientes:

- Estudio de los desarrollos actuales
- Estudio de las APIs de desarrollo para *Wii mote*
- Identificación de aplicaciones a desarrollar
- Construcción de aplicaciones básicas

Por otra parte, y en función de la complejidad de los anteriores objetivos, se desarrollarán aplicaciones de mayor complejidad.

Tras el estudio del estado del arte y las librerías, se han identificado las siguientes aplicaciones a desarrollar:

- Aplicación para navegación básica en un *media-center* (se utilizará *Meedio*) basada en pulsación de botones. Se trata de un *script* para *GlovePIE*.
- Aplicaciones para la navegación en *Meedio* utilizando las distintas posibilidades de movimiento del *Wii mote* que se detallarán más adelante. *Scripts* para *GlovePIE*.
- Aplicación para utilizar el *Wii mote* a modo de ratón, llamada *WiiClick*, permitiendo tanto hacer *click* como arrastrar y soltar (*drag & drop*)
- Aplicación para dibujar realizando movimientos de mano en el aire, llamada *WiiAirBoard*.
- Aplicación de reconocimiento de gestos (*WiiGestures*). Utilizará la anterior como base para añadir reconocimiento de gestos como trazados horizontales, verticales y diagonales.

- Programa para el manejo de dispositivos de domótica (*WiiHome*) como el encendido de lámparas, subida y bajada de persianas,...etc.

1.3. Metodología

Dado el carácter novedoso y abierto del proyecto es difícil utilizar una metodología clásica. Por una parte, al tratarse de un proyecto de innovación tecnológica, existe escasa documentación sobre las tecnologías utilizadas. Es el caso del *Wiimote* para el cual apenas hay referencias. Además, en cuanto a su apertura a nuevas ideas durante el desarrollo de cada aplicación, no es fácil determinar la complejidad de cada una de ellas.

Para abordar los problemas comentados anteriormente se utilizará una metodología basada en prototipos simples que evolucionarán en aplicaciones más complejas. Para ello se dejará a los prototipos la implementación del código para el manejo de las distintas tecnologías de *Wiimote* y *XIO*, y posteriormente añadiéndoles una interfaz de usuario más potente y configurable.

Para cada prototipo y aplicación se definen los siguientes apartados:

- **Requisitos:** En este apartado se definen los objetivos que deberá cumplir el prototipo o aplicación. Podrán añadirse más posteriormente.
- **Diseño:** Mediante la utilización de esquemas y flujos de ejecución se expondrá la manera de implementar cada prototipo o aplicación. Se especificarán también las tecnologías hardware y software utilizadas para llevarla a cabo.
- **Implementación:** Se expondrá la manera en que ha sido implementada cada solución.
- **Utilización:** Se dará una breve explicación sobre la forma de utilizar el prototipo o aplicación, como la preparación previa de dispositivos o el manejo básico. Se adjuntarán imágenes de la aplicación, en caso de disponer de interfaz propio.

1.4. Planificación

El proyecto ha sido dividido en varias fases que comprenden una duración total de cuatro meses y medio (de 14 de abril a 31 de agosto) y a lo largo de las cuales se han desarrollado otras sub-fases más específicas:

- **Fase 1 - Definición y arranque del proyecto:** En esta fase se establece contacto con el director del proyecto y se definen los principales objetivos para poder arrancar el mismo. Esta fase genera además la siguiente documentación:

Documentación generada:

- Descripción del proyecto

Duración: 14 a 18 de abril.

- **Fase 2 – Estudio de tecnologías y algoritmos matemáticos, preparación de elementos:** Aquí se estudian las tecnologías y algoritmos matemáticos necesarios para el desarrollo del proyecto, además de preparar los distintos elementos necesarios para el desarrollo y ejecución de las aplicaciones como la matriz de leds y el puntero de infrarrojos que requieren la compra de piezas y el ensamblaje de las mismas. Debido al carácter modular del proyecto, con distintas aplicaciones, las tecnologías más específicas de cada una de ellas se estudian justo antes de comenzar su desarrollo. En concreto se estudiarán y prepararán:

- **Lenguaje C#**
- **Controlador *Wiimote* y sus librerías.**
- **Protocolo *X10* y librerías.**
- **Matriz y puntero led.**
- **Software *GlovePIE*.**
- **Desarrollos de *Johnny Chung Lee*.**

Documentación generada:

- Tecnología utilizada.
- Explicaciones y figuras en los apartados de requisitos y diseño de las aplicaciones.

Duración: 19 de abril a 30 de julio.

- **Fase 3 – Desarrollo de *scripts* para *GlovePIE*:** En esta fase se codifican y documentan los diversos *scripts* creados para *GlovePIE*.

Documentación generada:

- Requisitos, diseño, implementación y utilización de cada *script*.

Duración: 12 a 25 de mayo.

- **Fase 4 – Desarrollo de *WiiClick*:** En esta fase se codifica y documenta la aplicación *WiiClick*.

Documentación generada:

- Requisitos, diseño, implementación y utilización de *WiiClick*.

Duración: 26 de mayo a 30 de junio. **Nota:** Parcialmente coincidente con exámenes finales de junio.

- **Fase 5 – Desarrollo de *WiiAirBoard*:** En esta fase se codifica y documenta la aplicación *WiiAirBoard*.

Documentación generada:

- Requisitos, diseño, implementación y utilización de *WiiAirBoard*.

Duración: 1 a 10 de julio. **Nota:** Parcialmente coincidente con exámenes finales de junio.

- **Fase 6 – Desarrollo de *WiiGestures*:** En esta fase se codifica y documenta la aplicación *WiiGestures*.

Documentación generada:

- Requisitos, diseño, implementación y utilización de *WiiGestures*.

Duración: 11 a 27 de julio.

- **Fase 7 – Desarrollo de *WiiHome*:** En esta fase se codifica y documenta la aplicación *WiiHome*.

Documentación generada:

- Requisitos, diseño, implementación y utilización de *WiiHome*.

Duración: 28 de julio a 20 de agosto.

- **Fase 8 – Documentación:** Esta fase engloba toda la documentación generada en otras fases, más la documentación que no tiene cabida en ninguna de ellas.

Documentación generada:

- Presupuesto.
- Conclusiones y trabajos futuros.
- Bibliografía.

Duración: 14 de abril a 31 de agosto.

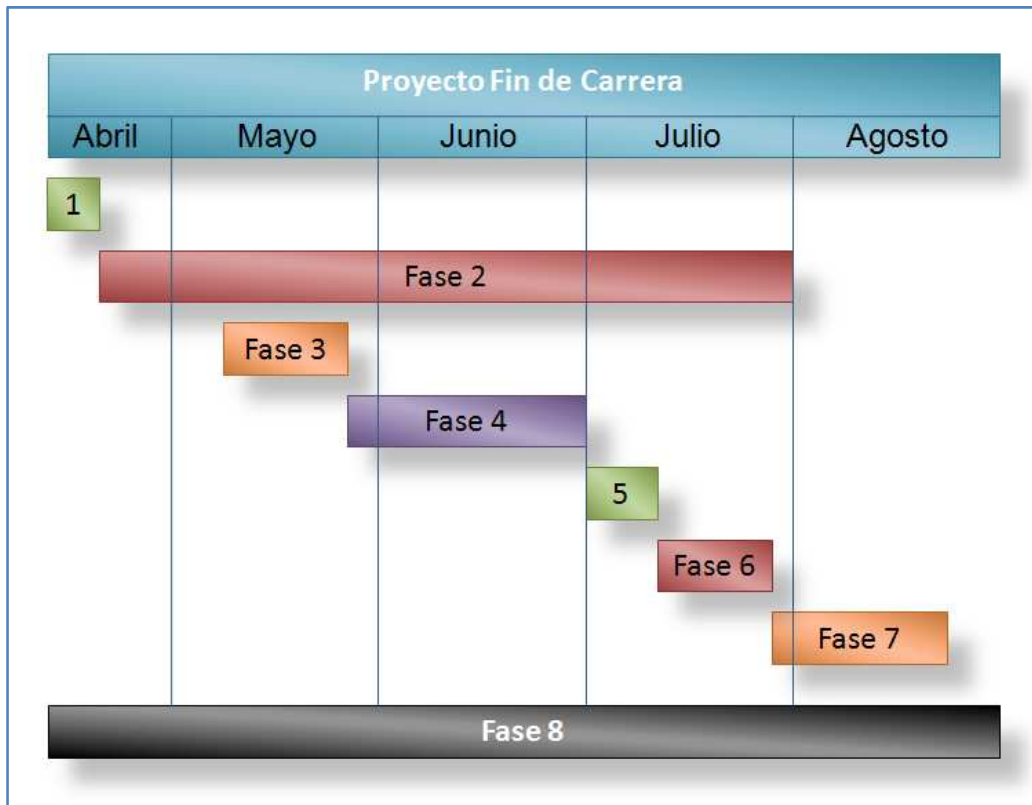


Figura 2.1.1. Esquema de planificación del proyecto

1.5. Recursos utilizados

Para la realización del proyecto se requiere la utilización de una serie de recursos tanto software como hardware.

▪ **Software:**

- **Visual Studio C#:** El API de desarrollo para la interacción con el *Wiimote* está desarrollada para su uso con C# y Visual Basic. Se utilizará el primer lenguaje por su mayor potencia.
- **.NET Framework:** Requerido para trabajar con las librerías de *Wiimote*.
- **Windows XP:** Se utilizará este S.O. como base para el desarrollo de las aplicaciones dado que es el más compatible con las librerías de *Wiimote*. No obstante se probarán también en Windows Vista para tratar de maximizar la compatibilidad.
- **Medio:** Software *media-center* con posibilidad de reproducir audio, vídeo e imagen además de disponer mediante plugins de funciones para el control de dispositivos de domótica.

▪ **Hardware:**

- **PC Portátil:** Un PC de gama media-alta que será suficiente para el desarrollo y pruebas de la aplicación.
- **Wiimote:** Será el dispositivo de capturar los movimientos del usuario mediante su cámara de infrarrojos y acelerómetros.
- **Dongle Bluetooth:** Para la comunicación PC-Wiimote.
- **Módulos X10:** Un controlador y diversos actuadores para el envío y recepción de señales en protocolo X10 para el encendido y apagado de luces a través de la línea eléctrica.

Capítulo 2

Tecnología

utilizada

2.1. Hardware

2.1.1. Controlador Wiimote

Este proyecto gira en torno al mando/controlador de la videoconsola *Wii*^[1] de *Nintendo* que va más allá de la simple pulsación de botones para el control de los videojuegos, ya que dispone de distintos sensores para la detección de los movimientos del usuario.



Figura 2.2.1.1. Wiimote



Figura 2.1.1.3. Nunchuck

El controlador *Wii Remote*^[2], referido de aquí en adelante como *Wiimote* (Figura 2.1.1.1.) está formado de los siguientes elementos:

- Botones (POWER, A, -, HOME, +, 1, 2 y B como gatillo en la parte inferior)
- Cruz direccional
- 4 leds: indican el número de jugador o el nivel de batería.
- 2 pilas AA: con una duración de entre 30 y 60 horas según el uso o no de la función de puntero.
- Acelerómetro *ADXL330* de 3 ejes: para detección de movimientos del usuario.
- Sensor óptico de infrarrojos: para detección de movimientos del usuario.
- Puerto de expansión: para la conexión de otros elementos como el controlador *Nunchuck* (Figura 2.1.1.3.)

- Conexión Bluetooth: para la comunicación entre consola y el propio *Wiimote*.
- Altavoz
- Motor de vibración
- Chip de memoria EEPROM de 16 KiB: destinado a guardar información del juego en ejecución y hasta 10 perfiles de usuario (denominados *Mii*)

2.1.1.1. Funcionamiento

Como se comentó anteriormente, el controlador *Wiimote* se basa, aparte de en la clásica pulsación de botones, en los movimientos del usuario para el control de las aplicaciones y juegos de la videoconsola *Wii*. Dichos movimientos se detectan utilizando dos elementos del controlador:

- Acelerómetro de 3 ejes
- Sensor de infrarrojos

El primero de ellos mide la inclinación del mando en 3 ejes y se transmite a la consola a través del enlace Bluetooth. Esta forma de detección del movimiento se usa por ejemplo en juegos de conducción, sosteniendo el mando en posición horizontal e inclinándolo en sentido anti horario para girar a izquierdas u horario para girar a derechas.

El segundo elemento, el sensor de infrarrojos, es similar al de una cámara de visión infrarroja y se utiliza para detectar hacia donde apunta el usuario con el mando de manera que pueda que use el mando como puntero para desplazarse por pantallas e interactuar con juegos de puntería, indicar al personaje la dirección de movimiento,... etc. Para su funcionamiento, se requiere una fuente de luz infrarroja que pueda “ver” el sensor para transmitir a la consola su posición. Para solucionar esto, la videoconsola *Wii* se suministra junto a una barra con leds infrarrojos llamada “*Wii Sensor Bar*” (*Figura 2.1.1.1.4*) que se coloca encima o debajo de la pantalla de TV conectada a la consola. En su interior existen 10 leds infrarrojos, 5 en cada extremo.



Figura 2.1.1.1.4. Sensor Bar

2.1.1.2. Conectividad

Dado que el *Wii* utiliza *Bluetooth* (de aquí en adelante BT) para la transmisión de datos con la videoconsola *Wii*, se puede aprovechar dicha tecnología para la comunicación con él mediante un PC, teléfono móvil o cualquier dispositivo que disponga de la misma.

Gracias a la posibilidad de conectar mediante BT el *Wii* y un PC, han surgido diversas librerías y aplicaciones para su utilización como dispositivo de entrada sustituyendo por ejemplo un ratón, teclado, o joystick. Es objeto de este proyecto estudiar las librerías y aplicaciones más representativas y el desarrollar nuevas aplicaciones basadas en ellas.

2.1.1.3. Accesorios

Además del propio *Wii*, como ya se ha comentado, se pueden utilizar accesorios conjuntamente con él, u otros de forma independiente, pero que cabe mencionar en estas líneas puesto que también son bastante novedosos y compatibles con PC.

2.1.1.3.1. *Nunchuck*

Este periférico mencionado y mostrado anteriormente se conecta a la parte inferior del *Wii mote*, y provee de cuatro elementos para el control de los juegos y aplicaciones de *Wii*:

- **Botón C**
- **Botón Z**
- **Stick analógico**
- **Acelerómetro:** Al igual que el propio *Wii mote*, este accesorio incluye un acelerómetro, por lo que se podrán realizar dos movimientos distintos al mismo tiempo para aumentar las posibilidades de interacción.

2.1.1.3.2. *Classic Controller*

El *Classic Controller* es la apuesta de Nintendo para una forma de control más clásica de los juegos. Se trata de un mando tradicional sin detección de movimiento con dos sticks analógicos y botones.



Figura 2.5.1.3.2.1. Wii Classic Controller

2.1.1.3.3. *Guitarra*

A finales de 2005, *Harmonix Music Systems* lanzó al mercado un juego musical llamado *Guitar Hero*, que supuso un gran éxito internacional. El juego, que trataba de emular a un gran guitarrista, estaba acompañado de una guitarra especial para su utilización con la consola *Playstation 2* que hacía mucho más satisfactorio el control del juego que con un simple mando tradicional.

La tercera entrega del juego fue lanzada entre otras consolas para *Wii*, lo que obligó a los desarrolladores a crear una guitarra específica para esta consola. La guitarra para *Wii* está pensada para albergar en su interior al *Wiimote* y comunicarse con él a través del puerto de extensiones, dotando a la guitarra de sensor de movimiento (implementado en el modelo para otras consolas de manera autónoma) y vibración (exclusiva para la versión *Wii* del juego). Este accesorio es actualmente compatible con PC a través de las librerías y programas que se comentarán en posteriores apartados.



Figura 2.6.1.3.3.1. Guitarra para Wii

2.1.1.3.4. *Wii Balance Board*

Este periférico es la nueva apuesta de *Nintendo* como controlador de juegos. Es independiente del *Wiimote*, por lo que está dotado de conexión *Bluetooth*. Se trata de una tabla con cuatro sensores de presión en su interior pensada para el control de software a través de movimientos de nuestro propio cuerpo por cambios en la presión de la misma.

La tabla fue lanzada recientemente junto al software *Wii Fit*, pensado no para su utilización como juego si no de entrenador físico. La aplicación propone diversos ejercicios utilizando la *Wii Balance Board*, situándonos de pie sobre ella o bien la realización de otros ejercicios como flexiones apoyando las manos. Ha sido un gran éxito internacional y ya es compatible con PC.



Figura 2.7.1.3.4.1. Wii Balance Board

2.1.2. Protocolo X10

Una de las aplicaciones realizadas en el proyecto hace uso del protocolo X10^{[3][4][5]}, utilizado para la comunicación entre dispositivos destinados a la automatización del hogar, habitualmente a través de la línea eléctrica aunque también mediante ondas de radio. Desarrollado en 1975 y siendo la primera tecnología domótica de la historia sigue siendo la más utilizada y extendida a día de hoy.

La tecnología domótica ofrece la posibilidad de controlar una amplia gama de elementos del hogar como pueden ser luces, persianas automáticas, sistemas de alarma, porteros automáticos, climatización... o recibiendo datos de diversos sensores como por ejemplo de humedad, temperatura o fuego para tomar las medidas pertinentes.

Es muy frecuente la utilización de macros para la ejecución de diversas órdenes de alguna manera relacionadas como puede ser la creación de diversos ambientes, por ejemplo el salón, como encender la televisión y apagar las luces en un supuesto “modo cine”, atenuarlas y encender el equipo de música cuando queramos relajarnos o simular presencia a determinadas horas para evitar robos. También es usual la toma de decisiones ante la recepción de determinados valores por parte de sensores, por ejemplo llamar a los bomberos en caso de detección de fuego o encender el aire acondicionado si se supera un determinado umbral de temperatura.

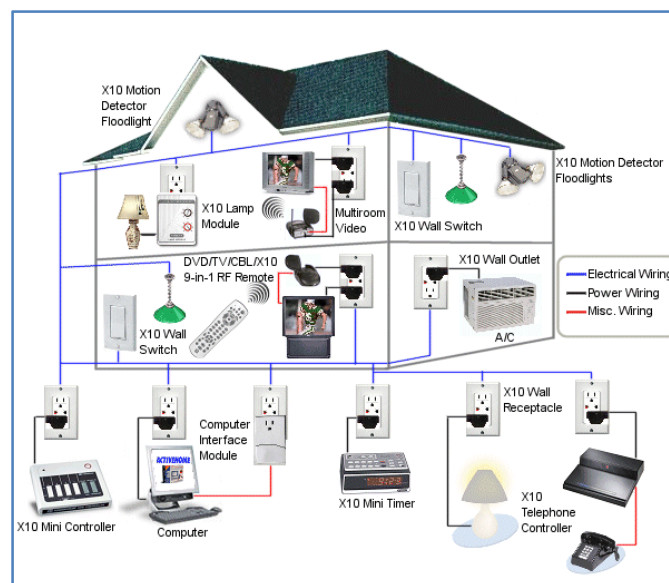


Figura 2.8.2.1. Esquema hogar X10

2.1.2.1. Dispositivos X10

Existen diversos dispositivos que funcionan con protocolo X10 y pueden ser de tres tipos:

- **Controladores:** Envío de órdenes y recepción de datos.
- **Actuadores:** Ejecución de órdenes.
- **Sensores:** Muestreo de datos y envío de los mismos.

2.1.2.1.1. Controladores

El controlador es el dispositivo esencial en cualquier sistema X10, ya que es el encargado de dar órdenes al resto de dispositivos, como por ejemplo a un módulo actuador para lámparas para que se encienda o apague.

Existen diversos tipos de controladores:

- **Miniprogramador:** Dispositivo a través del cual se pueden enviar directamente señales a dispositivos X10 o crear programas y macros.
- **Mandos a distancia RF:** Mandos a distancia universales para el control de electrodomésticos como la televisión además de dispositivos X10 a través de radiofrecuencia.
- **Mandos RF:** Similares a los anteriores pero sólo para el control de dispositivos X10.
- **Programador para PC:** El más completo ya que utilizando software comercial específico o implementando librerías al crear los nuestros propios, se pueden enviar y recoger señales X10 para un control total y sin límites de nuestro hogar. Cuentan con un conector serie o USB para su conexión al PC.

Tanto en el caso del miniprogramador como el de PC suelen estar conectados a la red eléctrica y opcionalmente pueden enviar y recibir señales por radiofrecuencia a dispositivos X10 inalámbricos.



Figura 2.9.2.1.1.1. Controlador USB para PC

2.1.2.1.2. Actuadores

Los actuadores son los encargados de escuchar las señales que envía el controlador y, en caso de estar destinadas a ellos, ejecutar la acción correspondiente. Generalmente se conectan a la corriente eléctrica y a ellos se conecta el dispositivo que controlan.

A continuación se listan los tipos de actuador más típicos que pueden encontrarse en el mercado:

- **Módulos de potencia:** Se conectan entre un enchufe o directamente al cableado del hogar y el dispositivo a controlar. Cuando reciben la señal de encendido dejan pasar corriente al dispositivo y cortan el suministro si reciben la de apagado.
- **Módulos de iluminación o de lámpara:** De igual funcionamiento que los de potencia pero además controlan el flujo de corriente que dejan pasar a la luz o lámpara que controlan de manera que pueden variar la intensidad de la misma.
- **Módulos de persiana:** En este caso controlan dispositivos con movimiento en dos direcciones como pueden ser persianas, toldos, cortinas...



Figura 2.10.2.1.2.1. Módulo de lámpara

2.1.2.1.3. Sensores

Mientras que los actuadores recogen información de la línea de comunicación y realizan acciones, los sensores se limitan a enviar diversas mediciones por la misma para que sean recibidos y procesados por el controlador. Normalmente utilizan comunicación inalámbrica para enviar datos al controlador.

