Prototipado rápido de aplicaciones Opensource para rehabilitación usando dispositivos inalámbricos

Autor: Carlos Pastor Herrán

Tutor: Juan González Gómez

Cotutor: Alberto Jardón Huete

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

PROYECTO FIN DE CARRERA

UNIVERSIDAD CARLOS III DE MADRID

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN:

SONIDO E IMAGEN

Leganés, Octubre de 2010

Agradecimientos

"Mientras el río corra, los montes hagan sombra y en el cielo haya estrellas, debe durar la memoria del beneficio recibido en la mente del hombre agradecido." (Virgilio)

El proyecto fin de carrera supone el final de un ciclo empezado hace cinco años. Durante todo este largo tiempo he tenido gente a mi lado que me ha ayudado tanto académicamente como personalmente haciendo que llegue hasta donde me encuentro ahora mismo. Con estas líneas intento demostrarles el agradecimiento hacia ellos y hacia lo que han hecho por mí.

A Juan González, mi tutor del proyecto. Ha demostrado una gran calidad docente en sus explicaciones y correcciones de los capítulos que integran el proyecto y otros aspectos formativos. Además, destacar la calidad humana demostrada en su apoyo y paciencia durante el desarrollo del proyecto.

A mis compañeros del colegio, a mis recientes compañeros de piso y a todas las personas que me ayudaron y no mencione por olvido o cuestiones de espacio (ruego me perdonen) aceptad mi más sincero agradecimiento.

A mis compañeros de clase. Todos los compañeros de clase nos hemos ayudado en el ámbito universitario pero hago especial mi agradecimiento a los que con el paso del tiempo se han convertido en amigos en Madrid. A Lidia, Tania, Marta y Guille, gracias.

A mis compañeros en la residencia. Muy especialmente a Santi y Pedrillo, con quien me une una muy buena amistad. Gracias por adoptarme en vuestra habitación continuamente y hacer que fuera de tres personas en vez de dos.

Hago extensible el agradecimiento a todos aquellos que me han apoyado e interesado por la carrera y el proyecto especialmente en el último año.

A mis otros amigos de la residencia. A Joaquín y Esperanza. Vosotros habéis sido algo parecido a mis padres en los cuatro años que estuve en la residencia. Gracias especialmente a Joaquín por nuestras largas charlas nocturnas y por sus valiosos consejos para la vida. También a Pepi, nuestra madre en la residencia.

A María José. Pese a que la vida nos haya separado, es de justicia dar las gracias a las personas que tanto han hecho por otra. Gracias por tu cariño y apoyo en todos los ámbitos durante la carrera. Por eso y por todo lo que me ayudaste, muchas gracias.

A mis amigos de León. Para ellos va un muy sentido agradecimiento ya que son los que siempre han estado a mi lado cuando lo he necesitado. Pese a la distancia, vuestra ayuda y apoyo me ha llegado como si estuviéramos juntos. Por esos momentos de vacaciones o puentes que pasaba en mi tierra con ellos, siempre cortos, pero muy intensos. A Pablo, Pela y María, muchas gracias. Extensible también este agradecimiento a Pere, con quien inicie la aventura madrileña tras su estancia en León.

A mi familia. A todos los miembros de mi familia que, de cerca, han seguido mis avances y me han aconsejado. A mis tíos, primos y abuela gracias por vuestra compañía y vuestro amor hacia mí.

Por último y no menos importante, a mis padres. A ellos les debo el poder estar aquí en este momento. Si no fuera por ellos, no sería ingeniero técnico de telecomunicación. Debo agradecer todo lo que han hecho por su hijo para que pudiera llegar hasta donde se encuentra. A mi madre, por su preocupación y cariño hacia su hijo, sin los cuales no habría podido avanzar. A mi padre, por sus sabias lecciones para la vida y su sereno apoyo, imprescindible para mí en los momentos más duros. A vosotros, mis mejores amigos y mi mejor modelo vital, MUCHÍSIMAS GRACIAS.

Dedico este trabajo a mi abuela Carmen y especialmente a mi abuelo José. Gracias por estar siempre pendientes del nieto que se fue a Madrid para que sus primos le ayudasen en todo lo que fuera necesario. Pese a que no os pueda enseñar el proyecto y deciros que ya soy ingeniero, espero que os sintáis orgullosos de vuestro nieto.

Índice general

Índice de figuras

Capítulo 1
Introducción

1.1. MOTIVACIÓN

La motivación principal de este proyecto es mejorar las técnicas de rehabilitación y fisioterapia usando las nuevas tecnologías. Las sesiones de rehabilitación son generalmente repetitivas y aburridas para el paciente, por lo que resultan poco motivadoras. La inclusión de tecnología en forma de juego en un ordenador o en otro soporte de similar función se convierte en una interesante opción para motivar al paciente. De esta manera puede seguir ejercitándose y mejorar el resultado ya que tendrá una mejor disposición al trabajo.

Esta idea es válida para un proceso rehabilitador o unas sesiones de fisioterapia pero se puede hacer extensible a otros ámbitos. El uso de un juego puede resultar más atractivo para pacientes que se encuentren en el área de terapia ocupacional o para pacientes que sufran enfermedades neurodegenerativas.

Este proyecto además se engloba dentro del proyecto ASIBOT del RoboticsLab de la Universidad Carlos III de Madrid [1]. ASIBOT es un robot portátil con funciones asistenciales para personas de edad avanzada y enfermos. Se ha probado en entornos reales con usuarios, pudiendo ayudarles en tareas tales como comer, beber, afeitarse, etc. El trabajo con dispositivos inalámbricos nos puede ofrecer una interfaz que podamos adaptar al robot. Así, con dispositivos de bajo coste, podremos controlar total o parcialmente el robot

en su desplazamiento o en los movimientos que ayudan al paciente a realizar las acciones.

1.2. OBJETIVOS

Los objetivos del proyecto son:

- Estudio de los diferentes dispositivos inalámbricos que pueden ser usados en aplicaciones asistenciales para rehabilitación y fisioterapia.
- Establecimiento y conocimiento del hardware básico a usar en el desarrollo de aplicaciones.
- Desarrollo de plataforma software de base para la creación rápida de aplicaciones a medida.
- Desarrollo de aplicaciones Opensource para rehabilitación y fisioterapia.

La consecución de estos objetivos nos llevará a la consecución de un prototipado rápido de aplicaciones Opensource para rehabilitación usando dispositivos inalámbricos.

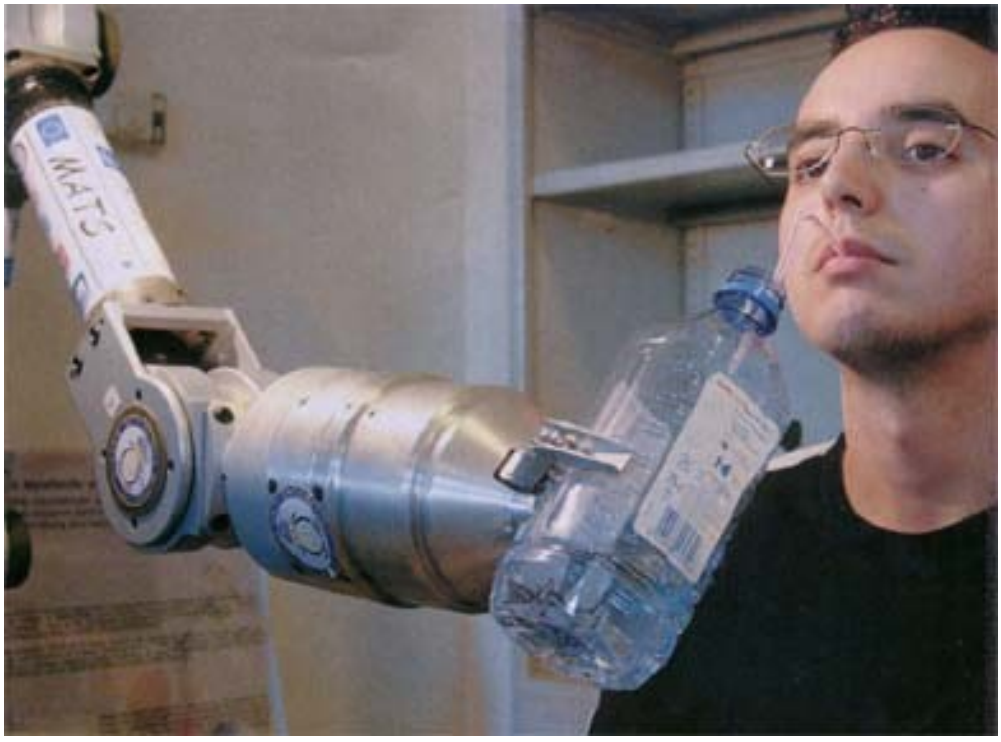

Figura 1.1. Robot ASIBOT asistiendo en la acción de beber

1.3. ORGANIZACIÓN

Este proyecto se divide en siete capítulos, detallados en el Índice. A continuación, se realiza una breve descripción de los mismos para una mejor comprensión de la organización del documento:

En el capítulo 2 se realiza un estudio del estado del arte de los dispositivos inalámbricos existentes en el mercado. Además se presentan las alternativas software para la creación de aplicaciones y se justifican las elecciones tomadas para el desarrollo de la plataforma software deseada.

Los capítulos desde el 3 al 6 tienen una estructura similar. Atienden cada uno a un dispositivo hardware diferente o a una función diferenciada de éste. En primer lugar, se detallan las características técnicas y el modo de funcionamiento. En segundo lugar, se expone el API utilizada para la programación de aplicaciones con estos dispositivos, incluyendo ejemplos para cada uno de ellos. Y en tercer lugar, se exponen las aplicaciones desarrolladas que persiguen cubrir los objetivos del proyecto. Los dispositivos por orden de capítulo son:

- Capítulo 3: Wii Remote
- Capítulo 4: Wii Remote+IR LEDs
- Capítulo 5: Wii Remote+Wii Motion Plus
- Capítulo 6: Wii Balance Board

En el capítulo 7 se exponen las conclusiones y se presentan los trabajos futuros que darían continuidad a este proyecto.