Tipos de sensores X10:

- **Sensores no X10:** Utilizando un transmisor universal X10 se pueden recibir señales de diversos tipos de sensor como detectores de humo y fuego, rotura de cristal, fuga de agua y gas, termómetros, medidores de humedad...
- **Sensores de presencia:** Funcionan por sensibilidad de luz.
- **Termostatos X10**



Figura 2.11.2.1.3.1. Sensor de presencia

2.1.2.2. Funcionamiento del protocolo X10

El protocolo X10 se basa en el envío de paquetes, ya sea por el cableado eléctrico del hogar o por radiofrecuencia. Para poder referirnos a un determinado dispositivo, el protocolo define un sistema de direccionamiento basado en un código de hogar (16 códigos que van de la A a la P) más un código de dispositivo (otros 16 numerados del 1 al 16). Normalmente el código de hogar es único para una misma vivienda, aunque en caso de necesitar más de 16 direcciones por vivienda, se pueden usar varios códigos de hogar. El protocolo permite asignar la misma dirección a más de un dispositivo, por ejemplo para encender varias luces a la vez con un sólo comando.

Un problema típico del protocolo X10 es la posibilidad de recibir señales de otro vecino si el cableado de ambos no está suficientemente aislado. Para solucionarlo se puede acordar el uso de distintos códigos de hogar o bien instalar un capacitador que no deje propagar las señales a otras viviendas.

Un paquete o trama de datos dispone de tres campos:

- 1. Código de inicio de trama (4 bits):** Se trata de 4 bits fijos (1110) que sirven para identificar el comienzo de cada trama de datos.
- 2. Código de casa (4 bits)**
- 3. Código de función (5 bits):** El código de función puede ser un código de dispositivo (indicado con un 0 en el último bit) o bien un comando (p. ej.: ON, OFF... indicado con un 1 en el último bit).

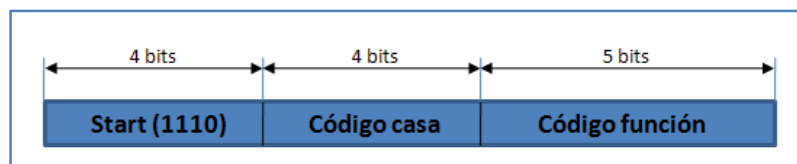


Figura 2.12.2.2.1. Paquete X10

Por tanto, cuando se desee dar una orden a un determinado dispositivo, se envía primero una trama que lo identifique y luego el comando correspondiente.

El protocolo requiere enviar cada trama dos veces seguidas, sea del tipo que sea, para evitar errores. Además requiere también enviar una secuencia de seis ceros seguidos “000000” cuando los datos cambien de una dirección a otra, de una dirección a un comando, o de comando a comando.

HOUSE CODES				KEY CODES						
	H1	H2	H4	H8	D1	D2	D4	D8	D16	
A	0	1	1	0	1	0	1	1	0	0
B	1	1	1	0	2	1	1	1	0	0
C	0	0	1	0	3	0	0	1	0	0
D	1	0	1	0	4	1	0	1	0	0
E	0	0	0	1	5	0	0	0	1	0
F	1	0	0	1	6	1	0	0	1	0
G	0	1	0	1	7	0	1	0	1	0
H	1	1	0	1	8	1	1	0	1	0
I	0	1	1	1	9	0	1	1	1	0
J	1	1	1	1	10	1	1	1	1	0
K	0	0	1	1	11	0	0	1	1	0
L	1	0	1	1	12	1	0	1	1	0
M	0	0	0	0	13	0	0	0	0	0
N	1	0	0	0	14	1	0	0	0	0
O	0	1	0	0	15	0	1	0	0	0
P	1	1	0	0	16	1	1	0	0	0
				All Units Off	0	0	0	0	1	1
				All Lights On	0	0	0	1	1	1
				On	0	0	1	0	1	1
				Off	0	0	1	1	1	1
				Dim	0	1	0	0	1	1
				Bright	0	1	0	1	1	1
				All Lights Off	0	1	1	0	1	1
				Extended Code	0	1	1	1	1	1
				Hail Request	1	0	0	0	1	1
				Hail Acknowledge	1	0	0	1	1	1
				Pre-Set Dim	1	0	1	X	1	1
				Extended Data (analog)	1	1	0	0	1	1
				Status-on	1	1	0	1	1	1
				Status-off	1	1	1	0	1	1
				Status Request	1	1	1	1	1	1

Figura 2.13.2.2.2. Códigos de casa, dispositivo y comandos

En cuanto al aspecto físico del protocolo, las señales de control X10 se basan en la transmisión de ráfagas de pulsos RF (120 KHz) sincronizados en el cruce por cero de la señal de red (de 50 o 60 Hz dependiendo del país). Un cero se representa por la ausencia de pulso en el cruce por cero seguido por la presencia de un pulso. El uno se representa a la inversa.

Haciendo cálculos, el protocolo da una tasa de transferencia aproximada de 20 bits/s, hoy en día excesivamente baja, aunque perfectamente válida para señales sencillas como son las de X10. Es por ello que al dar una orden se nota un cierto retardo en observar los resultados.

2.1.3. Matriz de leds infrarrojos

Algunas de las aplicaciones desarrolladas en este proyecto requieren la utilización o bien de una matriz de leds infrarrojos o bien de un puntero de luz infrarroja, tal y como se explica en la descripción y requisitos de cada aplicación.

La matriz de leds se utiliza para proyectar luz infrarroja hacia el usuario de la aplicación y que, utilizando algún material reflectante, sea capaz de reflejar dicha luz para que el *Wiimote*, a través de su cámara de infrarrojos, pueda hacer un seguimiento de los movimientos del usuario.

Existen multitud de matrices de leds infrarrojos en el mercado, normalmente destinadas a su utilización con cámaras de vigilancia con modo de visión nocturna, para lo cual se necesita proyectar luz infrarroja, que siendo invisible para el ser humano, es capaz de iluminar a una persona para poder ser reconocida por la cámara. La ventaja de estas matrices es que se venden montadas y suelen ir encerradas en una carcasa que facilita su transporte, orientación, y las hace más resistentes.



Figura 2.14. Distintas matrices de Leds infrarrojos.

También existe la posibilidad de montar una matriz de infrarrojos personalizada comprando por separado los leds, resistencias y una placa donde soldar dichos componentes, además de una fuente de alimentación o baterías. La ventaja de montar una matriz personalizada reside en que se puede dejar un hueco en medio de todos los leds para poder colocar el *Wiimote*.

Es importante elegir leds de buena calidad tanto al montar nuestra propia matriz como a la hora de elegir una ya montada, puesto que su luminosidad determinará la distancia máxima a la que podremos situarnos del *Wimote* para que pueda ver reflejada la luz. También influye en gran medida la cantidad de Leds por el mismo motivo, y su ángulo de emisión de luz, que debe ser suficiente para que podamos realizar movimientos en un área suficientemente grande.

En cuanto al material necesario para reflejar la luz proporcionada por la matriz, se pueden adquirir rollos de cinta reflectante utilizados para dotar de mayor visibilidad a señales, peatones o vehículos en la oscuridad. Este tipo de cinta puede adquirirse en tiendas de señalización o de productos náuticos.



Figura 2.15. Cinta reflectante adhesiva 3M.

2.1.4. Puntero de luz infrarroja

Tal y como se comentó en el anterior apartado, algunas aplicaciones requerirán la utilización de una matriz de leds infrarrojos junto a un material reflectante, o bien algún dispositivo que emita luz infrarroja para utilizarlo a modo de puntero.

Debido a la dificultad para encontrar en el mercado punteros de luz infrarroja, se ha tenido que optar por la construcción de los mismos. Existen dos posibilidades para construirlos: conseguir algún puntero con led de cualquier color y sustituirlo por uno de tipo infrarrojo, o bien construirlo desde cero.

Para construir un puntero se pueden utilizar los siguientes elementos^[6]:

- **Rotulador:** Utilizado como carcasa. Debe poderse desmontar por ambos extremos. Se recomienda el rotulador *Pilot Super Color Marker*.
- **Led infrarrojo:** En principio cualquiera es válido, aunque se recomiendan los modelos *Vishay TSAL6400* o *TSAL5300*.
- **Pulsador:** Los programas requieren hacer desaparecer y reaparecer la luz infrarroja, por lo que se necesita una manera sencilla de hacerlo. Un pulsador es la mejor solución, ya que se puede encender la luz manteniéndolo pulsado y apagarla soltándolo.
- **2x Pilas AAA:** Para alimentar el led. Se recomiendan alcalinas para una duración más prolongada.
- **Resistencia 12 ohm ¼ vatio**

Una vez se disponga de todos estos elementos se puede proceder a construir el puntero siguiendo el siguiente esquema:

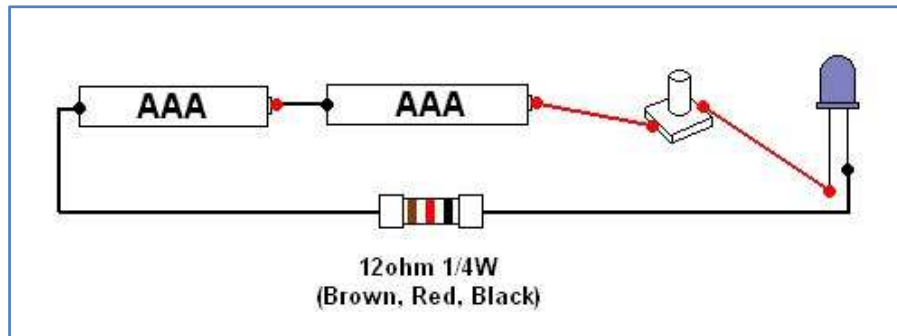


Figura 2.16. Esquema montaje puntero infrarrojo.

2.2. Software

2.2.1. Librerías para C# y Visual Basic

Para la codificación de los programas realizados en este proyecto se ha utilizado C# como lenguaje de programación y .NET como plataforma de desarrollo. No obstante el *Wiimote* no está soportado nativamente ni por las librerías estándar de C# ni por .NET, por lo que se requiere la utilización de unas adicionales.

La librería utilizada, llamada *WiimoteLib* y creada por *Brian Peek*^[7], es válida para su utilización tanto en lenguaje C# como Visual Basic y permite tanto leer los valores capturados por los acelerómetros, la cámara infrarroja de *Wiimote* y los botones como del dispositivo conectado al mismo a través del puerto de expansión. Recientemente también han sido añadidas nuevas funcionalidades como la posibilidad de leer los valores capturados por otros accesorios como la *Wii Balance Board*, suministrada con el juego *Wii Fit*, que utilizada como tabla para la realización de distintos ejercicios físicos situando pies o manos sobre ella, es capaz de detectar la presión ejercida por el usuario. Además, permite enviar algunas órdenes al *Wiimote* como encender las distintas luces LED de las que dispone. También está en desarrollo la posibilidad de enviar sonidos para reproducirlos a través del altavoz del *Wiimote*.

Utilizar el *Wiimote* en un programa realizado en C# o Visual Basic consta de los siguientes pasos:

1. **Asociar el *Wiimote* a Windows**
2. **Realizar la conexión de nuestro programa con el *Wiimote***
3. **Leer informes de *Wiimote* y enviar órdenes**

2.2.1.1. Asociar *Wiimote* con Windows

La manera de hacerlo se explica detalladamente en el Anexo I.

2.2.1.2. Conectar programa con *Wiimote*

Al asociar el *Wiimote* con Windows, este es detectado como un *HID* (Human Interface Device). Por tanto para utilizarlo se requiere acceder a las APIs de gestión de dispositivos y HIDs de Windows. Dado que .NET no dispone de acceso a dichas APIs, se debe utilizar una técnica llamada *P/Invoke* (Platform Invoke) que permite acceder desde .NET a los métodos de la API de Windows. Los métodos *P/Invoke* de estas librerías están definidos en la clase “HIDImports”.

Para conectar con el *Wiimote* se deben seguir los siguientes pasos:

1. Obtener el *GUID* (Globally Unique Identifier) de la clase HID definida por Windows.
2. Obtener una lista de dispositivos que son parte de dicha clase.
3. Obtener información de cada uno de los dispositivos listados.
4. Comparar *VID* (Vendor Identifier, Identificador de fabricante) y *PID* (Product Identifier, Identificador de producto) de cada dispositivo con los del *Wiimote* hasta encontrarlo.
5. Crear un *Stream* (tipo *FileStream*) para leer y escribir en el dispositivo.
6. Limpiar la lista obtenida en el paso 2.

Aunque ya se tiene creado un *Stream* para comunicarnos con el *Wiimote*, aún queda crear la comunicación. Para ello se debe hacer uso de los denominados “informes” que no son más que buffers de datos que utilizan los dispositivos de interfaz humana para mostrar su estado. Se trata de crear un proceso de lectura asíncrona que vaya llenando el buffer. Una vez lleno se llamará a otro proceso que interprete y haga las operaciones pertinentes con los datos del buffer para luego volver al estado de lectura.

Una vez creados dichos procesos, ya se puede interactuar con el dispositivo leyendo su estado y enviándole órdenes.

No obstante la librería utilizada facilita la tarea estableciendo la comunicación con tan sólo crear una instancia de la clase *Wiimote* e invocar su método de conectar.

2.2.1.3. Leer informes del *Wiimote* y enviar órdenes

Tal y como se ha explicado antes, el estado del *Wiimote* se conoce a través de los informes generados. Dichos informes pueden ser de los siguientes tipos:

- **Status:** Estado general del *Wiimote*.
- **ReadData:** Para lectura de datos de la memoria interna.
- **Buttons:** Información sobre botones pulsados.
- **ButtonsAccel:** Información sobre botones y acelerómetros.
- **IRAccel:** Información sobre botones, acelerómetros y puntos infrarrojos.
- **ButtonsExtension:** Información sobre botones y extensiones del mando (periféricos adicionales conectados al *Wiimote*)
- **ExtensionAccel:** Información sobre botones, acelerómetros y extensiones.
- **IRExtensionAccel:** Información sobre botones, acelerómetros, infrarrojos y extensiones.

Al establecer la conexión, se debe especificar qué tipo de informe se desea obtener.

Los informes pueden tratarse de dos maneras distintas, realizando gestión de eventos (habrá uno cada vez que haya disponible un nuevo informe) o bien usando la técnica de *polling*, es decir, comprobando periódicamente el estado del *Wiimote*.

2.2.1.4. Funciones de la librería

Los tipos de informe dejan entrever muchas de las funciones que ofrece esta librería para *Wiimote*, no obstante se listan todas las características de la misma:

- Soporte para múltiples *Wiimotes*.
- Soporte para *Wii Balance Board*.
- Soporte para extensiones del mando: *Wii Classic Controller*, *Nunchuck*, *guitarra de Guitar Hero*.
- Funcionamiento por *polling* o por *eventos*.

- Soporte para el acelerómetro.
- Soporte para la cámara de infrarrojos.
- Lectura de pulsación de botones.
- Lectura del estado de la batería.
- Encendido/Apagado de los Leds.

2.2.2. Otras librerías y drivers

A día de hoy existen diversas librerías y drivers para poder utilizar el *Wiimote* con distintos lenguajes y sistemas operativos:

- **motej (Java)**
- **WiiRemoteJ (Java)**
- **WiiYourself! (C++, Windows)**
- **Wiim (C++, Windows)**
- **wiuse (C, Windows/Linux)**
- **CWiid (C, Linux)**
- **Wiimote-to-joystick daemon/wm2js (Linux)**
- **NI LabVIEW Wiimote Drivers (LabVIEW)**
- **WiiuseJ (Java)**

2.2.2.1. motej

motej^[8] es una librería Java para la comunicación con *Wiimote*, bajo licencia ASL 2.0. Está basada en dos paquetes: librería y extras. La librería ofrece acceso básico al *Wiimote*. Los extras por su parte extienden la funcionalidad básica ofrecida por la librería añadiendo funciones más complejas para facilitar el trabajo al programador.

La arquitectura de esta librería puede comprenderse mediante el siguiente esquema. Como se puede observar, la librería depende de una implementación del estándar de Java *JSR82*, por lo que se requiere una API adicional para el acceso al dispositivo. El módulo de extras accede a las funciones básicas de la librería para extender la funcionalidad de la misma.

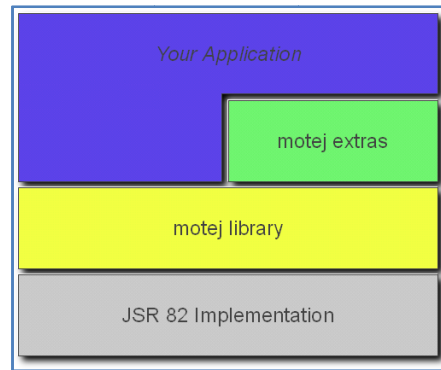


Figura 2.2.2.1.1. Arquitectura motej

motej ofrece las siguientes funcionalidades, tanto para comprobar el estado del *Wiimote* como para ejecutar acciones sobre el mismo:

- Descubrimiento y conexión a dispositivos *Wiimote*.
- Cámara de infrarrojos.
- Acelerómetro.
- Botones.
- Activación/desactivación de la función de vibración.
- Encendido/apagado de Leds.
- Acceso a la memoria EEPROM del dispositivo.
- Acceso a los registros del *Wiimote*.
- Acceso al estado del *Wiimote*.
- Acceso a los datos de calibración.

2.2.2.2. WiiremoteJ

Esta librería^[9] es muy similar a *motej*, también requiere una implementación del estándar *JSR82* y el listado de funcionalidades es idéntico pero añadiendo algunas funciones como soporte para extensiones del mando y del nuevo dispositivo *Wii Balance Board*.

2.2.2.3. WiiYourself!

WiiYourself!^[10] es una librería basada en la creada por *Brian Peek* y utilizada en este proyecto, pero orientada a C++ y extendida en funcionalidad

Además de las funciones que aporta la librería original, añade algunas novedades como:

- **Estimación de la orientación:** Mediante los métodos añadidos es posible detectar el ángulo del *Wiimote* de una manera más sencilla.
- **Mayor soporte de gestores Bluetooth:** Es capaz de utilizar cualquier programa gestor de conexiones Bluetooth.

2.2.2.4. Wiim

Esta librería^[11] también está desarrollada para ser usada en C++, aunque es mucho menos completa que la anterior dado su estado de desarrollo.

2.2.2.5. wiiuse

Aunque no tiene tantas funciones como otras librerías, la ventaja de *wiiuse*^[12], creada para funcionar con lenguaje C, es la posibilidad de utilizarla tanto para programas *Windows* como *Linux*, por lo que resulta más fácil portar programas o juegos de una plataforma a otra.

Sus funciones son las siguientes:

- Soporte para acelerómetros
- Infrarrojos
- Extensiones (*nunchuck*, *classic controller*)
- Soporte para el controlador/guitarra de *Guitar Hero*.

2.2.2.6. CWiid

CWiid^[13], como cuyo nombre indica, está realizada en lenguaje C y orientada a sistemas *Linux*. Actualmente está en una fase de desarrollo muy temprano con una API muy reducida y más complicada de usar que otras. Soporta funciones muy básicas del *Wiimote*.

2.2.2.7. Wiimote-to-joystick daemon/wm2js

wm2js^[14] no es una librería sino un *demonio* (programa ejecutado segundo plano) para utilizar el *Wiimote* en *Linux* como si se tratara de un joystick. La ventaja es que se puede utilizar el *Wiimote* con todos los juegos que soporten joystick aunque a cambio de perder todas las funciones adicionales de movimiento que ofrece el dispositivo.

2.2.2.8. NI LabVIEW Wiimote Drivers

De nuevo no nos encontramos frente a una librería como tal sino con un driver para ser utilizado en programas realizados en *LabVIEW*^[15].

2.2.2.9. WiiuseJ

WiiuseJ^[16] no es más que la adaptación de las librerías *wiiuse* para ser usadas en Java en lugar de C. Por lo tanto las características son idénticas.

2.2.3. GlovePIE

GlovePIE^[17] (Glove Programmable Input Emulator) fue inicialmente creado para la emulación de teclados, ratones y joysticks utilizando guantes de realidad virtual (o VR) modelo P5. Se trata de un intérprete de *scripts* (con extensión *.PIE*) con un lenguaje específico parecido a *Basic*, para la asignación de acciones del guante a pulsaciones del teclado o movimientos de un supuesto joystick o ratón. Actualmente este programa no sólo sirve para la emulación de guantes VR P5 si no que es compatible con una gran cantidad de dispositivos entre los que se incluyen otros modelos de guantes VR, micrófonos, joysticks o gamepads, dispositivos de juego que funcionen por puerto paralelo, máquinas de remo, dispositivos MIDI, y lo que es más interesante, compatibilidad total con *Wii mote*.

Este intérprete tiene dos cualidades que le han hecho cultivar un gran éxito en el mundo del *Wii mote* en PC. Por una parte realizar *scripts* para él es muy sencillo a la vez que completo. Cualquier persona que haya programado en cualquier lenguaje no tardará en adaptarse, e incluso los no programadores encuentran bastante fácil su utilización. La mayor parte de las líneas son simples asignaciones del tipo:

```
Mouse.RightButton = Wiimote.A //Botón A del Wiimote equivale a click derecho  
Mouse.LeftButton = Wiimote.B //Botón B del Wiimote equivale a click izquierdo
```

Aunque también permite la creación de estructuras de control básicas como la clásica IF/ELSE y el uso de variables.

La otra ventaja es que no requiere realizar modificaciones en las aplicaciones o juegos que se deseen utilizar, ya que simplemente *mapea* acciones del *Wii mote* a acciones de teclado, ratón o joystick, por lo tanto es compatible con prácticamente todos los programas y juegos del mercado.

Por último, el éxito que ha cosechado ha hecho que existan cientos de *scripts* en Internet disponibles para su descarga y utilización gratuita, por lo que utilizar el *Wii mote* como dispositivo de juego para PC es muy sencillo.

2.2.4. Desarrollos de Johnny Chung Lee

Entre el software ya creado para su utilización en PC y el *Wiimote* destacan las aplicaciones creadas por *Johnny Chung Lee*^[18], que si bien algunas no tienen una aplicación práctica directa sí que sirven como inspiración para crear novedosas aplicaciones como las desarrolladas en este proyecto. Todas ellas destacan en que utilizan el *Wiimote* de una manera muy distinta a la que originalmente fue destinado.

2.2.4.1. Aplicación de seguimiento de dedos

Esta aplicación ha servido de inspiración para varias de las creadas a lo largo de este proyecto como se detallará en el siguiente capítulo.

La idea principal de la aplicación no es el movimiento del *Wiimote* por el espacio para detectar movimientos del usuario, sino todo lo contrario, dejando el periférico en una posición fija y haciendo uso de su cámara de infrarrojos “ver” las acciones del usuario, en concreto las de sus dedos. Para ello se sirve de una matriz de leds infrarrojos que proyectan luz sobre el usuario y este refleja con sus dedos gracias a una cinta reflectante pegada a los mismos. El efecto es el mismo que utilizar un led infrarrojo apuntando hacia el *Wiimote*, pero de esta manera es mucho más cómodo e intuitivo. La siguiente figura muestra un esquema del funcionamiento de la aplicación.

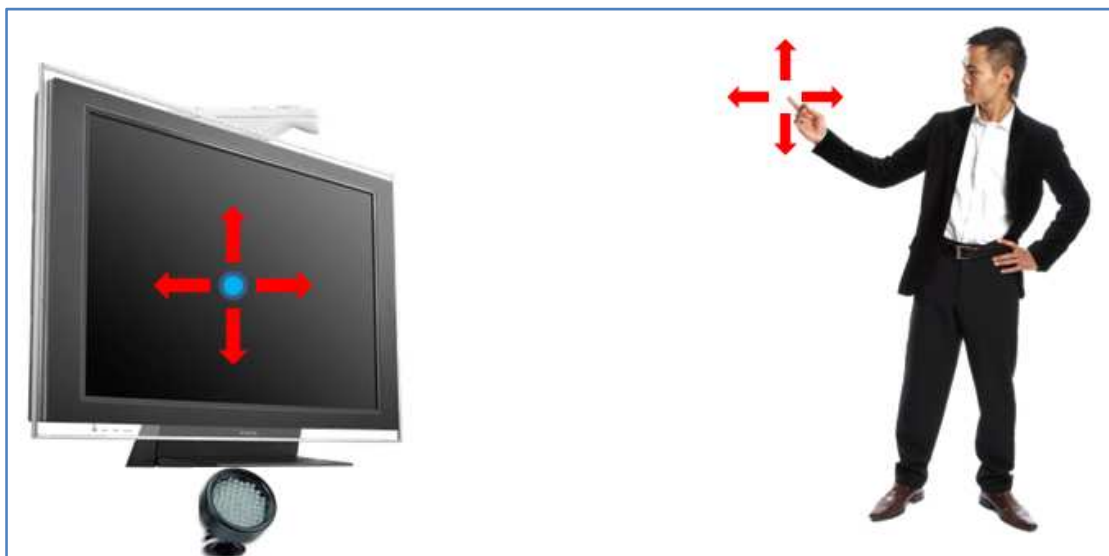


Figura 2.2.4.1.1. Esquema funcionamiento aplicación de seguimiento de dedos

La implementación de la aplicación es bastante sencilla. La mayor parte del código de la misma está destinado a una función extra incluida y es la manipulación de un plano, que utilizando dos puntos de luz (uno en cada dedo índice, es decir, un reflectante en cada uno de ellos) es capaz de mover (desplazamiento de un solo dedo) girar (moviendo los dedos “dibujando” en el aire una circunferencia), agrandar (alejar los dedos entre sí) y encoger (acercándolos).



Figura 2.2.4.1.2. Esquema de las acciones posibles en la aplicación

2.2.4.2. Aplicación de pizarra interactiva

Las pizarras interactivas, que actualmente se están introduciendo en algunos centros de enseñanza y oficinas, son aparentemente pizarras normales que realmente van mucho más allá, ya que sobre ellas se proyecta la imagen procedente de un ordenador, y sobre las que el profesor o ponente puede dibujar o escribir con naturalidad, pero con la ventaja de que no se utiliza tinta ni tiza, sino que el ordenador reconoce los trazados y los proyecta, dando la sensación de que efectivamente el profesor está pintando sobre la pizarra. Esto conlleva una serie de ventajas. Según lo sofisticado que sea el software utilizado, el profesor/ponente puede hacer zoom sobre los elementos proyectados, dotar de física a sus dibujos, proyectar imágenes guardadas en el ordenador o visualizar aplicaciones, por ejemplo, de videoconferencia. Además se pueden guardar capturas de la pizarra, para enviarla a los alumnos/asistentes o utilizarla en otra clase/reunión. Un sistema de pizarra interactiva está compuesto por un ordenador, un proyector y la propia pizarra.

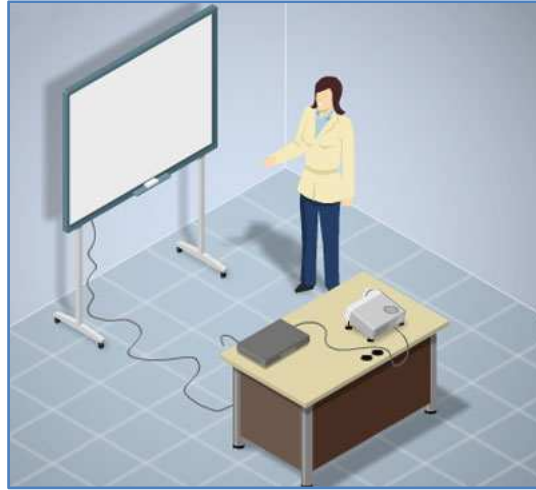


Figura 2.2.4.2.1. Pizarra interactiva convencional

Esta aplicación parte de la idea de la pizarra interactiva pero para implementarla utilizando un *Wii mote* y un dispositivo señalador con forma de bolígrafo con un led infrarrojo en la punta. Lo que se hace es proyectar con normalidad la imagen del ordenador usando un proyector y una pantalla normal, elementos disponibles en gran cantidad de oficinas y centros de enseñanza, y apuntando el *Wii mote* hacia la pantalla, éste es capaz de reconocer la posición de la fuente de luz infrarroja realizando un previo calibrado para calcular el ángulo y distancia del *Wii mote* a la pantalla, además de su tamaño. El siguiente esquema muestra el funcionamiento de la aplicación.



Figura 2.2.4.2.2. Esquema de aplicación de pizarra interactiva

Gracias a esta aplicación, el coste de tener una pizarra digital interactiva es en muchos casos prácticamente o totalmente nulo, ya que disponiendo de un proyector y ordenador, tan sólo es necesario adquirir un *Wiimote* y comprar o construir el bolígrafo emisor de luz infrarroja.

2.2.4.3. Aplicación de realidad virtual

El autor de la aplicación explica el concepto de esta aplicación haciendo un símil entre las aplicaciones/juegos de ordenador y una fotografía enmarcada o una ventana. El ser humano, percibe una fotografía de forma idéntica desde cualquier ángulo. Por mucho que se mueva alrededor de ella siempre la verá igual. Mientras que si mira a través de una ventana podrá ver distintas zonas del exterior según su posición. Los esquemas mostrados a continuación muestran gráficamente la idea:

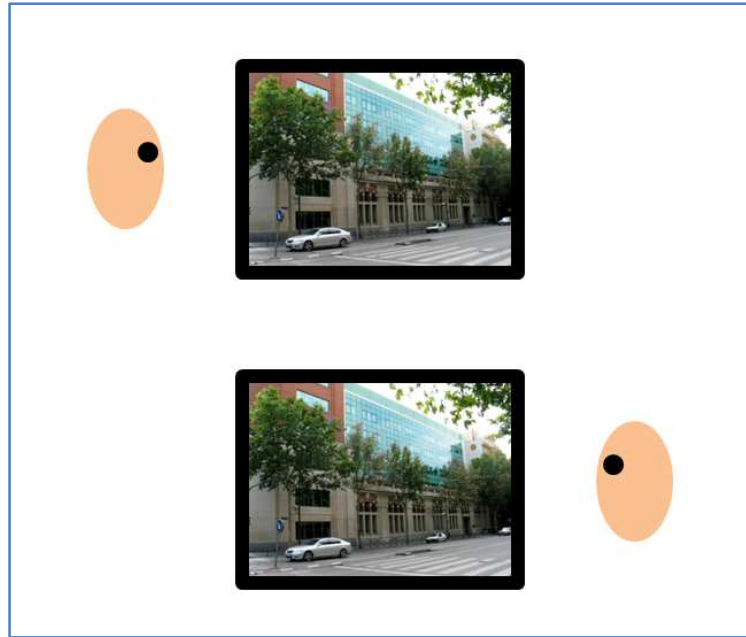


Figura 2.2.4.3.1. Usuario mirando una fotografía

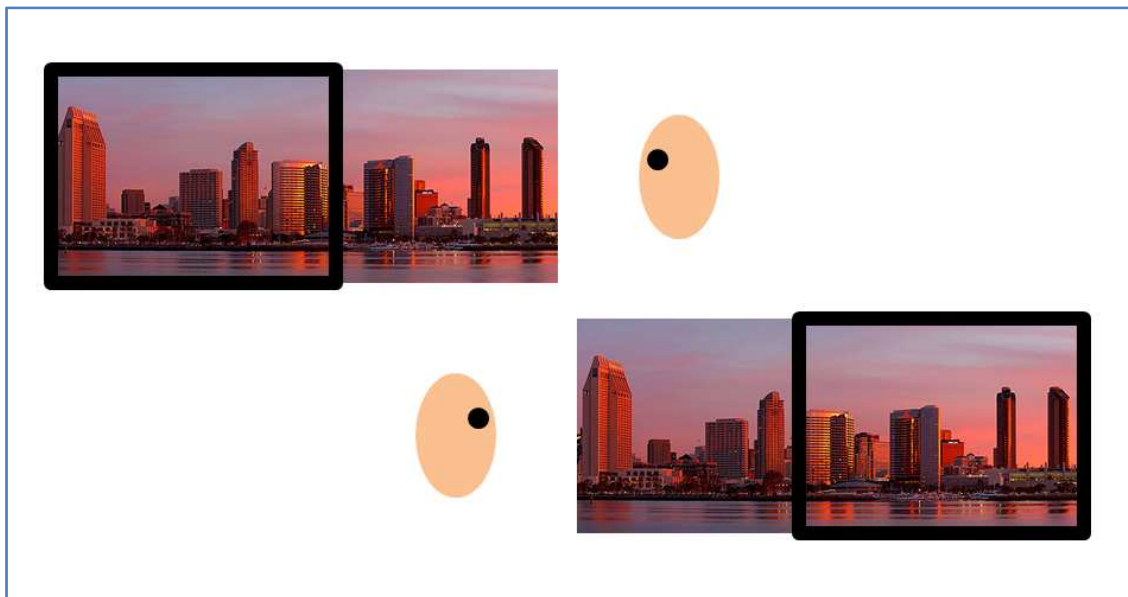


Figura 2.2.4.3.2. Usuario mirando por una ventana

Esta aplicación actúa de una forma similar a una ventana, es decir, muestra distintas áreas de una imagen o espacio tridimensional según hacia dónde estemos mirando en lugar de basarse únicamente en las acciones de teclado, ratón o dispositivo de juego. Para ello se requiere la proyección de luz infrarroja hacia el *Wiimote* desde nuestra cabeza, utilizando por ejemplo una gorra. De esta manera el programa es capaz de interpretar la posición de la misma y mostrar distintas imágenes según hacia donde

estemos mirando y nuestra situación. El siguiente esquema muestra el funcionamiento de la aplicación:

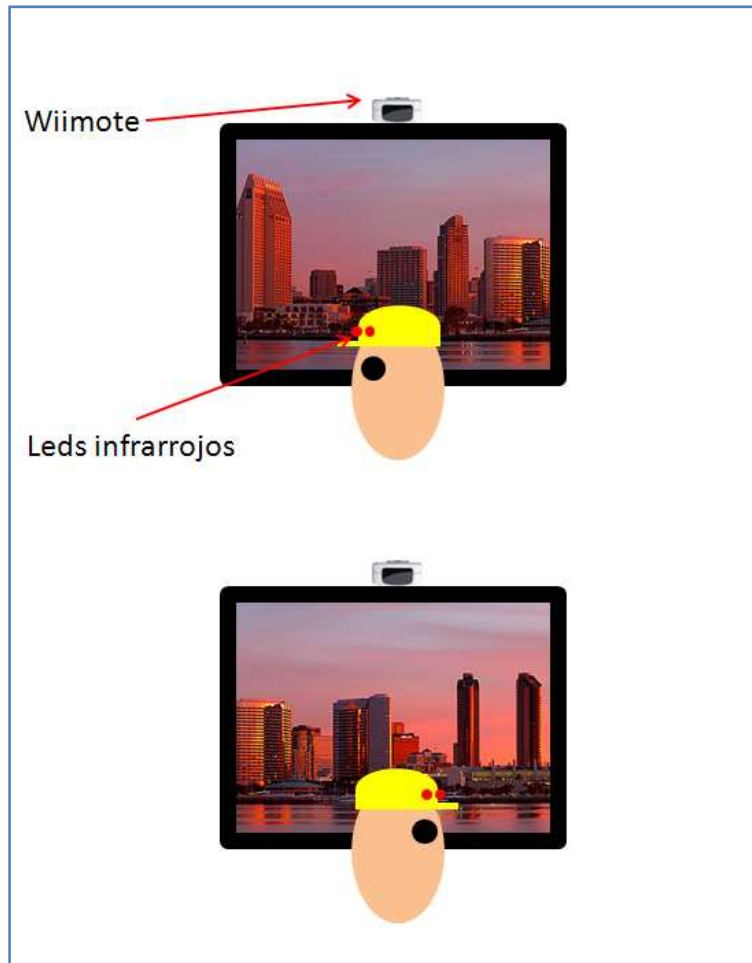


Figura 2.2.4.3.3. Esquema aplicación realidad virtual

Johnny Chung Lee consigue con esta aplicación una implementación barata y efectiva de realidad virtual utilizando nuestros movimientos de cabeza y nuestra situación para mostrar distintas escenas.

Capítulo 3

Desarrollo de

Software para

Wiimote

3.1. Creación de Scripts para GlovePIE

En este apartado de la memoria se describen los diversos desarrollos llevados a cabo para la integración del *Wiimote* con el programa *Meedio* para el control del mismo, utilizando para ello GlovePIE. Cabe recordar que GlovePIE es un intérprete de *scripts* escritos en un lenguaje propio, compatible con diversos dispositivos, entre ellos *Wiimote*, y que no hace más que asignar acciones de ratón y teclados a botones y movimientos del dispositivo.

Dado que se han realizado realizado tres *scripts* GlovePIE diferentes, los separarán en tres subapartados:

1. Mando a distancia convencional.
2. Dispositivo apuntador.
3. Desplazamiento por aceleración.

3.1.1. Mando a distancia convencional

Se creará un script para el uso del *Wiimote* como un mando a distancia convencional, esto es, sin aprovechar sus posibilidades de movimiento, controlando *Meedio* únicamente usando los botones del dispositivo.

3.1.1.1. Requisitos

Realizar un mapeo de los botones de *Wiimote* para el control de *Meedio* con una distribución lo más intuitiva posible accediendo a las funciones más importantes del mismo. Asimismo, deben aprovecharse todas las teclas del *Wiimote*. Las acciones requeridas para un manejo suficiente del programa, es decir, de poder utilizar todas sus posibilidades aunque sea necesario prescindir de atajos por la limitación del número de botones, son las siguientes:

- **Desplazamiento** por menús, tanto verticales como horizontales.
- **Selección de cualquier opción o módulo** (vídeo, música, domótica,...) del programa presentada a través de los menús.
- Posibilidad de **volver a la pantalla anterior** así como al menú principal.
- **Subir y bajar el volumen.**
- En caso de sobrar botones para las funciones anteriores, utilizar los restantes para el acceso directo a las pantallas que se consideren más importantes como el menú principal, el menú de vídeo, de audio, de domótica, ...

Además, se encenderá el Led número 1 del *Wiimote* para que el usuario sepa que está activo y ejecutando esta aplicación (la número 1)

3.1.1.2. Diseño

El diseño de la estructura de este *script* es bastante trivial, ya que se basa únicamente en el *mapeo* de botones *Wiimote/Teclado*, es decir, en relacionar botones del *Wiimote* con botones de teclado, y la orden de encendido del Led 1.

Tras hacer un estudio de las distintas posibilidades de distribución de las teclas a cada una de las funciones, se ha decidido adoptar la siguiente configuración:

- **D-Pad:** Movimiento por menús
- **A:** Seleccionar opción
- **B (gatillo):** Volver
- **Home:** Menú principal
- **+ / -:** Volumen UP / DOWN
- **1:** Menú de música
- **2:** Menú de vídeo

3.1.1.3. Implementación

El código del *script* está compuesto por dos secciones. En la primera se enciende el Led 1 del *Wiimote*, mientras que en la segunda se realizan las asignaciones de botones o mapeo.

3.1.1.4. Utilización

El proceso para utilizar el *script* creado es el siguiente:

1. Si fuera necesario, ajustar los mapeos de botones en la configuración de *Meedio*. Sólo será necesario si no se utiliza la configuración por defecto, ya que la asignación realizada en el *script* está realizada en base a ella.
2. Emparejar *Wiimote* y PC mediante el gestor de conexiones *Bluetooth* de *Windows*, el suministrado por el fabricante de nuestro *dongle bluetooth* o bien mediante *BlueSoleil*.
3. Ejecutar *GlovePIE* y abrir el *script*. Ejecutarlo pulsando “*Run*”.
4. Abrir *Meedio* y manejar según la configuración mencionada anteriormente.

3.1.2. Dispositivo apuntador

Este *script* tiene por objetivo ampliar el comportamiento del anterior, pero añadiendo la posibilidad de navegar por los menús de *Meedio* utilizando el *Wiimote* como apuntador. Para ello se utilizará la función de visión de infrarrojos, lo que consecuentemente requerirá colocar una fuente de luz infrarroja en línea directa con el usuario (normalmente encima o debajo de la televisión o monitor)

3.1.2.1. Requisitos

Por una parte se requerirá que el código permita mover el cursor del ratón utilizando para ello las posibilidades de movimiento por luz infrarroja del *Wiimote*. Por otra, hacer un mapeo de botones que ayude al control por movimiento, dado que este no será suficiente para el control de la aplicación. La asignación debe ser lo más intuitiva posible y al menos deben poderse realizar las siguientes acciones:

- **Desplazamiento** por menús, tanto verticales como horizontales. Si bien se desea utilizar el movimiento del mando para elegir cada opción del menú, en ocasiones, cuando existan menús largos, no será posible ver todas las opciones.
- **Selección de cualquier opción o módulo** (vídeo, música, domótica,...) del programa presentada a través de los menús.
- Posibilidad de **volver a la pantalla anterior** así como al menú principal.
- **Subir y bajar el volumen.**
- En caso de sobrar botones para las funciones anteriores, utilizar los restantes para el acceso directo a las pantallas que se consideren más importantes, como el menú principal, el menú de vídeo, de audio, de domótica, ...

Se requiere también el encendido del Led número 2 del *Wiimote* para que el usuario sepa que está activo y ejecutando esta aplicación (la segunda).

3.1.2.2. Diseño

El *script* estará compuesto de tres partes diferenciadas:

1. **Función de movimiento del ratón** según las coordenadas que transmita el *Wiimote*. Para que resulte más fácil la selección de una opción en el programa pulsando para ello un botón, deben ignorarse los movimientos del mando durante un tiempo de 250 ms desde la pulsación del botón.
2. **Mapeo de botones auxiliares**. Se ha decidido utilizar la misma configuración que en el anterior *script* para poder acceder a todas las funciones del programa de la manera más intuitiva y fácil de recordar posible:
 - **D-Pad:** Movimiento por menús
 - **A:** Seleccionar opción
 - **B (gatillo):** Volver
 - **Home:** Menú principal
 - **+ / -:** Volumen UP / DOWN
 - **1:** Menú de música
 - **2:** Menú de vídeo
3. **Encendido del Led número 2.**

3.1.2.3. Implementación

La implementación de los puntos 2 y 3 del apartado anterior es trivial. El primer punto requiere la utilización de una variable que controle la pulsación de los botones. Si dicha variable indica que se está pulsando un botón, no deben reflejarse los movimientos del dispositivo en un periodo de 250 ms.

3.1.2.4. Utilización

La forma de utilizar este *script* es igual a la de cualquier otro en *GlovePIE* aunque puede requerir la configuración de algunas teclas en *Meedio*:

1. Si fuera necesario, ajustar los mapeos de botones en la configuración de *Meedio*. Sólo será necesario si no se utiliza la configuración por defecto, ya que la asignación realizada en el *script* está realizada en base a ella.
2. Emparejar *Wiimote* y PC mediante el gestor de conexiones *Bluetooth* de *Windows*, el suministrado por el fabricante de nuestro *dongle bluetooth* o bien mediante *BlueSoleil*.
3. Ejecutar *GlovePIE* y abrir el *script*. Ejecutarlo pulsando “*Run*”.
4. Abrir *Meedio* y manejar según la configuración mencionada anteriormente.

3.1.3. Desplazamiento por aceleración

Este último *script* está orientado al manejo de GlovePIE utilizando las funciones de aceleración del dispositivo en lugar de utilizarlo como apuntador. La gran ventaja de utilizar esta función frente a la anterior es que no requiere el uso de la barra de luz infrarroja ni la línea de visión directa del usuario con ningún otro dispositivo.