Por último se incluyen los Anexos y la Bibliografía.

Capítulo 2
Estado del arte

2.1. INTRODUCCIÓN

En este capítulo se introducen los conceptos de rehabilitación y los sistemas comerciales existentes en este campo. Además se exponen los sistemas alternativos actuales capaces de ayudar en sistemas de rehabilitación y el uso que se le da a la consola Wii. Por último, se presentan las herramientas software que ayudan al desarrollo de aplicaciones y se justifica la elección que se toma en este proyecto.

2.2. REHABILITACIÓN

La rehabilitación es el diagnóstico, evaluación, prevención y tratamiento de la incapacidad encaminados a facilitar, mantener o devolver el mayor grado de capacidad funcional e independencia posibles [2]. Incluye todas las medidas destinadas a reducir el impacto de las condiciones de incapacidad y minusvalía y hacer posible que las personas incapacitadas y minusválidas alcancen la integración social. Los objetivos de la rehabilitación son los de entrenar a personas incapacitadas y minusválidas a adaptarse a su entorno y el de intervenir en éste para facilitar la integración social.

La especialidad médica que coordina el proceso rehabilitador es la **medicina física y rehabilitación** [3]. Esta especialidad comprende el estudio, detección y diagnóstico, prevención y tratamiento clínico o quirúrgico de los

enfermos con procesos discapacitantes. Dentro de este proceso rehabilitador cabe destacar dos zonas de acción:

- **Alcance del completo potencial físico**. La fisioterapia es el arte y ciencia del tratamiento por medio de ejercicio terapéutico, calor, frío, luz, agua, masaje y electricidad. Se ocupa de la recuperación física y de la prevención cuando el paciente ha perdido o se encuentra en riesgo de perder o alterar de forma temporal o permanente el adecuado movimiento.

  Dentro de la fisioterapia podemos encontrar varias tecnologías sanitarias que colaboran con la herramienta principal, la mano. Se pueden citar algunas como electroterapia, hidroterapia, termoterapia o mecanoterapia. Ésta última se define como la utilización terapéutica e higiénica de aparatos mecánicos destinados a provocar y dirigir movimientos corporales regulados en su fuerza, trayectoria y amplitud [4].

- **Alcance del potencial psicológico y social**. La terapia ocupacional es el arte y ciencia de dirigir la respuesta del hombre a la actividad seleccionada para favorecer y mantener la salud, para prevenir la incapacidad, para valorar la conducta y para tratar o adiestrar a los pacientes con disfunciones físicas o psicosociales [5]. En terapia ocupacional se abordan las deficiencias en los componentes motores, cognitivos, sensoperceptivos y psicosociales que afectan al desempeño funcional de un individuo en las áreas de autovalimiento, productividad y esparcimiento.

2.3. SISTEMAS COMERCIALES

La rehabilitación ha avanzado notablemente con la tecnología, especialmente en el campo de la fisioterapia donde encontramos láser-terapia, magnetoterapia, hipertermia de contacto o terapia por ondas de choque radiales [6]. Existen en la actualidad laboratorios que cuentan con las últimas tecnologías y que desarrollan programas de fisioterapia orientados principalmente a la reparación de los daños neurológicos o secuelas ortopédicas [7].

Para ilustrar el alcance de estos avances se pueden citar tres ejemplos de tecnología al servicio de personas con discapacidad o que requieren de rehabilitación:

- **Análisis cinemático de la actividad diaria de beber de un vaso en una población con lesión cervical de la médula espinal** [8]. Se trata de un análisis realizado por la Unidad de Biomecánica y Ayudas Técnicas del Hospital Nacional de Parapléjicos de Toledo. En él comparan los

movimientos del brazo de un grupo de personas sin patologías con dos grupos de personas con diferentes niveles de lesión medular cervical durante el gesto de coger un vaso y beber. Utilizan un sistema de marcadores activos que envían una señal de posición que captan dos sensores. Esto permite hacer un modelo digital en el ordenador (véase Figura 1.1) y valorar rangos de movimiento, amplitudes, velocidades y otras variables. Estos datos se podrían usar posteriormente para una mejora en las prótesis o en sistemas de terapia con realidad virtual y captura del movimiento.

- **Silla de ruedas que se controla con la nariz para personas severamente discapacitadas** [9]. Esta silla de ruedas desarrollada por el instituto Weizmann en Israel se controla mediante el olfato (véase Figura 1.2). Tiene un gran uso práctico en las personas con discapacidades severas ya que este sentido es, generalmente, el único mecanismo que perdura en la mayoría de los pacientes. El control del olfato depende de la posición del velo del paladar, el cual está enervado por múltiples nervios craneanos. El dispositivo mide la presión nasal y la convierte en señales eléctricas. Se permite así que el olfato controle un activador con velocidades similares a la de una mano que utiliza un ratón o una palanca de un joystick.