3.1.3.1. Requisitos

En primer lugar se requerirá que el código permita desplazarse por los menús utilizando para ello las funciones de aceleración del mando, es decir, que agitándolo hacia arriba suba una opción en el menú de *Meedio*, agitándolo hacia abajo, baja una opción, y lo mismo para derecha e izquierda. Por otra parte, hacer un mapeo de botones que ayude al control por movimiento, dado que este no será suficiente para el control de la aplicación. La asignación debe ser lo más intuitiva posible y al menos deben poderse realizar las siguientes acciones:

- **Selección de cualquier opción o módulo** (vídeo, música, domótica,...) del programa presentada a través de los menús.
- Posibilidad de **volver a la pantalla anterior** así como al menú principal.
- **Subir y bajar el volumen.**
- En caso de sobrar botones para las funciones anteriores, utilizar los restantes para el acceso directo a las pantallas que se consideren más importantes como el menú principal, el menú de vídeo, de audio, de domótica, ...

Se requiere también el encendido el Led número 3 del *Wiimote* para que el usuario sepa que está activo y ejecutando esta aplicación (la tercera).

3.1.3.2. Diseño

El *script* estará compuesto de tres partes diferenciadas:

1. **Función de desplazamiento por menús:** según la aceleración del mando hacia arriba, abajo, derecha o izquierda. No debe reconocerse más de un

desplazamiento por cada *250ms* para evitar efecto “rebote” al volver la mano a la posición original.

2. **Mapeo de botones auxiliares.** En este caso, dado que agitando el mando el usuario se puede desplazar por todos los menús, no se requiere utilizar el D-Pad o cruceta para el movimiento, si no que se le dará funciones de accesos directos a distintos módulos:

- **Arriba:** Música
- **Abajo:** DVD
- **Izquierda:** Vídeo
- **Abajo:** Imágenes
- **A:** Seleccionar opción
- **B (gatillo):** Volver
- **Home:** Menú principal
- **+ / -:** Volumen UP / DOWN
- **1:** Menú de música
- **2:** Menú de vídeo

3. **Encendido del Led número 3.**

3.1.3.3. Implementación

La implementación de la segunda y tercera parte es trivial. La primera requiere la utilización de una variable para controlar el tiempo entre desplazamientos y la simulación de que se pulsa una determinada tecla (arriba, abajo, izquierda y derecha) al detectar un determinado movimiento, que se realiza comprobando las variables de aceleración del *Wiimote*.

3.1.3.4. Utilización

La forma de utilizar este *script* es igual a la de cualquier otro en *GlovePIE* aunque puede requerir la configuración de algunas teclas en *Meedio*:

1. Si fuera necesario, ajustar los mapeos de botones en la configuración de *Meedio*. Sólo será necesario si no se utiliza la configuración por defecto, ya que la asignación realizada en el *script* está realizada en base a ella.
2. Emparejar *Wiimote* y PC mediante el gestor de conexiones *Bluetooth* de *Windows*, el suministrado por el fabricante de nuestro *dongle bluetooth* o bien mediante *BlueSoleil*.
3. Ejecutar *GlovePIE* y abrir el script. Ejecutarlo pulsando “*Run*”.
4. Abrir *Meedio* y manejar según la configuración mencionada anteriormente.

3.2. Desarrollos basados en movimientos del usuario

En este apartado se engloban varias aplicaciones desarrolladas que se basan en los movimientos del propio usuario utilizando el *Wii mote* como dispositivo que hace un seguimiento de los mismos.

El control de las aplicaciones debe poder realizarse de dos maneras:

- Mediante movimientos de un dedo.
- Utilizando un puntero de luz infrarroja.

El siguiente esquema muestra cómo debe manejarse la aplicación mediante el movimiento del dedo índice (aunque puede utilizarse cualquier otro):

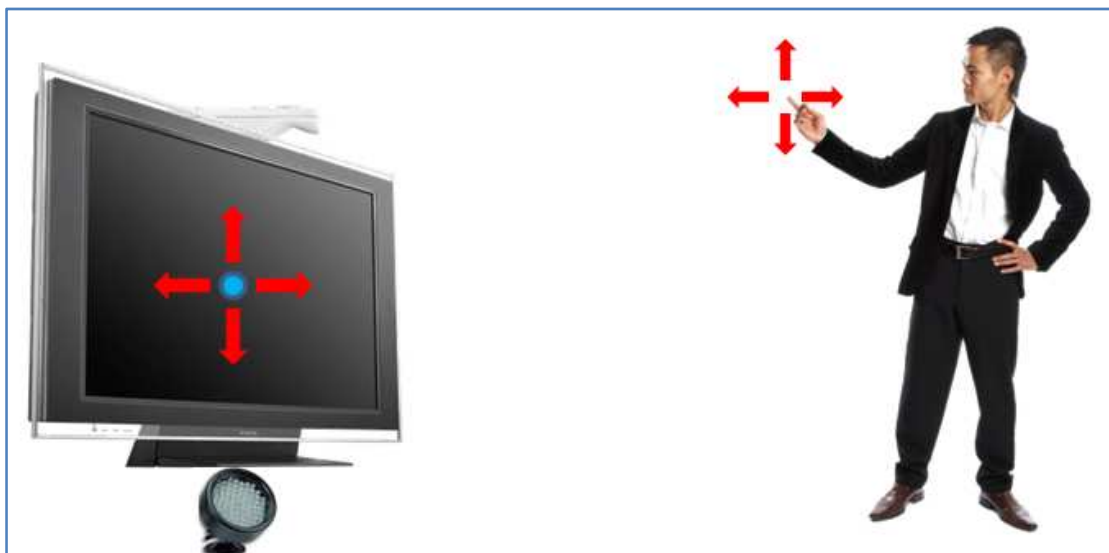


Figura 3.2.1. Esquema funcionamiento aplicación WiiClick con el dedo

Como se puede observar, esta aplicación requiere varios elementos para funcionar:

- **Wii mote:** Se coloca enfrente del usuario (normalmente encima o debajo del monitor, televisión o pantalla donde se proyecta la imagen) y gracias a su cámara de infrarrojos seguirá el movimiento del dedo del usuario.

- **Luz infrarroja:** Se requiere proyectar luz infrarroja hacia el usuario para que éste pueda reflejarla y que el *Wiimote* haga el seguimiento de sus movimientos. Se utilizará un array de leds como el mostrado en el punto 2.1.3.
- **Material reflectante** (no visible en la figura): Para poder reflejar mediante nuestro dedo la luz infrarroja del dispositivo proyector, se debe utilizar algún material reflectante como la cinta comentada en el punto 2.1.3.
- **Pantalla y ordenador:** Resulta evidente que se necesita un monitor, televisión o pantalla y proyector para poder ver el escritorio/aplicaciones utilizadas y nuestras acciones, al igual que un ordenador para ejecutar el programa. A ellos se conectará mediante *bluetooth* el *Wiimote*.

Este otro esquema muestra cómo debe funcionar la aplicación utilizando un puntero de luz infrarroja en lugar del dedo:

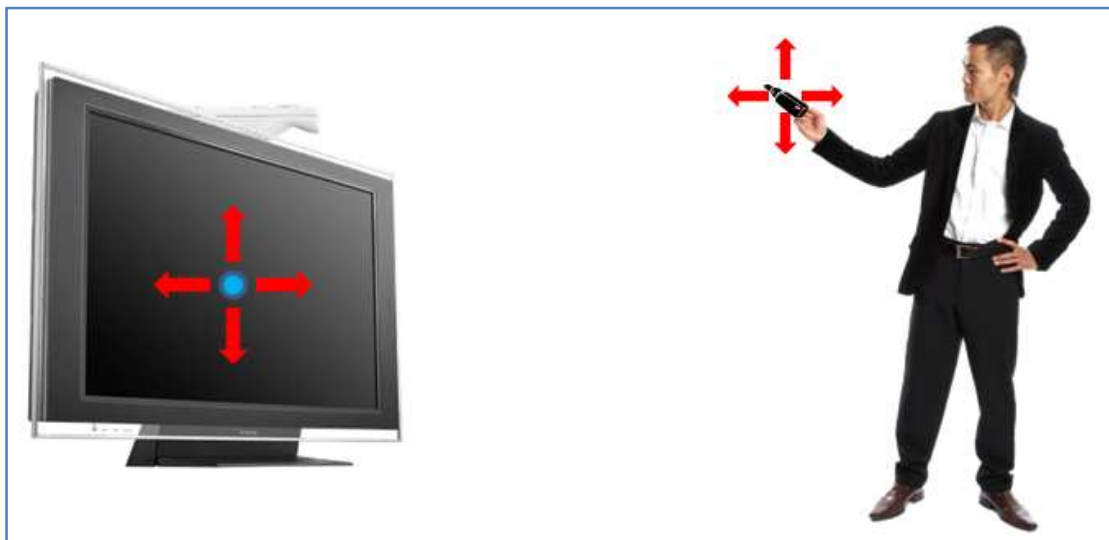


Figura 3.2.2. Esquema funcionamiento aplicación WiiClick utilizando un puntero

Los elementos necesarios para utilizar la aplicación mediante este método son los siguientes:

- **Wiimote:** Se coloca enfrente del usuario (normalmente encima o debajo del monitor, televisión o pantalla donde se proyecta la imagen) y gracias a su cámara de infrarrojos seguirá el movimiento del puntero de luz infrarroja.

- **Puntero de luz infrarroja:** Se utilizará un dispositivo, normalmente con forma de bolígrafo por su comodidad, para emitir luz infrarroja. En el apartado 2.1.4. se describe más a fondo las características del mismo.

- **Pantalla y ordenador:** Resulta evidente que se necesita un monitor, televisión o pantalla y proyector para poder ver el escritorio/aplicaciones utilizadas y nuestras acciones, al igual que un ordenador para ejecutar el programa. A ellos se conectará mediante *bluetooth* el *Wiimote*.

3.2.1. Aplicación para el control del cursor (*WiiClick*)

Esta es la primera aplicación como tal creada para el proyecto, aunque en realidad se trata de una aplicación prototipo, que, aunque perfectamente funcional, está pensada para servir como base a las desarrolladas posteriormente.

Se ha utilizado como idea base la aplicación de *Johnny Chung Lee* de seguimiento de dedos comentada en el punto 2.2.4.1. de este proyecto, para mover el puntero del ratón y ejecutar acciones básicas de *click* y *click* derecho que dan nombre a la aplicación.

3.2.1.1. Requisitos

Para esta aplicación se han definido cuatro requisitos:

- **REQ-3.2.1.1 Movimiento del cursor:** Debe poderse mover el cursor (puntero del ratón) en un entorno *Windows* utilizando para ello el movimiento de un dedo en el aire, o bien utilizando un puntero de luz infrarroja similar a los señaladores láser.
- **REQ-3.2. 1.2 Calibrado de puntero:** La aplicación debe poderse utilizar desde cualquier distancia del usuario respecto del *Wii mote*, siempre que las características del mismo y de la potencia de la fuente de luz infrarroja utilizada lo permitan.
- **REQ-3.2.1.3 Función de click izquierdo:** Debe estudiarse la manera de implementar la función de realizar la acción equivalente a pulsar el botón izquierdo del ratón y codificarla.
- **REQ-3.2.1.4 Función de click derecho:** Igualmente debe estudiarse la manera de que se pueda ejecutar la acción equivalente a la pulsación del botón derecho del ratón. En conjunción con el requisito anterior, será posible manejar cualquier aplicación básica de *Windows* sin utilizar un ratón.

3.2.1.2. Diseño

En este apartado se estudiará la manera de implementar cada requisito, y una vez lo se haga con todos, diseñar la estructura general del programa para juntar todo lo anterior.

3.2.1.2.1. REQ-3.2.1.1 Movimiento del ratón

Para controlar el movimiento del cursor mediante la cámara de infrarrojos del *Wiimote*, se deben leer las coordenadas proporcionadas por ella cada cierto tiempo.

El movimiento del cursor en función de las coordenadas que ofrece la cámara no es trivial. La cámara tiene una resolución de 1024x768 píxeles, mientras que el monitor utilizado puede tener la misma o una distinta. Es por ello que se requiere una transformación para poder abarcar toda el área visible.

El siguiente esquema muestra la resolución del *Wiimote* y de varios monitores así como sus sistemas de coordenadas:

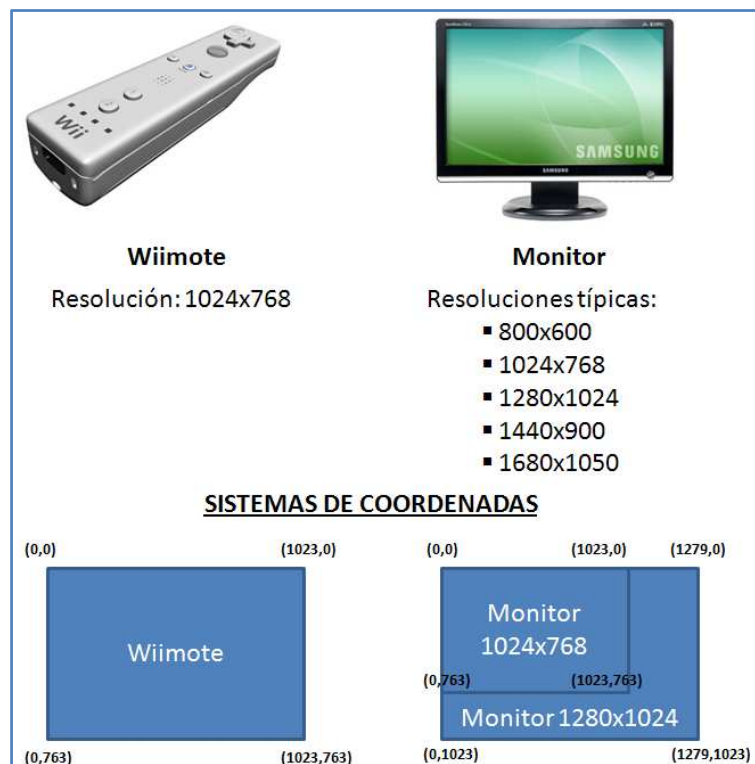


Figura 3.2.1.2.1.1. Resoluciones y sistemas de coordenadas de *Wiimote* y monitores

El siguiente esquema muestra la implementación del requisito:

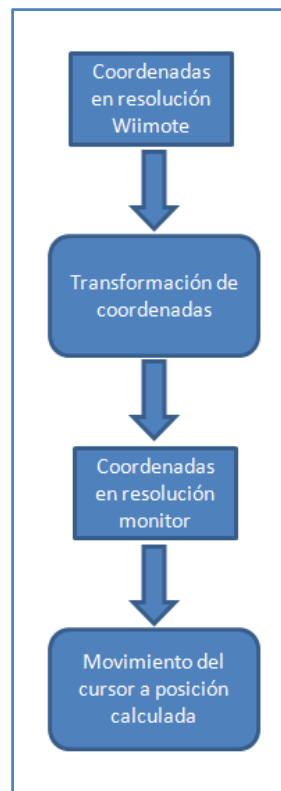


Figura 3.2.1.2.1.2. Transformación de coordenadas

3.2.1.2.2. REQ-3.2.1.2 Calibrado del puntero

Este requisito exige que se pueda utilizar el programa a cualquier distancia del *Wiimote*, siempre que las posibilidades del mismo y de la luz utilizada lo hagan posible. El problema está en el área de visión de la cámara (según un ángulo de 45°) abarcada según la distancia del usuario al *Wiimote*. Dependiendo del área, se requerirá un movimiento mayor o menor del brazo para mover el cursor. La siguiente figura muestra este problema:

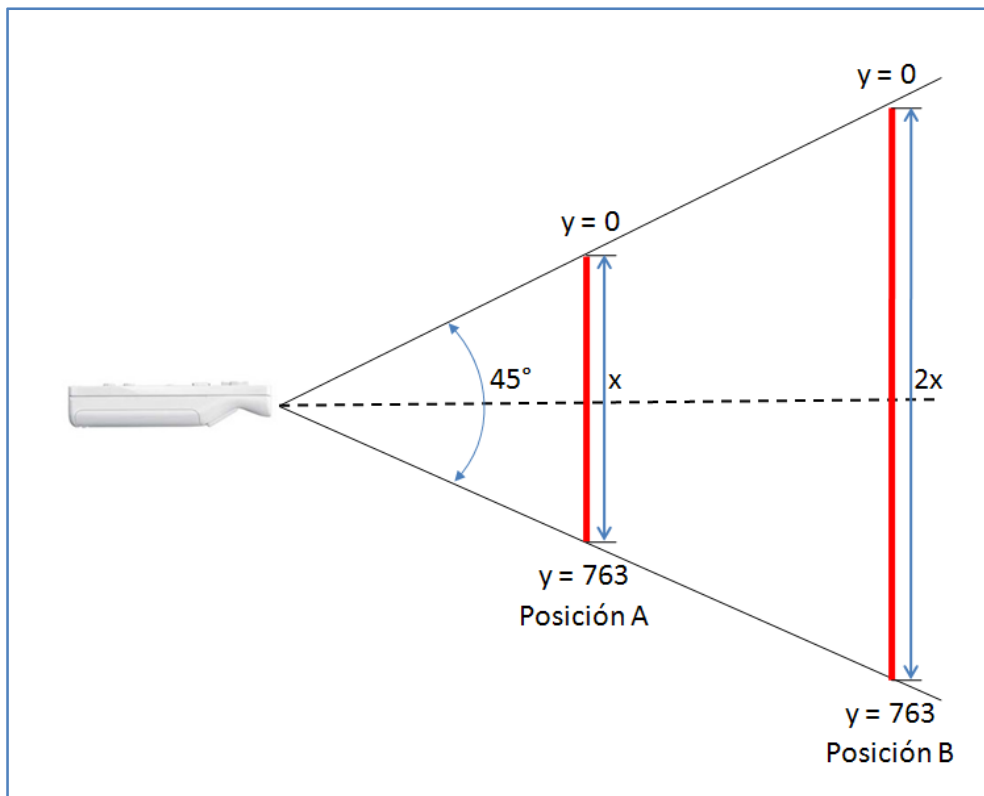


Figura 3.2.1.2.2.1. Área de visión según distancia

Cómo se puede apreciar, dependiendo de la posición en la que sitúe el usuario, el área abarcada es distinta. En el ejemplo se ha mostrado la variación de la posición de los extremos de la coordenada y leída por el Wiimote. El área abarcada por la posición B, a una distancia del doble del *Wiimote* respecto a la posición A, es también el doble.

Este problema requiere también una transformación en las coordenadas. Como base se tomarán las coordenadas ya calculadas para el REQ-3.2.1.1. La transformación se realiza de manera que el resultado sea un área igual a la abarcada en una distancia al *Wiimote* de 30cm. Es decir, se comprobará únicamente la parte central de todo el área que contemple cada caso según la distancia y añadiremos sensibilidad. Para entender mejor este concepto se puede observar la siguiente figura:

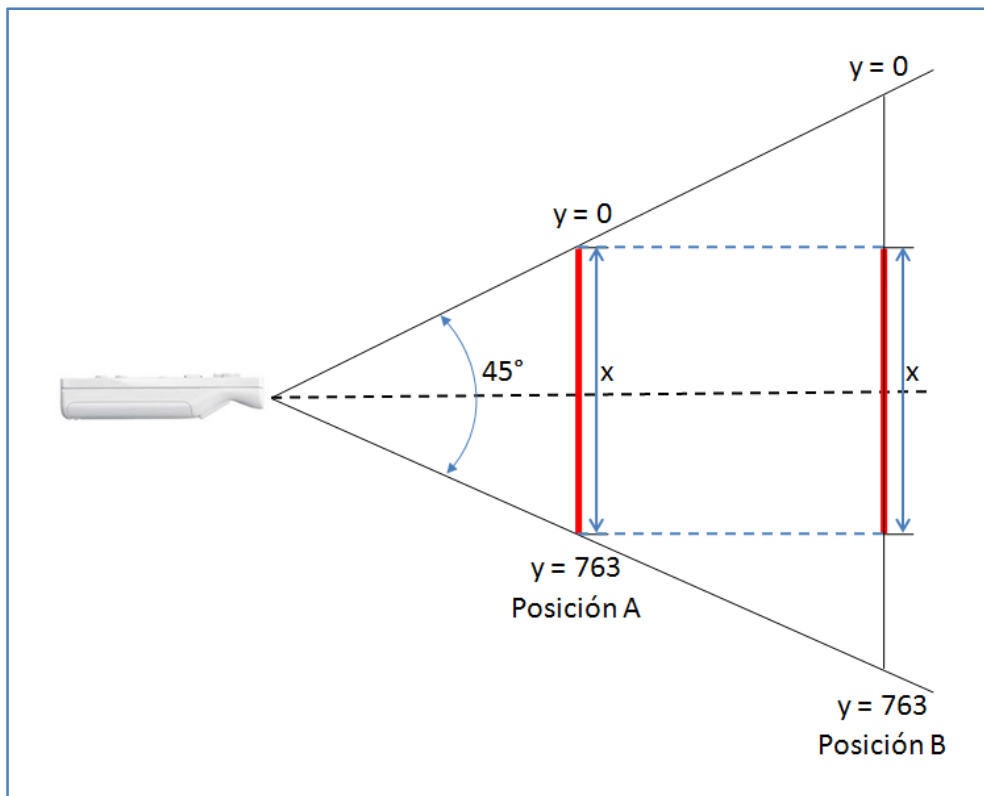


Figura 3.2.1.2.2.2. Área de visión transformada

Aunque las coordenadas captadas por el *Wiimote* siguen siendo las mismas, tan sólo se observa la parte central y se le añade un multiplicador (sensibilidad) para poder desplazarnos por toda la pantalla utilizando un área menor.

El siguiente esquema muestra la implementación del requisito:

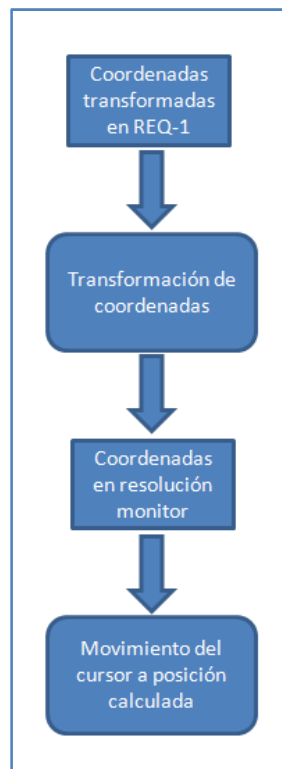


Figura 3.2.1.2.2.3. Calibrado del puntero

Debido a la necesidad de aplicar esta transformación a las coordenadas calculadas en la anterior, no se mueve el puntero dos veces, lo que provocaría un desagradable efecto, si no que se realizarían ambas transformaciones consecutivamente en la misma iteración para luego mover el cursor a la posición calibrada.

3.2.1.2.3. REQ-3.2.1.3 Función de click izquierdo

Para la implementación de esta función se ha decidido interpretar que el usuario ha querido ejercer la acción de click derecho cuando la luz desaparezca y vuelva a aparecer en un intervalo menor a 1 seg. En caso de que el usuario esté usando el dedo para desplazarse, basta con que lo encoja y vuelva a estirarlo en dicho intervalo. Si por el contrario utiliza un puntero, debe apagar y encender la luz en el mismo intervalo.

3.2.1.2.4. REQ-3.2.1.4 Función de click derecho

Esta función se implementará de la misma manera que la de click izquierdo pero utilizando un intervalo que va de 1 a 3 segundos.

3.2.1.2.5. Estructura general del programa

Por último se muestra el diagrama de flujo del programa juntando todo lo explicado en los apartados anteriores.

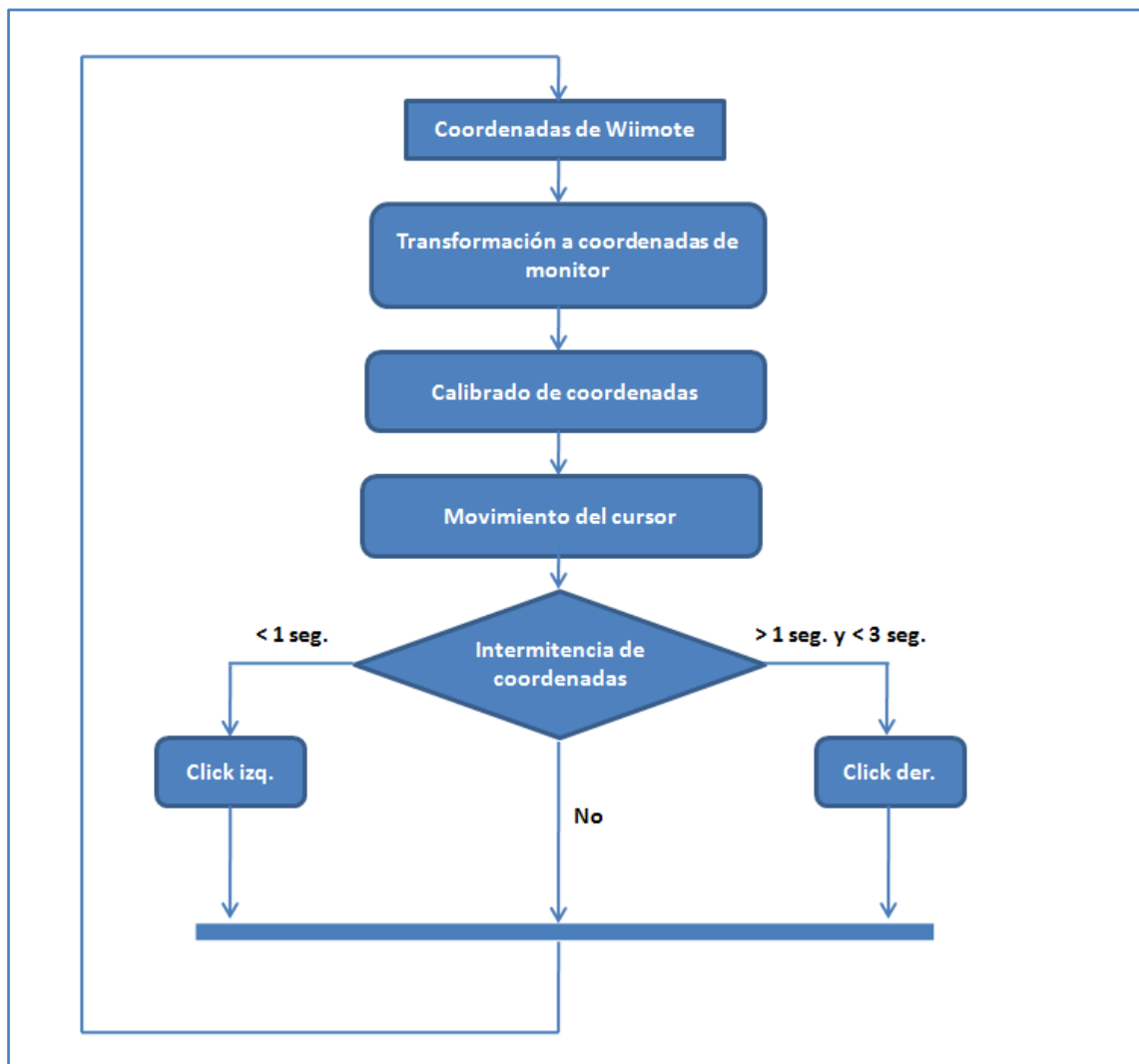


Figura 3.2.1.2.5.1. Diagrama de flujo de WiiClick

3.2.1.3. Implementación

Para codificar esta aplicación se han utilizado los siguientes recursos software:

- **Microsoft Visual Studio C# 2008 Express:** Esta herramienta de desarrollo está integrada con .NET, y existe multitud de proyectos de ejemplo creados en la herramienta para las librerías utilizadas. Además es 100% gratuita.
- **Librerías *Wiimote* para C# de *Brian Peek*:** Estas librerías están bastante documentadas y existen múltiples ejemplos disponibles. Debido a los requisitos de constante muestreo y las diversas transformaciones matemáticas, se requiere un lenguaje potente y rápido como es C# para que la aplicación se ejecute de la forma más fluida posible.

La lectura de los informes generados por el *Wiimote* que informan de las lecturas realizadas por el mismo se implementa mediante gestión de eventos, evitando así la espera activa (o técnica de *polling*). Se generará un evento cada vez que el *Wiimote* tenga disponible un nuevo informe. De él se extraerá la información necesaria para la transformación de coordenadas.

Tal y como se explicó detalladamente en el apartado de diseño, se deben realizar dos transformaciones de coordenadas:

- **Conversión coordenadas *Wiimote* a área visible del monitor:** Se utilizan las siguientes variables y fórmulas para esta transformación:

Variables:

- **anMonitor:** Resolución del monitor a lo ancho (en píxeles)
- **alMonitor:** Resolución del monitor a lo alto (en píxeles)
- **xWiimote:** Valor en eje x de la luz detectada por el *Wiimote*.
- **yWiimote:** Valor en eje y de la luz detectada por el *Wiimote*.
- **xMonitor:** Valor en eje x de la posición del cursor en el monitor.
- **yMonitor:** Valor en eje y de la posición del cursor en el monitor.

Fórmulas:

- **xMonitor** = $\text{anMonitor} - \text{anMonitor} * \text{xWiimote} / 1023$
- **yMonitor** = $\text{alMonitor} - \text{alMonitor} * \text{yWiimote} / 767$

- **Calibrado de las coordenadas según distancia del usuario al Wiimote:** Por otra parte, hay que reducir el área de visión del *Wiimote* a la abarcada por el mismo cuando el usuario se sitúa a 30 cm. Se utilizan las siguientes variables, fórmulas y algoritmos para conseguirlo:

Variables:

- **distancia:** Distancia desde *Wiimote* hasta el usuario.
- **distancia_nr:** Distancia normalizada.
- **anMonitor:** Resolución del monitor a lo ancho (en píxeles)
- **alMonitor:** Resolución del monitor a lo alto (en píxeles)
- **xTr:** Coordenada *x* transformada (REQ-3.2.1.1)
- **yTr:** Coordenada *y* transformada (REQ-3.2.1.1)
- **xMonitor:** Valor en eje *x* de la posición del cursor en el monitor.
- **yMonitor:** Valor en eje *y* de la posición del cursor en el monitor.

Fórmulas:

- **distancia_nr** = $\text{distancia} / 0.3$

Algoritmo:

```

Si (xTr > anMonitor / 2) //Transforma coordenada x
{
    xMonitor = anMonitor / 2 + (xTr - anMonitor / 2) * distancia_nr
}
en otro caso
{
    xMonitor = anMonitor / 2 - (anMonitor / 2 - xTr) * distancia_nr
}

Si (yTr > alMonitor / 2) //Transforma coordenada y
{
    yMonitor = alMonitor / 2 + (yTr - alMonitor / 2) * distancia_nr
}
en otro caso
{
    yMonitor = alMonitor / 2 - (alMonitor / 2 - yTr) * distancia_nr
}

Si (xMonitor < 0) //Ignora el resto del área de x, asignándola a los extremos
{
    xMonitor = 0
}
en otro caso si (xMonitor > anMonitor)

```

```
{
    xMonitor = anMonitor
}

Si (yMonitor < 0) //Ignora el resto del área de y, asignándola a los extremos
{
    xMonitor = 0
}
en otro caso si (yMonitor > alMonitor)
{
    yMonitor = alMonitor
}
```

En cuanto a la interpretación de las órdenes de simulación de click izquierdo y derecho, se utiliza una variable que registra el último momento en que el *Wiimote* detectó luz infrarroja. Se ignoran las reapariciones de luz en un intervalo menor de 100 *ms* para evitar lecturas falsas de click, ya que es imposible que un usuario realice la acción tan rápidamente, y el *Wiimote* tiende a “perder de vista” los puntos durante periodos muy cortos de tiempo, cuando realmente sí que están. Si por el contrario la luz reaparece en un intervalo de 100 a 1000 *ms*, se simulará la pulsación del botón izquierdo del ratón, y si va de 1000 a 3000 *ms*, una pulsación del botón derecho.

3.2.1.4. Utilización

La forma de ejecutar la aplicación para el manejo del cursor de *Windows* es bastante sencilla e intuitiva:

- Emparejar *Wiimote* y PC mediante el gestor de conexiones *Bluetooth* de *Windows*, el suministrado por el fabricante de nuestro *dongle bluetooth* o bien mediante *BlueSoleil*.
- Especificar al programa la distancia a la que nos situaremos del *Wiimote* para que pueda calibrar el movimiento del cursor.
- Utilizar un array de leds más un trozo de cinta reflectante en el dedo o bien un puntero con led infrarrojo para realizar movimientos que pueda captar el *Wiimote* y que interprete el programa. Para ejecutar las acciones de click:

- **Click izquierdo:** Si se utiliza el dedo, encogerlo y estirarlo en un intervalo menor a un segundo. Si se utiliza un puntero, apagar y volver a encender la luz en el mismo intervalo de tiempo.
- **Click derecho:** Igual que el derecho pero el intervalo debe ser de entre uno y tres segundos.

Esta forma de interactuar con el sistema se puede utilizar con cualquier aplicación de *Windows*, ya que simula el mismo movimiento y acciones que si se utilizara el ratón.

3.2.2. Aplicación para dibujo por movimiento (*WiiAirBoard*)

Utilizando *WiiClick* como base, se ha desarrollado una variante con interfaz gráfica destinada a ser utilizada en conjunción con programas de dibujo (como *Paint*) o en otros que tengan funciones de dibujo (como *Powerpoint*), de manera que se puedan hacer gestos de dibujo en el aire y que dichos dibujos queden representados en la aplicación deseada. Esta aplicación se ha denominado *WiiAirBoard*.

3.2.2.1. Requisitos

Se han definido los siguientes requisitos para esta aplicación:

- **REQ-3.2.2.1 Captura de trazados del usuario:** Se deben capturar los trazados realizados por el usuario y representarlos en pantalla. Dichos trazados pueden ser realizados mediante el movimiento del dedo en el aire o utilizando una luz infrarroja.
- **REQ-3.2.2.2 Calibrado del puntero:** Al igual que en la aplicación anterior, el usuario debe poder utilizar *WiiAirBoard* desde cualquier distancia respecto al *Wii mote*. El calibrado del puntero debe poderse hacer desde una ventana de configuración.

3.2.2.2. Diseño

En la fase de diseño se estudia la manera en la que se implementará cada módulo y el conjunto de todos ellos. Debido a que los requisitos son bastante amplios se puede considerar que cada uno de ellos es un apartado importante del programa.

3.2.2.2.1. REQ-3.2.2.1 Captura de trazados del usuario

Para poder implementar este requisito se reutiliza el código creado para el movimiento del cursor de la aplicación *WiiClick*, pero adaptándolo para este programa. La adaptación consiste en simular la pulsación continua del botón izquierdo del ratón

mientras se detecte luz infrarroja a través de la cámara del *Wiimote*. Debido a que las aplicaciones de dibujo se basan en la pulsación de dicho botón para la creación de trazados, de esta manera se verán reflejados los trazados realizados por el usuario en el aire.

3.2.2.2.2. REQ-3.2.2.2 Calibrado del puntero

Esta función también será reutilizada del programa *WiiClick*, por lo que el diseño de la misma ya ha sido anteriormente tratado en profundidad.

3.2.2.3. Implementación

Para codificar esta aplicación se han utilizado los siguientes recursos software:

- **Microsoft Visual Studio C# 2008 Express:** Esta herramienta de desarrollo está integrada con .NET, y existen multitud de proyectos de ejemplo creados en la herramienta para las librerías utilizadas. Además es 100% gratuita.
- **Librerías *Wiimote* para C# de *Brian Peek*:** Estas librerías están bastante documentadas y existen múltiples ejemplos disponibles. Debido a los requisitos de constante muestreo y las diversas transformaciones matemáticas, se requiere un lenguaje potente y rápido como es C# para que la aplicación se ejecute de la forma más fluida posible.

La parte de conversión de coordenadas y calibración ya fueron detalladas en la anterior aplicación. Dado que se ha reutilizado el código de la misma, no ha habido cambios en la implementación.

Lo único que se ha cambiado en esta aplicación son los efectos producidos por la detección de trazados del usuario, ya que ahora mientras se mueve el cursor por la pantalla, se debe simular que se mantiene pulsado el botón izquierdo del ratón. Para evitar que se simule dicha pulsación incluso durante el movimiento del propio ratón, se controla la presencia o no de luz infrarroja.

3.2.2.4. Utilización

La forma de ejecutar *WiiAirBoard* bastante sencilla e intuitiva:

- Emparejar *Wii mote* y PC mediante el gestor de conexiones *Bluetooth* de *Windows*, el suministrado por el fabricante de nuestro *dongle bluetooth* o bien mediante *BlueSoleil*.
- Calibrar el programa en función de la distancia del usuario al *Wii mote*. Esta calibración permitirá que el usuario no deba realizar movimientos muy extensos, pero a la vez sacrificará la resolución del dibujo. Si se desea una calidad óptima del dibujo se recomienda configurar una distancia de 0,3 metros, aunque realmente se dibuje a una distancia mayor.
- Utilizar un array de leds más un trozo de cinta reflectante en el dedo o bien un puntero con led infrarrojo para realizar movimientos que pueda captar el *Wii mote* y que interprete el programa.
- Arrancar el programa de dibujo deseado.

A partir de ese momento se puede dibujar sobre el lienzo cuando proyectemos luz hacia el *Wii mote*. Si deseamos movernos a otra parte del lienzo basta con apagar la luz del puntero o doblar el dedo en caso de usar reflectante para que el *Wii mote* deje de recibir luz, movernos a la parte que queramos, y volver a emitir luz.

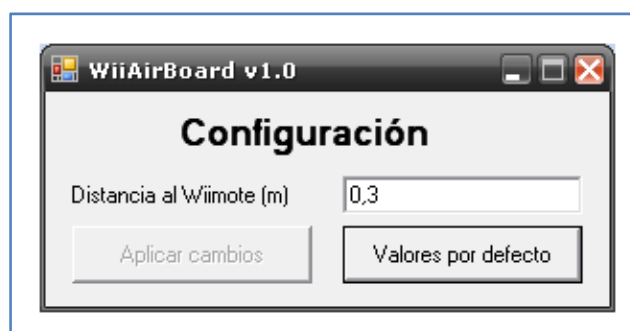


Figura 3.2.2.4.1. Ventana de configuración de *WiiAirBoard*

3.2.3. Aplicación de reconocimiento de gestos (*WiiGestures*)

WiiGestures es una aplicación basada en *WiiClick*, que hace un seguimiento de la mano del usuario y reconoce ciertos gestos a los que asigna ciertas acciones configurables, como puede ser simular un *click* del ratón o la pulsación de una determinada tecla. La aplicación se ejecutará en segundo plano, por lo que podrá utilizarse para manejar cualquier otra.

3.2.3.1. Requisitos

Se han definido los siguientes requisitos obligatorios para esta aplicación:

- **REQ-3.2.3.1 Movimiento del cursor:** Al igual que en *WiiClick*, debe ser capaz de mover el cursor del ratón ya sea moviendo un dedo en el aire o mediante un puntero de luz infrarroja.
- **REQ-3.2.3.2 Calibrado del puntero:** También debe copiarse esta funcionalidad de *WiiClick*. El usuario debe poder utilizar *WiiGestures* desde cualquier distancia respecto al *Wiimote*.
- **REQ-3.2.3.3 Reconocimiento de gestos:** La aplicación debe ser capaz de reconocer diversos gestos del usuario según las lecturas proporcionadas por el *Wiimote*:
 - **Trazados horizontales:** Detectando el sentido del trazado, hacia la izquierda o hacia la derecha.
 - **Trazados verticales:** También debe detectar el sentido, hacia arriba o hacia abajo.
 - **Trazados diagonales:** Debe reconocer diagonales tanto inclinadas hacia la derecha como hacia la izquierda y en ambos sentidos, existiendo un total de cuatro tipos distintos.

- **Gesto de selección de opción:** Este gesto se detectará cuando el usuario haga un número de parpadeos con la luz y simulará un click izquierdo de ratón. Dicho número de parpadeos será configurable.
- **REQ-3.2.3.4 Creación de interfaz de usuario para configuración:** El programa permitirá, a través de formularios, configurar todas las variables importantes del mismo. Se debe poder configurar la misma de manera que cada gesto simule la pulsación de una determinada tecla del ratón o del teclado según las necesidades del usuario, ya que esta aplicación está pensada para ejecutarse en segundo plano, y las necesidades variarán según qué aplicación se desee controlar mediante gestos. Por otra parte se deben poder modificar otros parámetros como la distancia del usuario al *Wiimote*, el número de parpadeos necesarios para el gesto “selección de opción”, y todos los márgenes implicados en la detección de gestos.

Como principal novedad respecto a *WiiClick*, deben reconocerse diversos gestos del usuario, enumerados en el requisito REQ-3.2.3.3, se muestran de forma gráfica en el siguiente dibujo:

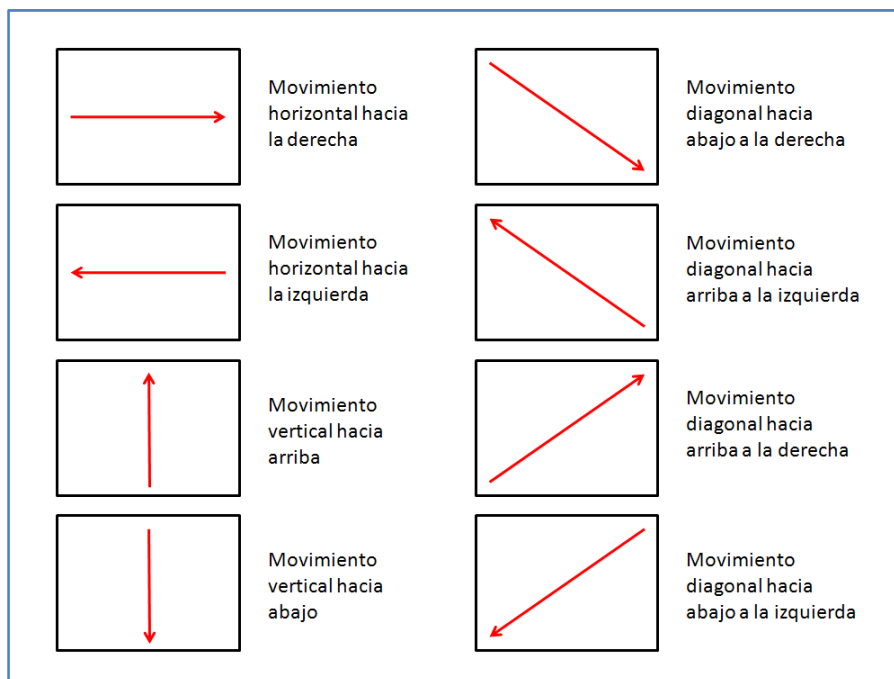


Figura 3.2.3.3. Gestos de tipo trazado reconocidos por *WiiGestures*

En cuanto al gesto de selección de opción, que no tiene representación, se basa en la desaparición y aparición de la luz emitida o reflejada por el usuario según las posibilidades comentadas anteriormente. Bastaría con encoger y estirar el dedo o bien apagar y volver a encender el puntero de luz infrarroja.