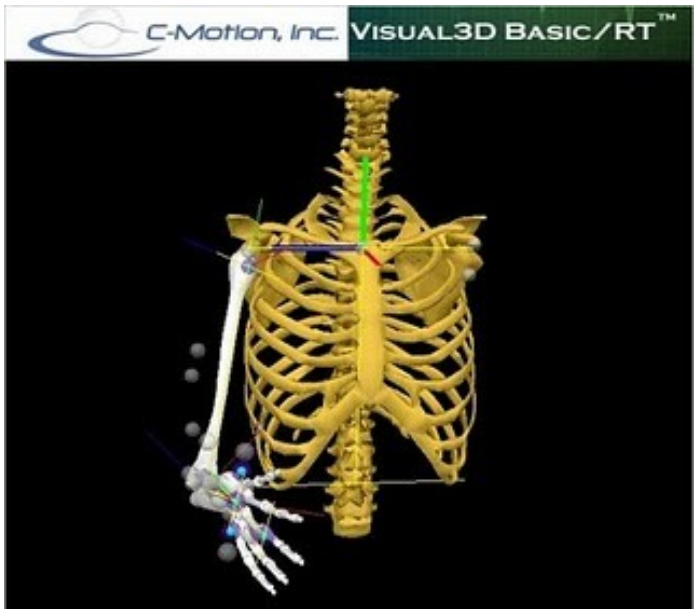

Figura 2.1. Vista del análisis cinemático del brazo en la actividad de beber

- **Lokomat** [10] [11]. Lokomat se trata de una terapia de locomoción mejorada programable a las necesidades de la persona. Se trata de un exoesqueleto que soporta el peso del paciente, el cual queda suspendido sobre una cinta rodante (véase Figura 1.3). Los motores que incorpora mueven las piernas del paciente trazando trayectorias que imitan los patrones de marcha fisiológicos. Una interfaz de usuario permite al terapeuta manejar el dispositivo de forma sencilla y adaptar los parámetros del entrenamiento. Posteriormente se puede consultar la evaluación del paciente y documentación del proceso de terapia.

Figura 2.2. Usuario controlando su silla de ruedas con su nariz

Figura 2.3. Vista de un paciente haciendo uso de Lokomat

Figura 2.4. Zone 40

2.4. SISTEMAS ALTERNATIVOS

Como alternativa a los sistemas tradicionales de rehabilitación o fisioterapia, actualmente se puede contar con otros sistemas que sustituyan, complementen o mejoren los existentes. Estos sistemas los encontramos en el **sector de los videojuegos**, ya sea gracias a sus mandos o a las consolas. Se proponen cuatro alternativas de las que se podrían disponer para un uso de rehabilitación.

Zone 40 [12]

Zone 40 es una miniconsola lanzada por Sega en verano de 2010. La consola contará con los juegos ya instalados. Está inspirada en la Wii de Nintendo, tanto en su diseño como en su jugabilidad (véase Figura 1.4). El aspecto de esta consola que la hace candidata a un posible uso en rehabilitación es el de sus mandos con sensores de movimiento y comunicación inalámbrica.

Playstation Move

Playstation Move es un sistema de control de videojuegos mediante sensores de movimiento para la Playstation 3 [13]. Se hace uso para ello de un mando principal con sensores de movimiento y una bola en su extremo que se ilumina. El sistema se complementa con la cámata Playstation Eye que se encarga de detectar la posición del mando principal.

El componente primario del sistema es el **controlador de movimiento**. Éste es un controlador con forma de tubo que permite al usuario interactuar con la consola a través del movimiento y la posición frente a ésta. En su interior encontramos dos sensores inerciales: un acelerómetro lineal de tres ejes y un sensor de velocidad angular de tres ejes. Estos se utilizan para controlar el movimiento global y la rotación respectivamente. También se usa un magnetómetro interno para calibrar la orientación del controlador en relación al campo magnético de la Tierra y ayudar a corregir el error acumulado (deriva) de los sensores inerciales. Los sensores inerciales pueden ser utilizados para navegación por estima en los casos en los que el seguimiento por la cámara es insuficiente.

Este componente cuenta además con una esfera en su extremo que puede brillar en varios colores con ayuda de un emisor LED RGB. Sobre la base de colores del entorno de usuario capturado por la cámara, el sistema selecciona un color que sea fácilmente distinguible en el resto de la escena. La luz sirve como marcador activo y cuya posición puede ser rastreada a lo largo del plano de la imagen por la cámara. La forma esférica y el tamaño uniforme conocido permite al sistema saber a través de la imagen a tamaño de la luz en qué posición se encuentra el usuario, permitiendo de esta manera un posicionamiento en tres dimensiones con alta precisión y exactitud. El cálculo de distancias realizado de esta manera hace que el controlador trabaje con la latencia de procesamiento mínimo en comparación con otras técnicas de control basadas en la cámara de Playstation.

Figura 2.5. Controlador de movimiento de Playstation Move

EL controlador dispone en su cara anterior de un gran botón primario de forma ovoide, los botones de acción clásicos de Playstation y un botón PS de tamaño regular y con una disposición similar a la que encontramos en un mando de Blu-Ray Disc. En sus laterales izquierdo y derecho encontramos un botón de selección y uno de inicio respectivamente. En la parte anterior se sitúa un disparador analógico o gatillo. En el extremo inferior se encuentra un puerto USB, un puerto de extensión y una correa para la muñeca.