3.2.3.2. Diseño

En la fase de diseño se estudia la manera en la que se implementará cada módulo y el conjunto de todos ellos. Debido a que los requisitos son bastante amplios, se puede considerar cada uno de ellos como un apartado importante del programa.

3.2.3.2.1. REQ-3.2.3.1 Movimiento del cursor y REQ-3.2.3.2 Calibrado

La parte de movimiento del cursor y el calibrado según la distancia del usuario al *Wii mote* ya fueron tratados con profundidad en el diseño del programa *WiiClick*, por lo que basta con reutilizar el mismo código adaptándolo al nuevo programa, *WiiGestures*.

3.2.3.2.2. REQ-3.2.3.3 Reconocimiento de gestos

El sistema de reconocimientos supone la parte más compleja de esta aplicación, ya que hay que implementar un algoritmo que sea capaz de detectar cuando el usuario ha querido realizar un gesto, su trayectoria y también su sentido.

Se ha decidido utilizar el método algebraico de los mínimos cuadrados que se basa en, partiendo de una serie de coordenadas, que no son más que las capturadas por el *Wii mote* a través de su cámara infrarroja, trazar la recta que más se ajuste a esos puntos. En base a esa recta se comprobará si el usuario ha querido realizar algún gesto y cuál de ellos. La siguiente figura muestra el resultado de aplicar el método a una serie de puntos:

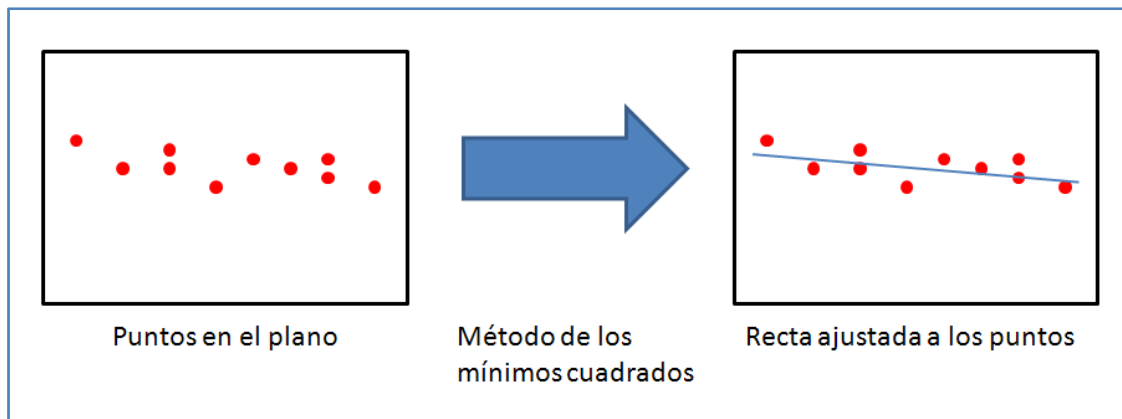


Figura 3.2.3.2.2.1. Método de los mínimos cuadrados

Para aplicar este método, se requiere hacer un muestreo constante de las posiciones de la luz infrarroja que proporciona el *Wiimote* y hacer los cálculos necesarios para hallar la recta que más se acerca a las coordenadas del conjunto de puntos muestreados.

El tipo de gesto más próximo a la recta calculada se determina en base a la inclinación de la misma, y una vez se conozca, se debe detectar el sentido en que el usuario ha realizado el movimiento, arriba o abajo si es vertical, derecha o izquierda en el caso del horizontal,... etc. Para ello se comprueba el orden creciente o decreciente de los últimos muestreos, mirando la coordenada que interese según el tipo de gesto (x para los horizontales, y para los verticales y una combinación de ambos para los diagonales) El siguiente diagrama de flujo muestra de manera gráfica cómo se estructura el módulo de reconocimiento de gestos de la aplicación:

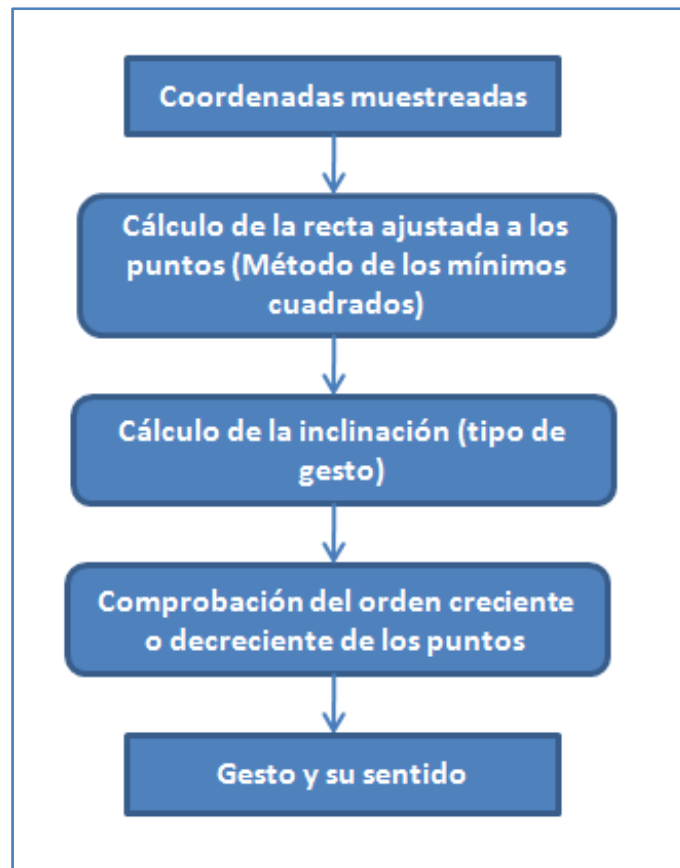


Figura 3.2.3.2.2. Diagrama de flujo del módulo de reconocimiento de gestos

Para detectar el gesto de tipo “selección de opción” basta con comprobar si la luz tiene un carácter intermitente.

3.2.3.2.3. REQ-3.2.3.4 Creación de interfaz de usuario para configuración

A través de la interfaz de usuario de la aplicación, debe ser posible configurar los siguientes parámetros:

- **Asignaciones de acción a gestos:** Para cada gesto debe haber una lista desplegable (*combobox*) con todas las posibles acciones que se le puedan asignar. Serán todas correspondientes a teclas del teclado y se implementarán las más significativas del mismo.
- **Distancia de usuario al *Wimote***

- **Número de parpadeos para seleccionar opción (*click*)**
- **Activar / Desactivar selección de opción:** Para evitar que la rápida realización de gestos pueda provocar gestos de “selección de opción” no intencionados, debe poderse desactivar la interpretación de los mismos.

Además de los parámetros descritos, se deben poder manipular las variables que permitan modificar los márgenes de detección de gestos (debido a que la mano humana no suele hacer trazados perfectos deben tenerse en cuenta ciertos márgenes de variación de ángulos, de distancia a la recta trazada,...)

Por último, existirá un botón que restaure todas las variables al valor por defecto.

El siguiente diagrama de flujo muestra el funcionamiento de la aplicación mientras se reciban coordenadas desde el *Wiimote*:

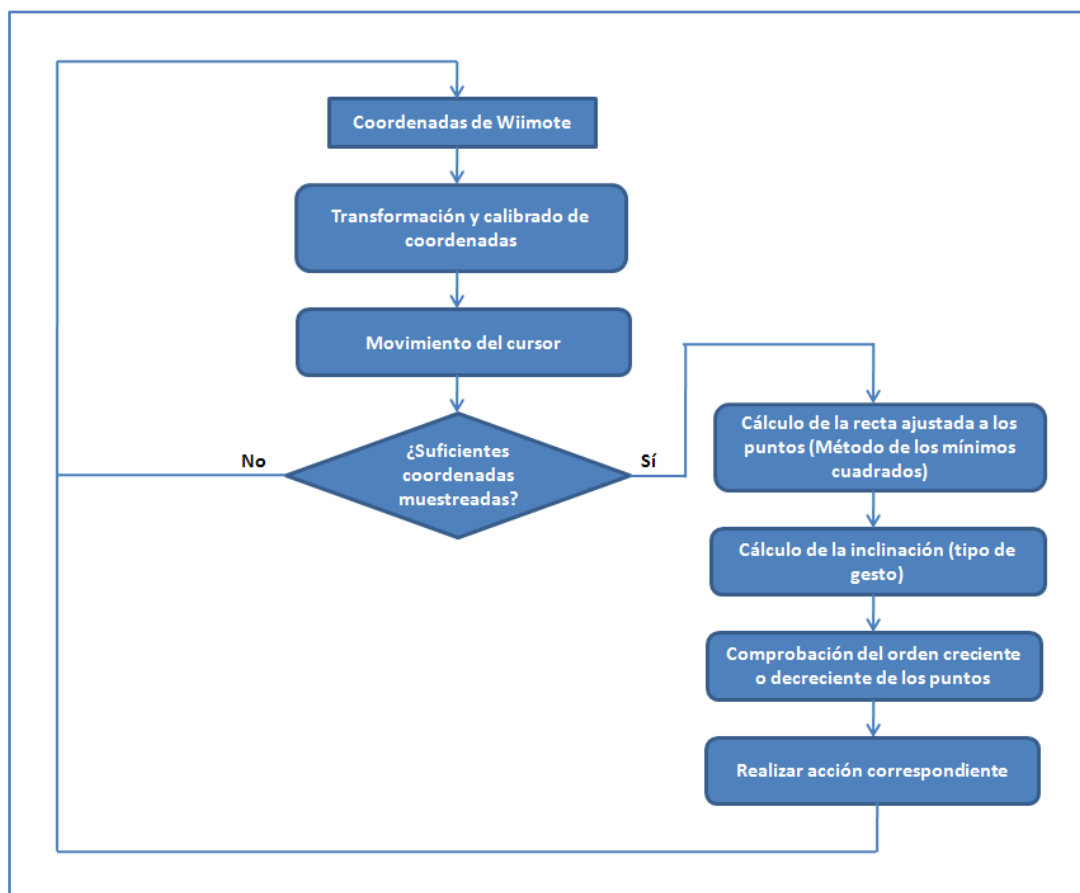


Figura 3.2.3.2.3. Diagrama de flujo de WiiGestures

3.2.3.3. Implementación

Para codificar esta aplicación se han utilizado los siguientes recursos software:

- **Microsoft Visual Studio C# 2008 Express:** Esta herramienta de desarrollo está integrada con .NET, y existe multitud de proyectos de ejemplo creados en la herramienta para las librerías utilizadas. Además es 100% gratuita.
- **Librerías *Wiiote para C# de Brian Peek:*** Estas librerías están bastante documentadas y existen múltiples ejemplos disponibles. Debido a los requisitos de constante muestreo y las diversas transformaciones matemáticas, se requiere un lenguaje potente y rápido como es C# para que la aplicación se ejecute de la forma más fluida posible.

La parte de conversión de coordenadas y calibración ya fueron detalladas en la aplicación *WiiClick*. Dado que se ha reutilizado el código de la misma, no ha habido cambios en la implementación.

Para implementar el módulo de reconocimiento de gestos se necesitan registrar cada cierto tiempo las coordenadas transformadas. Aunque se puede cambiar fácilmente mediante una variable, se ha decidido que se muestreará una vez por cada cinco lecturas de coordenada del *Wiiote* (equivale a unos 50 ms). Dichas coordenadas muestreadas se almacenan en un *array* de una determinada longitud, también configurable, que se ha fijado en 10 posiciones para que sea suficiente. Por tanto se buscarán gestos realizados en intervalos de medio segundo aproximadamente.

Una vez lleno el *array*, se traza la recta que más se ajuste a los puntos almacenados en el mismo mediante el método de los mínimos cuadrados.

El método de los mínimos cuadrados se basa en un sistema de ecuaciones formado por el conjunto de rectas que pasan por cada punto por los que se desea ajustar. Por una parte se tiene cada punto representado de la forma:

$$P_x = (a_x, b_x)$$

Para cada punto, el conjunto de las rectas que pasan por él se expresa:

$$b_x = a + ba_x$$

Donde a y b son constantes de la recta. Utilizando el conjunto de las rectas que pasan por cada punto:

$$\begin{aligned} b_1 &= a + ba_1 \\ b_2 &= a + ba_2 \\ &\dots \\ b_n &= a + ba_n \end{aligned}$$

De forma matricial, este sistema se representa:

$$\begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix} = \begin{pmatrix} 1 & a_1 \\ 1 & a_2 \\ \dots & \dots \\ 1 & a_n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

Despejando para poder calcular a y b (siendo A la matriz que representa a la de filas " $1 \ a_n$ "):

$$\begin{pmatrix} a \\ b \end{pmatrix} = (A^t * A)^{-1} * A^t \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$

Una vez calculadas a y b , se comprueba si los puntos ajustados están en su mayoría a una mínima distancia de la recta según una variable en forma de porcentaje mínimo de puntos a esa mínima distancia. Para hallar la distancia de un punto (x_1, y_1) a una recta se utiliza la fórmula:

$$\mathbf{Distancia} = \frac{|b * x_1 - y_1 + a|}{\sqrt{b^2 + 1}}$$

La inclinación de la recta compuesta por dos puntos (x_1, y_1) y (x_2, y_2) cualquiera (sacados de la recta calculada anteriormente) en grados se calcula mediante la siguiente fórmula:

$$\mathbf{Inclinación} \text{ (en grados)} = \tan^{-1} \frac{y_2 - y_1}{x_2 - x_1} \times \frac{180}{\pi}$$

Dependiendo del resultado (en valor absoluto) y por aproximación, se sabrá si tiene cabida en alguno de los tipos de gesto de trazado. Los márgenes en los ángulos deben ser ajustables a través de variables. Por defecto se utilizarán los siguientes intervalos:

- **De 0° a 10°:** Horizontal.
- **De 80° a 100°:** Vertical.
- **De 25° a 65°:** Diagonal.

Para poder saber si el usuario realmente ha querido hacer un trazado detectado o por el contrario el gesto no ha sido intencionado y su detección se debe a pequeños movimientos debidos al pulso del usuario, se debe comprobar si la distancia entre los puntos extremos de cada gesto es suficientemente larga, para ello se compara la distancia entre dichos puntos con unas distancias mínimas almacenadas en variables configurables a través de la interfaz. Por defecto estas variables serán inicializadas a la mitad del alto, ancho y diagonal de la resolución del monitor, que será detectada automáticamente.

Por último, si la prueba de distancia mínima ha resultado positiva, se debe detectar el sentido del gesto en función del crecimiento o decrecimiento de los valores muestreados.

3.2.3.4. Utilización

La forma de ejecutar *WiiGestures* es bastante sencilla e intuitiva:

- Emparejar *Wiimote* y PC mediante el gestor de conexiones *Bluetooth* de *Windows*, el suministrado por el fabricante de nuestro *dongle bluetooth* o bien mediante *BlueSoleil*.
- Especificar al programa la distancia a la que nos situaremos del *Wiimote*, para que pueda calibrar el movimiento del cursor y la acción que se desea que dispare cada gesto. En caso de que la detección no sea tan fluida como se desea, puede configurarse a más bajo nivel la aplicación a través de la configuración avanzada.

- Utilizar un array de leds más un trozo de cinta reflectante en el dedo o bien un puntero con led infrarrojo para realizar movimientos que pueda captar el *Wii mote* y que interprete el programa.

Esta forma de interactuar con el sistema se puede utilizar con cualquier aplicación de *Windows*, ya que simula el mismo movimiento y acciones que si se utilizara el ratón.

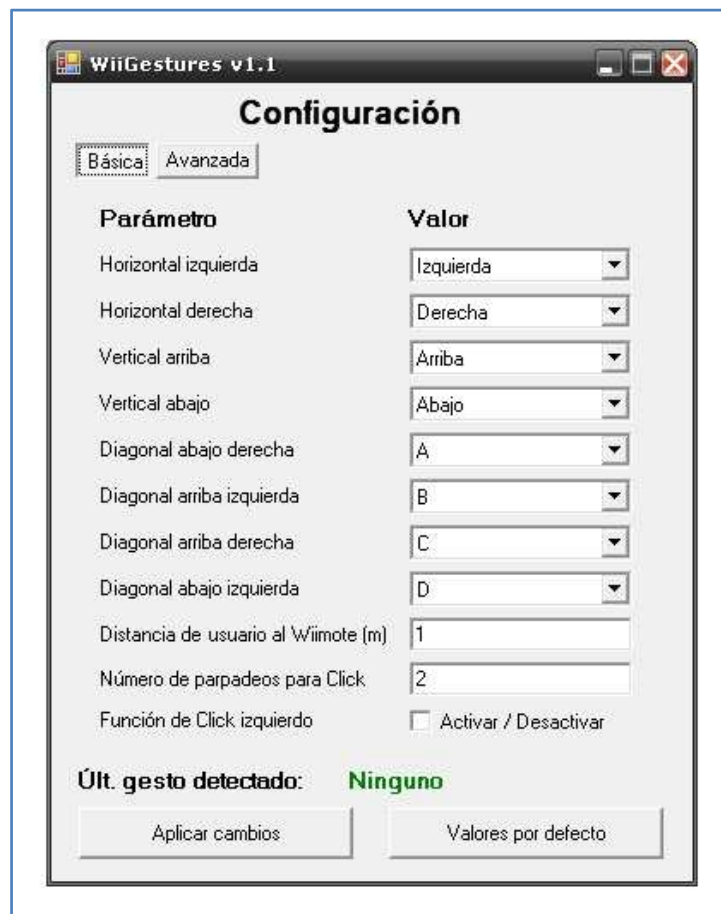
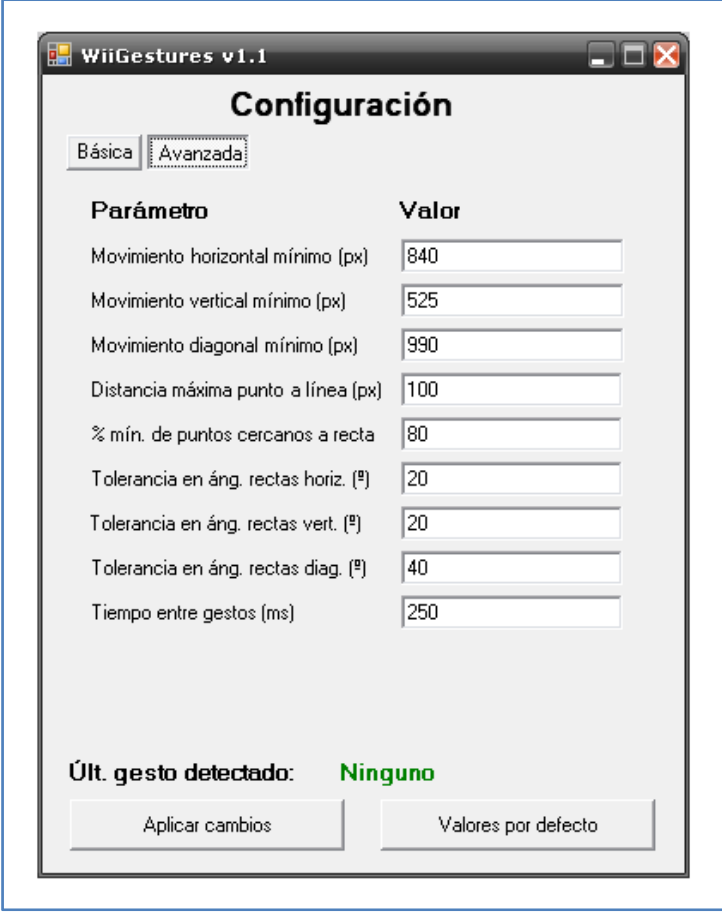


Figura 3.2.3.4.1. Configuración básica de *WiiGestures*



The screenshot shows a window titled "WiiGestures v1.1" with a "Configuración" (Configuration) header. There are two tabs: "Básica" (Basic) and "Avanzada" (Advanced), with "Avanzada" selected. Below the tabs is a table with two columns: "Parámetro" (Parameter) and "Valor" (Value). The table lists nine parameters with their corresponding values in input fields. At the bottom, there is a status indicator "Últ. gesto detectado: Ninguno" (Last gesture detected: None) and two buttons: "Aplicar cambios" (Apply changes) and "Valores por defecto" (Default values).

Parámetro	Valor
Movimiento horizontal mínimo (px)	840
Movimiento vertical mínimo (px)	525
Movimiento diagonal mínimo (px)	990
Distancia máxima punto a línea (px)	100
% mín. de puntos cercanos a recta	80
Tolerancia en áng. rectas horiz. (°)	20
Tolerancia en áng. rectas vert. (°)	20
Tolerancia en áng. rectas diag. (°)	40
Tiempo entre gestos (ms)	250

Últ. gesto detectado: **Ninguno**

Aplicar cambios Valores por defecto

Figura 3.2.3.4.2. Configuración avanzada de WiiGestures

3.2.4. Aplicación de domótica (*WiiHome*)

WiiHome, la última de las aplicaciones desarrolladas en este proyecto, está basado en *WiiClick*, que recordemos es una aplicación desarrollada para poder manejar el cursor del ratón utilizando para ello un dedo o un puntero infrarrojo. *WiiHome* añade a *WiiClick* una innovadora interfaz gráfica para el control de dispositivos domóticos X10.