Incluye una tecnología de respuesta háptica basada en vibración. Además de la referencia para el posicionamiento, la luz de la bola del controlador puede proveer una retroalimentación visual simulando efectos estéticos como puede ser el fogonazo de un arma o un pincel al pintar.

El segundo componente del sistema se trata del **controlador de navegación** (originalmente conocido como subcontrolador de movimiento). Es un controlador adicional diseñado para usarse en conjunción con el controlador de movimiento sujetándolo a una mano. Replica la funcionalidad principal de la parte izquierda de un mando inalámbrico estándar de Playstation. El controlador de navegación cuenta con una cruceta analógica, un pad direccional o dos disparadores analógicos o gatillos en la parte posterior. Cuenta además en su parte frontal con dos botones de acción y un botón PS. Teniendo en cuenta las funciones que desarrolla, un mando Sixaxis o un mando DualShock 3 pueden ser usados en lugar de éste.

Ambos controladores usan comunicación radio valiéndose del protocolo **Bluetooth** en su versión 2.0. Incorporan una batería de ion de litio la cual es cargada mediante un puerto USB Mini-B en el controlador.

Figura 2.6. Controlador de navegación de Playstation Move

El tercer elemento que completa el sistema es la cámara denominada Playstation Eye [14]. Se trata de un dispositivo **webcam** desarrollada por Sony para la Playstation 3. Es la sucesora de la Eye Toy, creada para la Playstation 2. La cámara es capaz de capturar video con una resolución de 640x480 píxeles a una frecuencia de 60 Hz o capturar video a resolución 320x240 píxeles a una frecuencia de 120 Hz. Estos valores suponen multiplicar por cuatro la resolución de la Eye Toy y doblar la frecuencia de muestreo de ésta.

Esta nueva cámara tiene el doble de sensibilidad que la anterior y ofrece una operación con mayor calidad cuando las condiciones de iluminación son bajas respecto al modo de uso normal. Incorpora dos lentes para hacer zoom de manera ajustable. Se selecciona manualmente girando el cilindro del objetivo el modo de funcionamiento, siendo de 56 grados de visión para una aplicación de chat o de 75 grados para el uso en aplicaciones de juegos físicos interactivos.

La Playstation Eye puede dar salida de vídeo sin comprimir o en formato comprimido usando el estándar JPEG. Todo el procesamiento de imagen tiene lugar a cabo en el microprocesador de tres celdas de la Playstation 3. El uso de la biblioteca de seguimiento de movimiento supone un gran impacto en el uso de memoria pero desde Sony afirman que estos efectos se reducirán al máximo. El uso de memoria para estas bibliotecas será de uno o dos megabytes del total del sistema.

La cámara incorpora un micrófono en un array de cuatro cápsulas con el cual puede emplear tecnologías para seguimiento y localización de voz de manera multidireccional, cancelación de eco y supresión de ruido de fondo. Esto permite al periférico ser usado para reconocimiento de voz y audio en ambientes ruidosos sin uso de un auricular. El micrófono opera en cada canal procesando muestras de 16 bits a una velocidad de 48 KHz y con una relación señal a ruido de 90 decibelios.

Figura 2.7 Playstation Eye

Este sistema se encuentra limitado en cuanto a número de jugadores debido a la comunicación inalámbrica. Solo se pueden usar cuatro controladores de movimientos en cada sesión de juego o dos controladores de movimiento con sus correspondientes controladores de navegación asociados.

Debido sus características se puede decir que sería un buen dispositivo para usar en ejercicios de rehabilitación por su detección precisa de movimientos. Además incorpora la posibilidad de reconocimiento tanto visual como de habla logrando así una mayor interactuación con el usuario. Gracias a estas capacidades sería un accesorio perfecto en rehabilitación para entrenar ABVD (Actividades Básicas de la Vida Diaria) [15]. La capacidad de interacción con todo el cuerpo es muy deseable en rehabilitación para que de esa manera se puedan entrenar la ingesta de alimento, la bebida, el aseo personal u otras ABVD. En estas aplicaciones el usuario se vería a si mismo de manera similar a como se mira en un espejo pero con la realidad aumentada por objetos y entornos digitales. Este efecto es la denominada “Realidad Aumentada” y se usa en otros sistemas de rehabilitación como IREX.

Kinect

Kinect (antes conocido con el nombre clave Protecto Natal) es un controlador de juego libre y entretenimiento de Microsoft para la videconsola Xbox 360 [16]. Permite a los usuarios controlar e interactuar con la Xbox 360 sin necesidad de usar mandos tradicionales o una interfaz táctil, solo usando gestos, comandos de voz y el propio cuerpo. Dispone de una cámara web, un escáner y un micrófono [17].