3.2.4.1. Requisitos

La aplicación *WiiHome* debe cumplir con los siguientes requisitos:

- **REQ-3.2.4.1 Movimiento del cursor:** La aplicación debe ser capaz de poder mover el cursor por la pantalla recibiendo como entrada las posiciones leídas por el *WiiMote* a través de su cámara de infrarrojos.
- **REQ-3.2.4.2 Calibrado del cursor:** El usuario debe poder utilizar la aplicación a cualquier distancia respecto al *WiiMote*.
- **REQ-3.2.4.3 Reconocimiento de selección de opción:** Para que el usuario pueda interactuar con la aplicación, además de moverse por ella debe ser capaz de seleccionar opciones de la misma (normalmente a través de botones)
- **REQ-3.2.4.4 Interfaz innovadora e intuitiva adaptada al control por *WiiMote*:** Aunque se podrá utilizar mediante teclado y ratón, esta aplicación está pensada para funcionar con un *WiiMote*, por lo que será necesario diseñarla de manera que esté adaptada lo más posible al mismo. También debe representar de alguna manera el estado en el que se encuentra cada dispositivo domótico.
- **REQ-3.2.4.5 Compatibilidad con dispositivos de domótica X10:** La aplicación debe ser compatible con dispositivos X10. Se conectará a la red eléctrica del hogar utilizando un controlador/programador para PC con interfaz cable de serie.

Se requerirá un controlador X10 como el de la figura mostrada a continuación, que, conectado por interfaz serie a nuestro ordenador, será el encargado de enviar órdenes por la red eléctrica de nuestro hogar a los actuadores y de recibir los datos que envíen los sensores.



Figura 3.3.3. Controlador X10 modelo CM 11

3.2.4.2. Diseño

En esta fase se estudia la manera en la que se implementará cada módulo y el conjunto de todos ellos. Debido a que los requisitos son bastante amplios, se puede considerar cada uno de ellos como un apartado importante del programa.

3.2.4.2.1. REQ-3.2.4.1 Movimiento del cursor y REQ-3.2.4.2 Calibrado

La parte de movimiento del cursor y el calibrado según la distancia del usuario al *WiiMote* ya fueron tratados con profundidad en el diseño del programa *WiiClick*, por lo que basta con reutilizar el mismo código adaptándolo al nuevo programa, *WiiHome*.

3.2.4.2.2. REQ-3.2.4.3 Reconocimiento de selección de opción

Para cumplir con este requisito, se debe diseñar una manera en la que al usuario le sea sencillo seleccionar una opción. Tras barajar varias posibilidades se ha decidido que lo mejor será seleccionar una opción “imitando” *clicks* de ratón, es decir, haciendo

desaparecer, y aparecer la luz infrarroja. El número de parpadeos (apagados y encendidos) de la luz será configurable por el usuario a través de la interfaz. Para realizar dichos parpadeos, el usuario deberá, en el caso de estar manejando la aplicación con un puntero de luz infrarroja, apagar y encenderla tantas veces como haya especificado en la configuración, y si por el contrario utiliza luz proyectada y cinta reflectante, encoger y estirar el dedo el mismo número de veces.

3.2.4.2.3. REQ-3.2.4.4 Interfaz

Se debe crear una interfaz de usuario que sea sencilla e intuitiva a la par que adaptada al *Wii mote*. Debido a la menor precisión de la que se dispone al manejar el cursor del ratón utilizando las dos posibilidades anteriormente comentadas respecto a la que se tendría si se utilizara, la aplicación debe estar adaptada de manera que sea sencillo seleccionar las distintas opciones que ofrece. Para ello se utilizarán botones grandes sobre los que sea fácil situarse y accionar. Asimismo, la interfaz debe permitir configurar la distancia del usuario al *Wii mote* y el número de parpadeos necesarios para seleccionar una opción.

Partiendo de la anterior premisa y de que el programa debe reflejar el estado de cada dispositivo domótico X10, se ha diseñado una interfaz basada en fotografías del hogar que señala qué dispositivos pueden ser accionados y su estado. Por ejemplo una lámpara se mostrará encendida o apagada según su estado, y su contorno estará señalado con una línea fácilmente visible que indicará que dicho dispositivo puede accionarse. También debe apreciarse un cambio al posicionar el cursor sobre el área que permite encenderla o apagarla al ser seleccionada.

Las siguientes imágenes representan la manera en la que se mostrará una lámpara según su estado:



Figura 3.2.4.2.3.1. Lámpara apagada



Figura 3.2.4.2.3.2. Lámpara apagada seleccionada



Figura 3.2.4.2.3.3. Lámpara encendida seleccionada



Figura 3.2.4.2.3.4. Lámpara encendida seleccionada

Además de mostrar el estado de cada elemento, la aplicación debe ser capaz de “rotar” en una zona (una habitación, el salón, cocina...), es decir, mostrar las distintas vistas de la misma de manera que se puedan ver todos los dispositivos X10 ubicados en

ella, y también poder cambiar a otras zonas de la casa a través de un plano. El siguiente esquema muestra la apariencia que debe tener la pantalla principal de la aplicación:

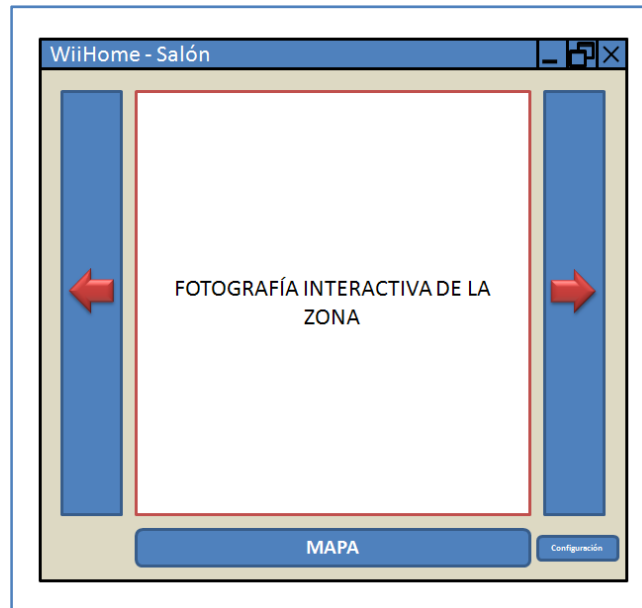


Figura 3.2.4.2.3.5. Ventana principal de WiiHome

Al pulsar el botón “MAPA” debe aparecer otra ventana que muestre el plano de la vivienda y permita seleccionar la zona que se desee visualizar:

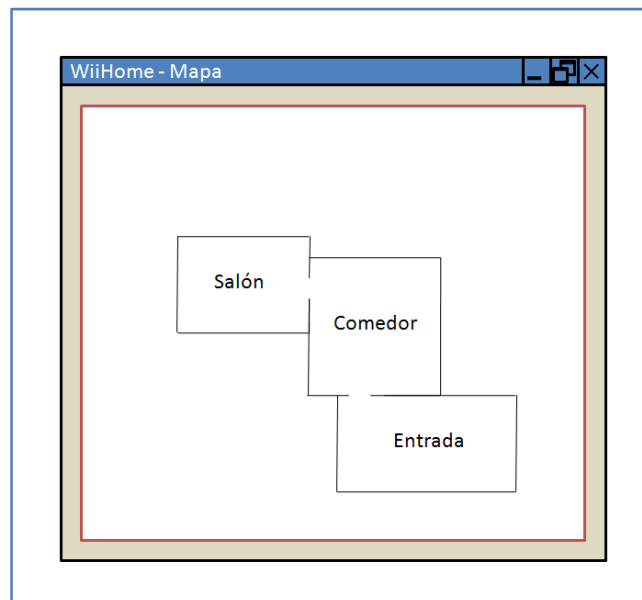


Figura 3.2.4.2.3.6. Mapa de la vivienda

Por otra parte, al pulsar en el botón de configuración debe aparecer una ventana similar a la mostrada a continuación:

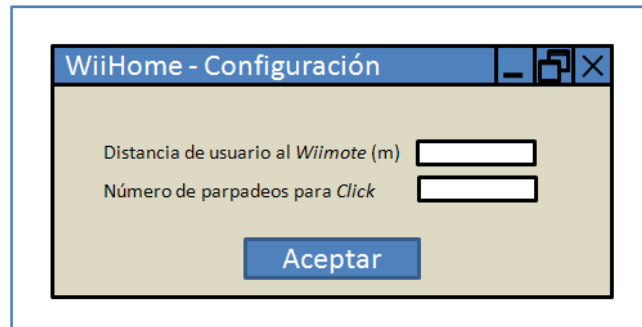


Figura 3.2.4.2.3.7. Configuración de WiiHome

3.2.4.2.4. REQ-3.2.4.5 Compatibilidad con dispositivos de domótica X10

Para hacer compatible la aplicación con dispositivos de domótica X10 basta con utilizar las librerías apropiadas para enviar órdenes y recibir mediciones de sensores a través del controlador.

3.2.4.3. Implementación

Para codificar esta aplicación se han utilizado los siguientes recursos software:

- **Microsoft Visual Studio C# 2008 Express:** Esta herramienta de desarrollo está integrada con .NET, y existe multitud de proyectos de ejemplo creados en la herramienta para las librerías utilizadas. Además es 100% gratuita.
- **Librerías Wiimote para C# de Brian Peek:** Estas librerías están bastante documentadas y existen múltiples ejemplos disponibles. Debido a los requisitos de constante muestreo y las diversas transformaciones matemáticas, se requiere un lenguaje potente y rápido como es C# para que la aplicación se ejecute de la forma más fluida posible.
- **C# X-10 Library:** Librería para el manejo de dispositivos X-10 en lenguaje C#. No se trata de una librería muy completa ni documentada puesto que está

desarrollada por un aficionado. Actualmente existen muy pocas librerías para C#, todas hechas por aficionados, poco documentadas y sin soporte.

- **Adobe Photoshop CS3:** Utilizado para el retoque fotográfico de las distintas fotografías realizadas para construir cada zona de la vivienda, así como para dibujar los planos de la misma.

La parte de conversión de coordenadas y calibración ya fueron detalladas en la primera aplicación (*WiiClick*). Dado que se ha reutilizado el código de la misma, no ha habido cambios en la implementación.

La construcción de la interfaz se ha realizado utilizando las herramientas para la creación de formularios de Visual Studio C# 2008 Express, añadiendo posteriormente las fotografías retocadas (añadido de bordes al contorno, tonalidad en rojo al posicionar el cursor sobre cada elemento....) en Photoshop CS3.

Utilizar la librería para dispositivos *X10* es sencillo pues basta con establecer una conexión con el controlador conectado al PC a través del cable serie, y a partir de ahí enviar órdenes de encendido/apagado a los dispositivos.

3.2.4.4. Utilización

La forma de ejecutar *WiiHome* es bastante sencilla e intuitiva:

- Emparejar *WiiMote* y PC mediante el gestor de conexiones *Bluetooth* de *Windows*, el suministrado por el fabricante de nuestro *dongle bluetooth* o bien mediante *BlueSoleil*.
- Especificar al programa la distancia a la que nos situaremos del *WiiMote* para que pueda calibrar el movimiento del cursor y la acción que se desea que dispare cada gesto.
- Utilizar un array de leds más un trozo de cinta reflectante en el dedo o bien un puntero con led infrarrojo para realizar movimientos que pueda captar el *WiiMote* y que interprete el programa.



Figura 3.2.4.4.1. Pantalla principal de WiiHome

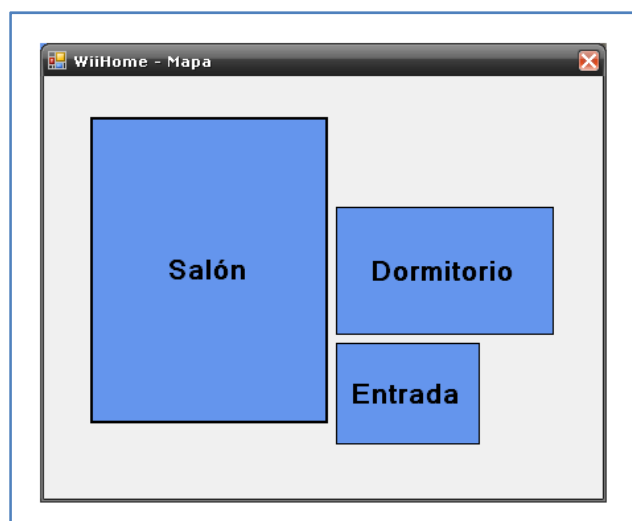


Figura 3.2.4.4.2. Mapa de WiiHome

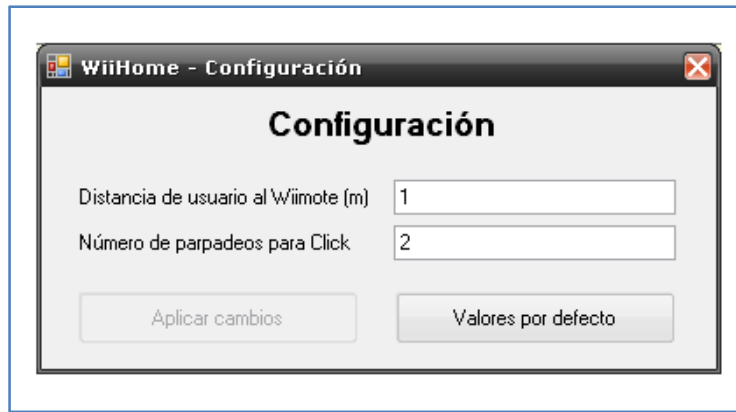


Figura 3.2.4.4.3. Configuración de WiiHome

Capítulo 4

Presupuesto

En este capítulo se analiza el coste total del proyecto desglosándolo en diversos conceptos divididos en dos grupos según su carácter:

- **Costes de adquisición de tecnología y elementos físicos:** En este apartado se detalla el coste de cada elemento físico necesario para el diseño, desarrollo, pruebas y documentación del proyecto.
- **Costes de recursos humanos:** Por otra parte, se explican los costes derivados del empleo de recursos humanos para la elaboración de todo el proyecto.

4.1. Costes de adquisición de tecnología y elementos físicos

Además de los elementos tecnológicos típicos de los que hace uso este proyecto como un PC con sistema operativo y procesador de textos, este apartado describe también el coste de todos aquellos derivados de su naturaleza, ya que se hace uso de hardware específico y de diversos elementos físicos adicionales.

La siguiente tabla detalla el coste de todos los elementos requeridos para el completo desarrollo del proyecto:

ELEMENTO	COSTE (€)
PC DELL XPS M1530	1.100
Microsoft Windows XP Professional	114
Microsoft Office 2007	129
Wiimote	45
Matriz infrarrojos	40
Punteros infrarrojos	15
Cinta reflectante (3 metros)	33
Fuente de alimentación matriz	12
Programador X10 CM 11	75
3x Módulo lámpara X10 LM 12	75
TOTAL	1.638

4.2. Costes de recursos humanos

Aparte de los costes incurridos por los elementos físicos, hardware y software, todo proyecto lleva asociado unos costes de personal que se detallan en la siguiente tabla:

RECURSO	Nº Horas	Coste/hora	COSTE (€)
Programador	150	30	4.500
Analista	100	45	4.500
Jefe de proyecto	50	60	3.000
Director del proyecto	20	85	1.700
Coordinador del proyecto	15	100	1.500
TOTAL			15.200

4.3. Coste total del proyecto

Por último se calculará el coste total del proyecto basándonos en los costes individualizados de los recursos materiales y recursos humanos:

TIPO DE COSTE	COSTE (€)
Recursos materiales	1.638
Recursos humanos	15.200
COSTE TOTAL DEL PROYECTO	16.838

Capítulo 5

Conclusiones y trabajos futuros

A lo largo de los meses de trabajo en el proyecto se han estudiado, aprendido y desarrollado técnicas matemáticas, hardware, software y formas de interactuar con un ordenador nunca vistas hasta ahora o que requerían tecnología difícil de obtener o de alto precio. En concreto hemos aprendido:

- Lenguaje de programación C#.
- Protocolo X10.
- Estado del arte y librerías de Wiimote.
- Implementación de algoritmos matemáticos complejos.
- Elaboración de una documentación completa.

Gracias a la rápida expansión que está y seguirá experimentando la consola *Wii* de *Nintendo* y por tanto su controlador *Wiimote*, el acceso a la tecnología necesaria para utilizar el software creado en este proyecto, del ya existente anteriormente y del que surgirá en los próximos años, es prácticamente universal, por lo que el coste de utilizarlo es en muchos casos nulo o casi nulo.

Asimismo, si hace dos décadas la presencia de los ordenadores en el hogar era mayoritariamente nula y hoy en día es difícil encontrar hogares que no hagan uso de los mismos, actualmente la tendencia es que ya no se disponga de un único ordenador en casa. En pocos años y en parte gracias al modelo de distribución digital de música, películas y videojuegos que se está imponiendo poco a poco, la gran mayoría de hogares contará con equipamiento informático en su salón que les permita acceder a todo el contenido almacenado en el mismo, en otros equipos del hogar, o bien en Internet así como al control automatizado de las luces, las persianas,... gracias a los dispositivos de domótica. Debido a que la utilización de un teclado y un ratón en el salón es bastante incómoda, se necesita nuevo hardware y software para el manejo de todas las posibilidades que estos equipos ofrecerán, y es ahí donde entra en juego el software desarrollado en este proyecto.

Además, debido a la poca movilidad que requiere esta forma de interacción con los programas, este software puede servir también a personas de movilidad reducida (o *PMR*) para manejar prácticamente cualquier aplicación con movimientos de dedo o de cabeza. Por otra parte, tal y como se ha podido comprobar, una interfaz sencilla e intuitiva de manejar como la desarrollada en *WiiHome* para el manejo de diversos

elementos del hogar tiene una gran acogida entre personas que nunca han manejado un ordenador, como pueden ser los mayores.

Como trabajos futuros se pueden evolucionar los desarrollos realizados por el autor ampliando sus posibilidades. Una de estas ampliaciones puede ser dotar a *WiiGestures* de un mayor número de reconocimiento de gestos, que incluyan algunos más avanzados como el trazado de círculos, cuadrados, triángulos o incluso caracteres. Por otra parte se puede crear una versión más avanzada de *WiiHome*, que permita la fácil integración de zonas y planos de la casa mediante un sencillo editor que no requiera saber manejar software de retoque fotográfico ni programación. Todas estas mejoras se podrán llevar a cabo de una manera más sencilla, gracias a los manuales elaborados.

Capítulo 6

Bibliografía

- [1] Wikipedia. *Wii*.
<http://en.wikipedia.org/wiki/Wii>
- [2] Wikipedia. *Wii Remote*.
http://en.wikipedia.org/wiki/Wii_Remote
- [3] Domótica Viva. *Curso X10*.
<http://www.domoticaviva.com/X-10/X-10.htm>
- [4] Wikipedia. *X10*.
[http://en.wikipedia.org/wiki/X10_\(industry_standard\)](http://en.wikipedia.org/wiki/X10_(industry_standard))
- [5] X10.com. *X10 Technology Transmission Theory*.
<http://www.x10.com/technology1.htm>
- [6] Wiimoteproject. *How to make tip pressure IR pen*.
<http://www.wiimoteproject.com/ir-pens/diy-how-to-make-tip-pressure-ir-pen/>
- [7] Coding4Fun. *Managed Library for Nintendo's Wiimote*.
<http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx>
- [8] SourceForge. *motej – A Wiimote Library for Java*.
<http://motej.sourceforge.net/index.html>
- [9] WiiLi. *Wiiremote*.
<http://www.wiili.org/Wiiremote>
- [10] WiiYourself!.
<http://wiiyourself.gl.tter.org/>
- [11] Digital retrograde. *Wiim*.
<http://digitalretrograde.com/projects/wiim/>
- [12] wiiuse.
<http://www.wiiuse.net/>
- [13] Abstrakraft. *CWiid*.
<http://abstrakraft.org/cwiid/wiki/libcwiid>
- [14] Wiimote2Joystick.
<http://spam.workaround.ch/wm2js/>
- [15] NI Forum. *Using a wiimote with LabVIEW*.
<http://forums.ni.com/ni/board/message?board.id=170&message.id=249428>
- [16] Google Code. *Wiiuse*.
<http://code.google.com/p/wiiuse/>
- [17] Carl Kenner. *GlovePIE*.
<http://carl.kenner.googlepages.com/glovepie>

[18] Johnny Chung Lee. *Wii Projects*.

<http://www.cs.cmu.edu/~johnny/projects/wii/>

[19] WiiLi. *Bluesoleil solution for not supported adapters*.

<http://www.wiili.org/forum/bluesoleil-solution-for-not-supported-adapters-t794.html>

Anexo I: Manual para conexión de Wiimote al PC

En este anexo se explicarán las dos posibilidades para conectar el *Wiimote* a nuestro PC con *Windows XP*:

- **Utilizando el gestor de *Windows XP*:** Utilizar este gestor no requiere la instalación de ningún software adicional, además es compatible con un mayor número de adaptadores para conexiones *bluetooth*, ya sea el interno típico de los portátiles o bien un *dongle bluetooth* conectado normalmente por *USB*. La desventaja que tiene utilizar este gestor es que no es compatible con todas las aplicaciones creadas o analizadas en este proyecto.

- **Utilizando *BlueSoleil*:** *BlueSoleil* es una aplicación dedicada a la gestión y conexión de dispositivos *bluetooth*. Como inconvenientes tiene que es de pago y no es compatible 100% con todos los adaptadores *bluetooth* bien sean internos o de tipo *dongle*. No obstante, una vez correctamente configurado hace funcionar correctamente todas las aplicaciones de este proyecto.

1. Conexión de *Wiimote* utilizando el gestor de *Windows XP*

El proceso de conexión del *Wiimote* con el gestor de conexiones *bluetooth* de *Windows XP* es muy sencillo:

1. Pulsar con el botón derecho del ratón sobre el icono del logo *bluetooth* de la barra de tareas y pinchar en “*Agregar dispositivo bluetooth*”.
2. En la ventana que aparecerá, seleccionar la casilla “*Mi dispositivo está configurado y listo para ser usado*”.
3. El programa comenzará a buscar dispositivos, mientras lo hace, mantener pulsados los botones *1* y *2* del *Wiimote*.
4. Seleccionar el *Wiimote* que aparece listado como “*Nintendo RVL-CNT-01*” y pulsamos en “*Siguiente*” mientras se mantienen pulsados los botones *1* y *2* del *Wiimote*.
5. En la siguiente ventana, seleccionar la opción “*No usar ninguna clave de paso*” mientras seguimos manteniendo pulsados los botones *1* y *2*.

Tras realizar todos los pasos anteriores, el *Wiimote* estará conectado y listo para utilizar las aplicaciones compatibles con este método.

El esquema mostrado a continuación muestra todo el proceso de forma gráfica:

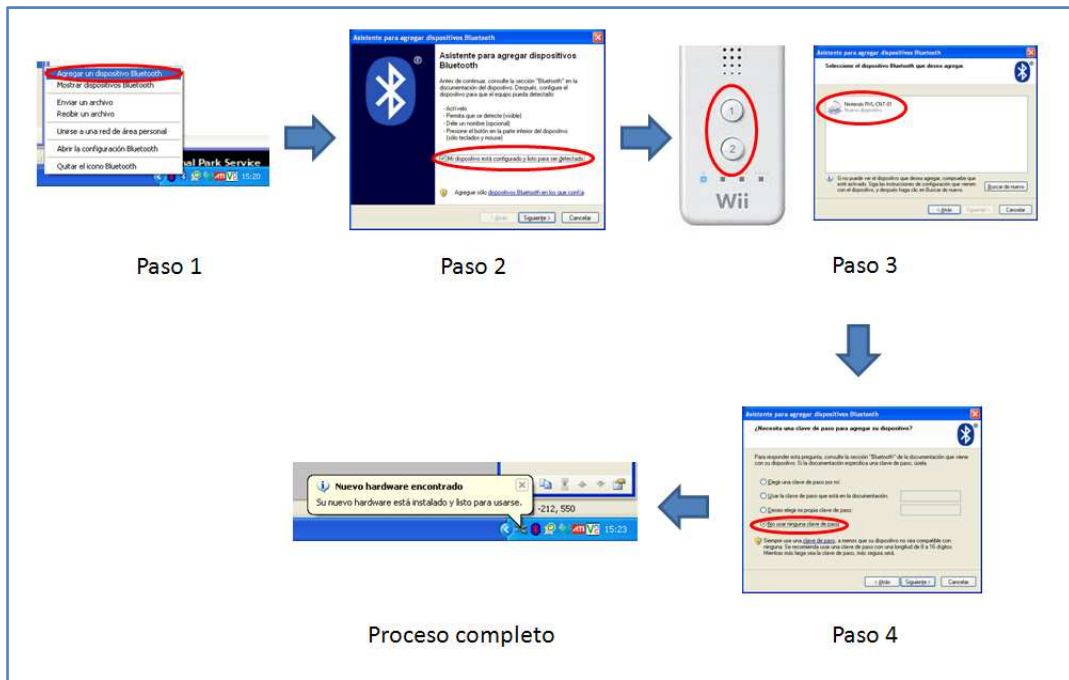


Figura A1.1.1. Proceso de emparejamiento del Wiimote con el gestor bluetooth de Windows XP

2. Conexión de *Wiimote* utilizando *BlueSoleil*

Para utilizar *BlueSoleil* se deben seguir los siguientes pasos:

1. Instalar y adaptar la aplicación.
2. Emparejar el *Wiimote* con *BlueSoleil*.

2.1. Instalar la aplicación

Instalar *BlueSoleil* no es una tarea más compleja que instalar cualquier otra aplicación destinada a usuarios sin muchos conocimientos. El problema está en que no todos los adaptadores *bluetooth* son compatibles con él, por lo que se explicará una forma de conseguir su compatibilidad en la mayoría de los casos en los que surja dicho problema^[19].

La siguiente figura muestra el comportamiento de *BlueSoleil* cuando éste no es capaz de encontrar nuestro adaptador *bluetooth*:



Figura A1.2.1.1. Problema de incompatibilidad de *BlueSoleil*

Para solucionarlo se deben realizar los siguientes pasos:

1. **Averiguar los valores *VID*** (Vendor ID o identificador del fabricante) y ***PID*** (Product ID o identificador de producto) **de nuestro adaptador *bluetooth***: Para ello se debe ir al administrador de dispositivos y buscar dentro de “dispositivos bluetooth” nuestro adaptador, pulsamos botón derecho del ratón sobre él y seleccionamos “propiedades”, luego la pestaña “detalles” y por último en el combobox seleccionamos “identificadores de hardware”. Debería aparecer algo similar a esto:

```
USB\Vid_050&Pid_0121&Rev_0100
USB\Vid_050&Pid_0121
```

En este caso nuestro *VID* sería “050” y el *PID* “0121”.

2. **Editar el fichero *btcusb.inf***: localizado habitualmente en la carpeta “C:\Archivos de programa\IVT Corporation\BlueSoleil\driver\usb” salvo que se escogiera otra ruta durante la instalación.

Ahora se debe buscar la sección identificada como “[ControlFlags]” y añadir una línea con la forma:

```
ExcludeFromSelect = USB\VID_050D&PID_0121
```

Sustituyendo el *VID* y *PID* con los que correspondan.

En el mismo fichero, buscar la sección “[IVT]” y encontrar alguna línea con el fabricante de nuestro adaptador. Por ejemplo si fuera “*Belkin*” habría que encontrar una línea así:

```
%Belkin.DeviceDesc%=BTusb_DDI, USB\VID_050D&PID_0081
```

Se copia y se pega debajo y se cambian los valores del *VID* y *PID* según corresponda.

3. **Editar el fichero *bttl.ini***: localizado habitualmente en “C:\Archivos de programa\IVT Corporation\BlueSoleil” salvo que se cambie la ruta por defecto de la instalación.

Ahora hay que buscar un bloque de código similar al mostrado a continuación cuyo fabricante indicado en la línea “*Manufacture*” sea el mismo que el de nuestro adaptador, copiarlo y pegarlo tras el último bloque de este tipo realizando las modificaciones oportunas de *VID* y *PID*:

```
[HW77]  
ID=USB\VID_050d&PID_0084  
Type=Bluetooth USB Dongle  
DLL=Driver\USB\btcusb.dll  
DLLD=Driver\USB\btcusbd.dll  
Infile=Driver\USB\btcusb.inf  
Manufacture=BelKin
```

Por último solo queda modificar la primera línea del bloque “[HWxxx]” cambiando el número por el siguiente al del último bloque. Es decir, si el último bloque está numerado como “[HW110]”, el nuestro copiado justo debajo será “[HW111]”. También hay que buscar la línea del fichero indicada como

“NUM=xxx” y aumentar en uno el número, haciéndolo coincidir con el que acabamos de modificar.

4. **Reinstalar el *driver* de nuestro dispositivo *bluetooth*:** Realizar la instalación con normalidad, pero en el punto donde pregunte dónde está el *driver*, especificar manualmente la ruta eligiendo el fichero *btusb.inf* anteriormente editado.

2.2. Emparejar el *Wiimote* con *BlueSoleil*

Una vez instalado correctamente *BlueSoleil*, se debe emparejar el *Wiimote* con el mismo. Para ello pinchamos con el botón derecho del ratón en el icono azul que representa el logotipo de *bluetooth* de la barra de tareas y elegimos “*Display classic view*”. Aparecerá la siguiente ventana:



Figura A1.2.2.1. Ventana de *BlueSoleil*

Para emparejar el *Wiimote* con el programa, se deben seguir los siguientes pasos:

1. **Mantener pulsados simultáneamente los botones 1 y 2 del *Wiimote*.**
2. **Hacer doble *click* sobre la bola amarilla de *BlueSoleil*:** Tras ello aparecerá el *Wiimote* representado como un *joystick*.
3. **Hacer doble *click* sobre el *Wiimote*:** aparecerán iluminados arriba los servicios que proporciona el dispositivo. En nuestro caso se iluminará el ratón, que indica que el *Wiimote* puede usarse como dispositivo de entrada.
4. **Hacer doble *click* sobre el icono del ratón:** A partir de ese momento, el *Wiimote* quedará emparejado con nuestro PC, listo para ser usado por nuestras aplicaciones.

El siguiente esquema muestra el proceso completo:

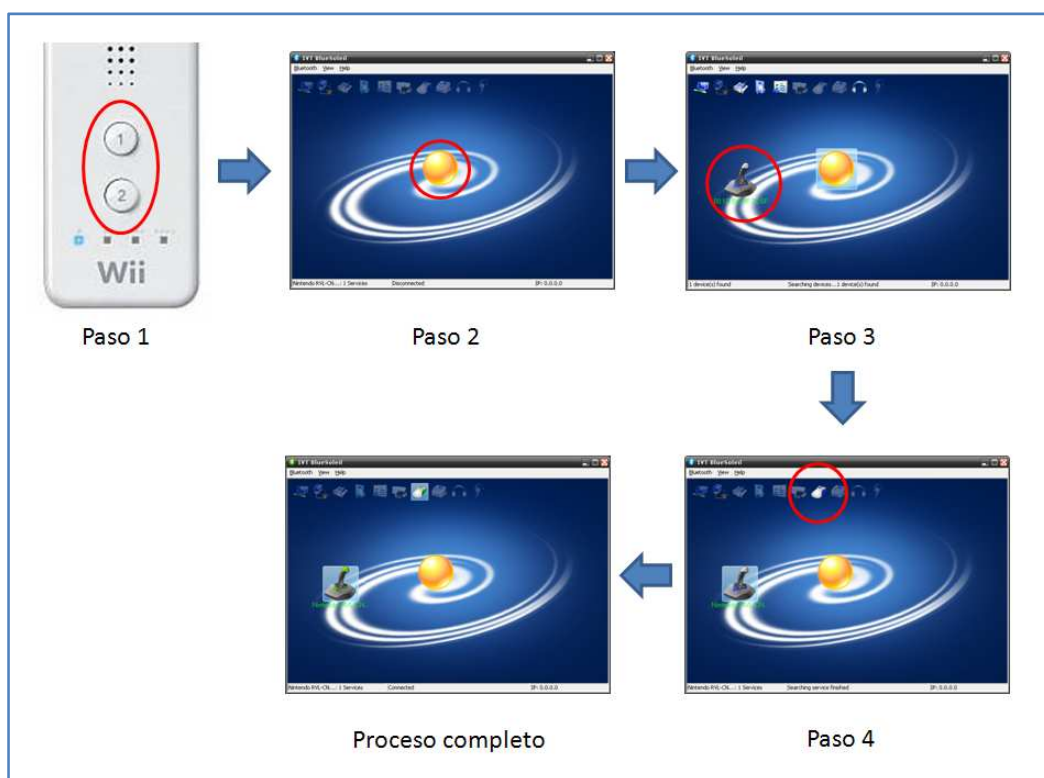


Figura A1.2.2.2. Proceso de emparejamiento de *Wiimote* con *BlueSoleil*

Anexo II: Manual de programación de Wiimote en C#

A través de este anexo se pretende crear un punto de partida para los programadores que deseen introducirse en el mundo de las aplicaciones que utilizan el *Wiimote* como dispositivo de entrada.

Aunque existen diversas librerías con las que poder acceder a la información que nos ofrece el dispositivo, se utilizará la misma que está integrada en los distintos desarrollos creados en este proyecto, *WiimoteLib* (*.NET Managed Library for the Nintendo Wii Remote*) creada por *Brian Peek*, y que puede ser descargada gratuitamente a través del siguiente enlace: <http://www.codeplex.com/WiimoteLib> . Por otra parte se utilizará el software de desarrollo *Microsoft Visual C# 2008 Express*, disponible también gratuitamente en la url: <http://www.microsoft.com/express/default.aspx>

Para poder asociar el *Wiimote* a nuestro PC, se recomienda seguir el apartado 2 (“Conexión del *Wiimote* utilizando *BlueSoleil*) del Anexo I. Una vez esté asociado ya podemos probar los desarrollos creados.

Para poder utilizar el *Wiimote* en nuestras propias aplicaciones C#, debemos seguir los siguientes pasos:

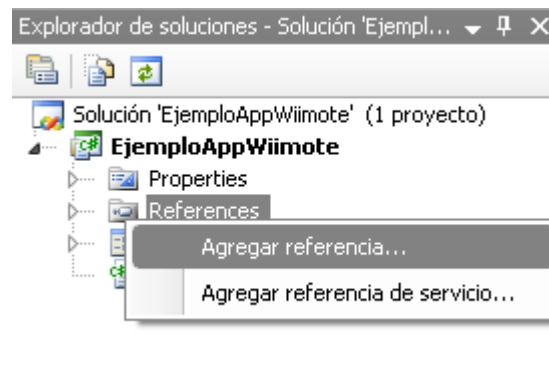
1. Descargar e importar la librería *WiimoteLib* a nuestro proyecto.
2. Establecer la conexión entre el *Wiimote* y nuestra aplicación.
3. Elegir el tipo de informe que deseamos obtener del *Wiimote*.
4. Leer los informes o enviar órdenes al *Wiimote*.

1. Descargar e importar la librería *WiimoteLib* a nuestro proyecto

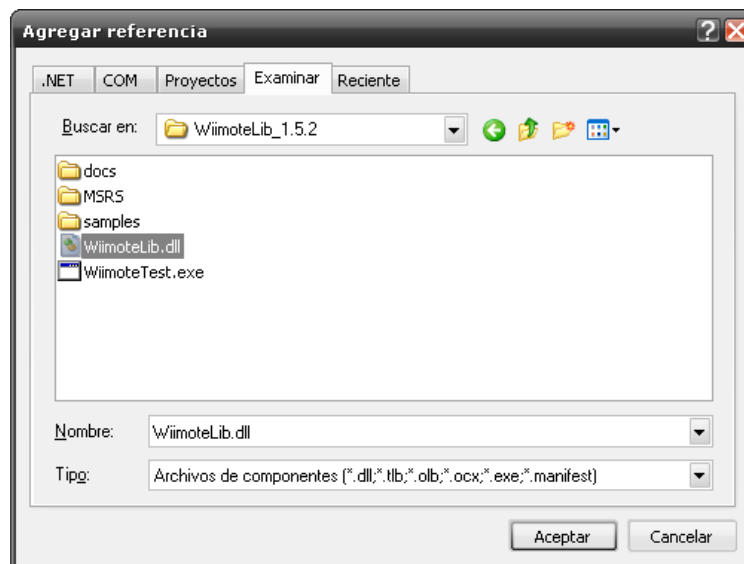
Las librerías se pueden descargar de <http://www.codeplex.com/WiimoteLib>. En el momento de la creación de este manual, la versión de las mismas es la 1.5.2 por lo que en caso de utilizarse una más nueva podrían requerirse ciertos cambios en el código para que funcione.

Aunque importar la librería a *Visual C# 2008 Express* es muy sencillo, se indica de forma gráfica a continuación.

En el explorador de soluciones, pinchamos en “Referencias” y seleccionamos “Agregar referencia”



En la ventana que aparecerá, seleccionar la pestaña “Examinar” y seleccionar el archivo *WiimoteLib.dll* almacenado en la carpeta donde hayamos descomprimido la librería.



Una vez pulsemos el botón “Aceptar” la librería quedará importada y referenciada. Tan sólo falta incluirla en la correspondiente sección de los ficheros de código fuente donde se quiera utilizar, esto se hace escribiendo la siguiente línea:

```
using WiimoteLib;
```

2. Establecer la conexión entre el *Wiimote* y nuestra aplicación

El siguiente paso consiste en buscar el *Wiimote* conectado al PC y establecer la conexión con el mismo. Debido a que la librería implementa toda la lógica de buscar el controlador, basta con utilizar las siguientes líneas de código para comenzar a comunicarnos con el *Wiimote*.

```
Wiimote wiimote;  
wiimote = new Wiimote();  
wiimote.Connect();
```

3. Elegir el tipo de informe que deseamos obtener del *Wiimote*

Tal y como se explicó en el apartado 2.2.1.3. de la presente documentación, la manera de conocer el estado de todas las variables del *Wiimote* está basada en la creación de informes por el mismo para la posterior lectura de dichos informes en nuestro programa. Tras realizar la conexión del *Wiimote* con nuestra aplicación, debemos seleccionar un tipo de informe entre los siguientes, en función de lo que deseemos conocer:

- **Status:** Estado general del *Wiimote*.
- **ReadData:** Para lectura de datos de la memoria interna.
- **Buttons:** Información sobre botones pulsados.
- **ButtonsAccel:** Información sobre botones y acelerómetros.
- **IRAccel:** Información sobre botones, acelerómetros y puntos infrarrojos.
- **ButtonsExtension:** Información sobre botones y extensiones del mando (periféricos adicionales conectados al *Wiimote*)
- **ExtensionAccel:** Información sobre botones, acelerómetros y extensiones.
- **IRExtensionAccel:** Información sobre botones, acelerómetros, infrarrojos y extensiones.

Suponiendo que se desea utilizar el tipo *IRExtensionAccel*, que es el más completo:

```
wiimote.SetReportType(InputReport.IRExtensionAccel, true);
```

4. Leer los informes o enviar órdenes al *Wiimote*

Una vez realizados todos los pasos anteriores, ya se puede trabajar con el *Wiimote* leyendo su estado a través de los informes o enviándole órdenes.

Aunque en la documentación incluida junto a la librería se detallan todos los eventos, métodos y variables que nos ofrece la misma, se verán algunos ejemplos de cómo trabajar con el *Wiimote* para facilitar la tarea.

4.1. Lectura de informes

La lectura de informes puede realizarse de dos formas, por eventos o por *polling*. A través de estas formas se puede averiguar la posición del *Wiimote* gracias al estado del acelerómetro o de la cámara de infrarrojos, que nos dará la situación de hasta cuatro puntos infrarrojos distintos.

Para realizar gestión de eventos, la API de la librería nos ofrece dos eventos distintos:

1. *WiimoteChanged*: Este evento se producirá cuando haya informes del propio *Wiimote* disponibles.
2. *WiimoteExtensionChanged*: Este segundo evento funciona igual que el anterior pero está dedicado a las extensiones del *Wiimote*, como el *Nunchuck* o la guitarra.

En caso de optar por realizar gestión por eventos, se debe especificar qué método va a encargarse de procesar los informes cuando se produzca un evento. Para ello utilizamos la siguiente línea:

```
wiimote.WiimoteChanged += nuevoInformeWiimote;
```

Y crear el método que gestionará los informes:

```
private void nuevoInformeWiimote(object sender, WiimoteChangedEventArgs args)
{
    //Aquí el código para leer la información que nos interese
}
```

Una vez hayamos implementado la gestión de eventos, podemos utilizar los distintos métodos para leer su posición detectada por el acelerómetro, la de los puntos infrarrojos detectados por la cámara y si se ha producido la pulsación de alguno de sus

botones. Para poder leer cualquier tipo de información se necesita previamente obtener el estado del *Wiimote*, que implementaremos utilizando la siguiente línea de código:

```
WiimoteState estado = args.WiimoteState;
```

4.1.1. Lectura de posición según acelerómetro

Se puede acceder a las coordenadas detectadas por el acelerómetro utilizando los siguientes atributos de la clase *WiimoteState*:

```
estado.AccelState.Values.X;
estado.AccelState.Values.Y;
estado.AccelState.Values.Z;
```

4.1.2. Lectura de posición de puntos infrarrojos

Debido a que el *Wiimote* puede obtener la posición en el plano de hasta cuatro puntos infrarrojos, la clase *WiimoteState* dispone de un array con cuatro posiciones para cada uno de ellos:

```
estado.IRState.IRSensors[0];
estado.IRState.IRSensors[1];
estado.IRState.IRSensors[2];
estado.IRState.IRSensors[3];
```

La primera posición en el array la ocupará el primer punto detectado, la segunda el segundo,... etc. Para obtener las coordenadas de, por ejemplo, el primer punto encontrado, utilizamos las siguientes líneas de código:

```
estado.IRState.IRSensors[0].RawPosition.X; //Posición del primer punto en eje X
estado.IRState.IRSensors[0].RawPosition.Y; //Posición del primer punto en eje Y
```

4.1.3. Lectura de pulsación de botones, estado de leds y batería

También se puede averiguar si alguno de los botones de los que dispone el *Wiimote* ha sido presionado (estos comandos devolverán un valor booleano):

```
estado.ButtonState.A; //Botón A
estado.ButtonState.B; //Botón B
estado.ButtonState.Up; //Cruceta arriba
estado.ButtonState.Down; //Cruceta abajo
estado.ButtonState.Left; //Cruceta izquierda
estado.ButtonState.Right; //Cruceta derecha
estado.ButtonState.One; //Botón 1
estado.ButtonState.Two; //Botón 2
estado.ButtonState.Minus; //Botón -
```

```
estado.ButtonState.Plus; //Botón +  
estado.ButtonState.Home; //Botón HOME
```

Para conocer el estado de cada led (devolverá un valor booleano):

```
estado.LEDState.LED1; //Estado Led 1  
estado.LEDState.LED2; //Estado Led 2  
estado.LEDState.LED3; //Estado Led 3  
estado.LEDState.LED4; //Estado Led 4
```

Y por último para conocer el nivel de carga de la batería (devolverá un valor de tipo *byte*):

```
Estado.Battery;
```

4.2. Envío de órdenes

La librería permite el envío de algunas órdenes básicas al *Wiimote* como el encendido y apagado tanto de cualquier led como de la función de vibración.

Para utilizar el comando que permite encender o apagar leds, se debe especificar uno por uno el estado en que se desea que quede cada led. Por ejemplo si queremos encender los dos primeros y apagar el tercero y el cuarto, el comando necesario será:

```
wiimote.SetLEDS(true, true, false, false);
```

Por otra, parte si queremos activar o desactivar la función de vibración, podemos utilizar el siguiente método:

```
wiimote.SetRumble(true); //Activar la función de vibración  
wiimote.SetRumble(false); //Desactivar la función de vibración
```