El sensor Kinect es una barra horizontal conectada a una pequeña base con un eje motor. Está diseñado para ser colocado longitudinalmente por debajo de la pantalla de visualización. El dispositivo cuenta con una **cámara RGB**, un **sensor de profundidad** y un **micrófono multi-array**, todos ellos ejecutando el software propietario de Microsoft que proporciona una captura de movimiento 3D de todo el cuerpo, reconocimiento facial y capacidad de reconocimiento de voz. El array de micrófonos permite que se lleve a cabo localización de la fuente acústica y supresión del ruido ambiente. La información de audio se transportara en cada canal de procesamiento de audio de 16 bits a una velocidad de 16 KHz.

Figura 2.8. Kinect

El sensor de profundidad se compone de un proyector de infrarrojos combinado con un sensor CMOS monocromo que permite a Kinect ver en tres dimensiones en cualquier condición de luz ambiental. El rango de detección del sensor de profundidad es ajustable, aunque con el software se calibre automáticamente basándose en el juego y en el ambiente físico del jugador. Kinect se basa en tecnología de software desarrollada internamente por Microsoft y en tecnología de cámara desarrollada por PrimeSense. El sistema permite el **seguimiento de hasta seis personas**, entre ellas dos agentes activos para el análisis de movimientos con una extracción de características de 20 articulaciones por jugador.

Kinect ofrece salidas de video a una velocidad de 30 Hz con una resolución de color de 32 bits VGA (640x480 píxeles) y la secuencia de video monocromo utilizado para detección de profundidad usando 16 bits QVGA (320x240 píxeles) y ofreciendo 65536 niveles de sensibilidad. El sensor tiene un límite práctico que va desde los 1.2 metros hasta los 3.5 metros de distancia. Tiene un campo angular de visión de 57º en horizontal y 43º en vertical. El eje motor es capaz de inclinarse 27º hacia arriba o hacia abajo.

Las características que tiene Kinect le hacen de un muy válido instrumento para rehabilitación. Su funcionamiento se basaría en el reconocimiento del paciente en 3 dimensiones y en el reconocimiento facial y del lenguaje. Sería de un uso más fácil ya que este sistema ofrece una interfaz de usuario natural. Estas interfaces son aquellas que no utilizan dispositivos de entrada y en lugar de estos usamos nuestras manos rompiendo el paradigma actual de las metáforas visuales [18]. Ofrecería una gran variedad de ejercicios para una persona o varias pudiendo así realizar el ejercicio en grupo, hecho menos tedioso para el paciente.

Wii [19]

Wii es una videoconsola de sobremesa producida por Nintendo y en cuyo desarrollo colaboraron IBM y ATI. La característica más distintiva de la consola es su **mando inalámbrico**, el cual puede usarse como un dispositivo apuntador además de cómo detector de movimiento en tres dimensiones. Dispone además de conexión a Internet por la cual puede recibir mensajes y actualizaciones.

Figura 2.9. Wii

El mando de Wii utiliza incorpora un **acelerómetro** que, en combinación con unos **LEDs infrarrojos** situados en la barra de sensores, permiten localizar en un espacio en tres dimensiones al usuario. La interactuación se produce mediante gestos físicos y presión sobre los botones. El mando se conecta a la consola de manera inalámbrica mediante el protocolo de comunicación **Bluetooth**. Incluye además un vibrador y un altavoz interno. Se ofrece con una correa de sujeción para evitar posibles lanzamientos de mando accidentales.

La consola Wii ofrece un considerable número de accesorios. Los más destacados por tener capacidad propia de toma de datos son el **Nunchuk** y la **Wii Balance Board**. El Nunchuk se trata de una extensión al mando original y se compone de un control analógico y dos botones de acción. En su interior cuenta con un acelerómetro que le dota de detección de movimiento. La Wii Balance Board se trata de una tabla a la cual hay que subirse para interactuar con el juego. Tiene en su interior sensores de presión que la hacen capaz de medir el peso y la posición del centro de gravedad del usuario.

Como en los anteriores sistemas, la capacidad de Wii para detectar movimiento y el apuntado en pantalla la hacen apta para un uso en rehabilitación. En la actualidad es un sistema que funciona en varios hospitales en las áreas de rehabilitación. Su uso se aplica tanto a lesiones físicas como mentales, como se explicará más adelante.

2.5. REHABILITACIÓN CON EL USO DE WII

La Wii como herramienta para rehabilitación lleva tiempo en uso en las áreas de rehabilitación de hospitales en varios países. Se ha formado un término para la denominación de este tipo de rehabilitación: **Wii-Habilitation** o **Wii-hab**. La consola se está utilizando como una actividad para complementar los tratamientos convencionales en lugar de ser usado como un tratamiento independiente.

Las características de la rehabilitación con Wii son [20]:

- **Wii como herramienta de motivación.** Además de ofrecer oportunidades para explorar el movimiento activo y facilitar el desarrollo de movimiento mejorado, el juego puede ser utilizado para distraer de otras actividades durante fisioterapia.

- **Competencia entre usuarios.** Hay juegos de Wii que ofrecen la posibilidad de juego de dos usuarios. Esto fomentaría la competencia, la cual nos llevaría a una mayor motivación a la hora de ejercitarse.

- **Uso de una férula.** Se puede utilizar una férula para pacientes que no tienen el suficiente agarre o destreza en las manos para acceder a los juegos (véase Figura 1.10).

- **Uso de Wiimote adaptado.** Es posible adaptar el mando de Wii para que sea más accesible en cuanto a pulsación de botones se refiere para personas que tienen limitaciones en las extremidades (véase Figura 1.11).

- **Sentarse sobre la Wii Balance Board.** Determinados juegos se pueden jugar en posición sentada. Es una herramienta útil para que el usuario pueda apreciar mejor la posición del cuerpo (véase Figura 1.12).

- **Trabajo en tareas que requieran secuenciación.** Se intenta mejorar en actividades que implican acciones repetidas de manera secuencial.

- **Simulación de actividades.** Tener una actividad cotidiana haciéndolo de manera virtual.

- **Mejorar las velocidades de reacción.** Se pueden mejorar las capacidades dependientes del sistema nervioso, las cuales reciben un estímulo, lo procesan y responden si es necesario.

- **Uso de la función de guardado de datos en la Wii.** El uso principal es el de permitir a los pacientes que usan regularmente la Wii en casa poder seguir usando sus datos en el hospital y viceversa.

- **Trabajar para la mejora de la destreza.** Se trabaja en actividades cotidianas para que el paciente sea más capaz y las ejecute con mayor destreza.
- **Estiramiento antes del uso.** Debido a la falta de resistencia a la hora de realizar los movimientos, el usuario es susceptible de lesionarse si antes no ha calentado los músculos.
- **Descanso durante el uso.** Se recomienda no excederse en el ejercicio cuando se juega a Wii ya que es probable que aparezcan lesiones debidas al sobreuso del Wiimote.

Figura 2.10. Usuario con adaptación en un accesorio del mando de Wii

Figura 2.11. Wiimote adaptado

Figura 2.12. Wii Balance Board en soporte para realizar actividades de rehabilitación

La Wii en terapia se puede utilizar para alcanzar diferentes objetivos [20]:

Promover el uso del tren superior

Los controladores principales de Wii (Wii Remote y Nunchuk) se utilizan con las manos. Hay varias formas principales de uso del movimiento dentro de los juegos:

- Actividades deportivas. Dentro de esta categoría encontramos al boxeo y al tenis como las actividades más enérgicas.
- Sacudida del mando. El movimiento repetitivo trabaja los músculos y contribuye a promover una mayor resistencia.
- Mini Juegos. Se trata de pequeños juegos que requieren de habilidad y control fino.

La Wii como herramienta dentro de la terapia se puede usar para mejorar las siguientes habilidades motoras de las extremidades superiores:

- Movimiento fino.
- Movimiento fuerte.

- Uso combinado y coordinado de movimientos.
- Destreza.
- Control bilateral del movimiento.
- Velocidad de reacción.

Promover la actividad del tren inferior

Existen dos maneras de ejercitar las piernas mediante el uso de la Wii Balance Board:

- En posición erguida sobre la tabla.
- En posición sentada. Se pueden realizar los ejercicios estando en una posición sentada y con los pies sobre la tabla. Este método es ideal para los pacientes con movilidad limitada o impedida por un tiempo.

Se pueden reforzar las extremidades inferiores utilizando los controladores y realizando ejercicios de fitness que incluyen un aumento de fuerza y no se requiere de Wii Balance Board. La terapia se puede centrar en el movimiento y desarrollo de fuerza del tren interior o en el control sobre la transferencia de peso sobre la tabla.

Mejora del equilibrio

Se puede trabajar para mejorar las habilidades de equilibrio que ayudan a mantener control de la postura y los músculos que ayudan la estabilidad. El equilibrio requiere para el cuerpo trabajar ambos lados por igual y juntos para mantener la postura deseada. La principal forma para trabajar el equilibrio es utilizar la Wii Balance Board.

Hay dos modalidades para el trabajo del equilibrio:

- En posición erguida sobre la tabla. Se utilizará esta posición si se tiene un equilibrio razonable.
- Sentado sobre la tabla. Si no se tiene suficiente equilibrio, los ejercicios pueden ser realizados sentándose sobre la tabla. Se ganará en estabilidad y se trabajarán los músculos del abdomen. Hacer el ejercicio de esta manera es una gran manera de reintroducir el movimiento del cuerpo superior.

Mejora de la coordinación

La propia interacción con la consola mediante el juego provoca la necesidad de una gran coordinación. Esta es necesaria ya que se presenta información y se ha de reaccionar a ella físicamente con el fin de crear el

movimiento deseado y mantener el equilibrio. Se pueden citar algún juego que permita trabajar el control y la coordinación como son los de esquí, los de fútbol o los que usan planos inclinados. Se logra los mayores niveles de coordinación con los juegos de yoga o de snowboard.

Mejora de la aptitud cardiovascular

El juego Wii Fit tiene una opción para correr. El Wiimote se coloca en el bolsillo del usuario u otro lugar especificado y éste detecta el movimiento. Este juego no utiliza la Wii Balance Board pero es parte de Wii Fit ya que se basa en la capacidad de los miembros inferiores para el trabajo del corazón y de los pulmones.

Los juegos que requieren un control poco fino y movimientos repetitivos ofrecen la oportunidad a los usuarios de incrementar el ritmo cardíaco y la circulación. Ejemplos de estos juegos son: Wii Sports Boxing, Wii Fit y los juegos olímpicos de correr y nadar.

2.6. HERRAMIENTAS SOFTWARE

Se citarán y explicarán a continuación posibles librería o drivers que se podrían utilizar para programar aplicaciones utilizando el Wiimote como control de las mismas:

- **motej** [21]

  motej es una librería Java para la comunicación con el Wiimote y que se encuentra bajo licencia ASL 2.0. Consiste en dos paquetes principales: la librería motej y los extras motej. La librería motej provee de un acceso básico al Wii Remote y depende solamente de una implementación Bluetooth. Los extras motej dependen de la librería motej y ofrecen funcionalidad adicional.

  La librería motej ofrece las siguientes características:

  - Descubrimiento y conexión con dispositivos Wiimote.
  - Cámara de infrarrojos en sus diferentes modos.
  - Acelerómetro.
  - Vibrador.
  - LEDs indicadores del jugador.
  - Botones.
  - Lectura de la memoria EEPROM.

- Escritura de los registros del Wiimote.
- Información de status del mando.
- Datos de calibración.

Los extras de motej, por otra parte, nos ofrecen las siguientes características:

- Soporte para controladores de extensiones: Wii Balance Board, mando clásico y Nunchuk.
- Algunas clases de ayuda o adaptación que hacen más fácil el desarrollo con el Wiimote.

❖ **WiiRemoteJ** [22]

WiiRemoteJ es una biblioteca Java desarrollada para permitir el acceso fácil a los desarrolladores de Java el hardware de Wii Remote. WiiRemoteJ se sitúa encima de la API de Java Bluetooth permitiendo que sea independiente de la plataforma.

Soporta todas las funciones principales del mando de Wii incluyendo: infrarrojo, acelerómetro, extensiones (mando clásico, Nunchuk, guitarra soportada por defecto y posibles extensiones personalizadas), entrada de botones, LEDs, vibración, altavoz, datos de lectura/escritura y soporte a múltiples controles remotos. También es compatible con la Wii Balance Board.

❖ **WiiYourself!** [23]

WiiYourself! es una librería escrita en C++ para el control total de las características del Wiimote. Está basada en una versión anterior denominada Managed Wiimote Library, la cual es extendida considerablemente. Tiene la limitación de que su ejecución está limitada al sistema operativo Windows. Las características que nos ofrece son:

- Soporte a múltiples Wiimotes.
- Soporte a extensiones.
- Lectura de batería, botones, acelerómetro, infrarrojos.
- Estimación de la orientación.
- Posibilidad de establecimiento de LEDs y vibrador.
- Soporta todas las pilas de Bluetooth.
- Soporte experimental de altavoz.

- Detección de pérdida o rotura de la conexión.

❖ **Wiim** [24]

Wiim es un set simple de clases escritas en C++ que permiten la conexión con el mando Wii Remote a través de la interfaz HID de Windows para enviar o recibir comandos. Se pueden recibir las pulsaciones de los botones y los datos de la detección de movimiento así como establecer el vibrador y los LEDs. No se soportan los dispositivos de infrarrojos o cualquier extensión.

❖ **wiiuse** [25]

wiiuse es una librería escrita en lenguaje de programación C y que conecta con el Wiimote. Soporta detección de movimiento, seguimiento de infrarrojo y las extensiones que corresponden al Nunchuk, mando clásico y guitarra. Su característica de no-bloqueo y de uso de un solo hilo la hacen una API limpia y ligera. Se encuentra bajo licencia GNU GPLv3 y GNU LGPLv3.

El aspecto más interesante de esta librería lo ofrece su compatiblidad. Es compatible con Linux y con las versiones 2000, XP y Vista de Windows, lo cual resulta en una mayor facilidad a la hora de portar juegos de una plataforma a otra.

❖ **libcwiid** [26]

Cwiid forman una colección de herramientas de Linux escritas en C para la conexión al Wiimote. CWiid está formado por los siguientes componentes:

- libcwiid. Es la librería en la que se incluyen todas las funciones para el manejo del mando. Incluye un módulo para el desarrollo de aplicaciones en Python.
- wminput. Es un controlador con arquitectura de complemento al que se le puede dar uso de ratón o joystick.
- wmgui. Es una interfaz gráfica capaz de monitorizar distintos valores del Wiimote y representarlo en pantalla.
- wmdemo. Es una aplicación de demostración y testeo del mando.

2.7. JUSTIFICACIÓN DE LA SOLUCIÓN ADOPTADA

Teniendo en cuenta lo expuesto, tanto Playstation Move como Kinect o Wii son herramientas válidas para rehabilitación. Zone sería menos indicado pues está muy limitado y, por el momento, solo está disponible en Reino Unido. Respecto a los otros tres sistemas, actualmente, lo más adecuado será usar Wii. Esto se debe a varias razones:

- Disponibilidad. Wii y Playstation Move son los únicos sistemas disponibles. Kinect todavía no se han lanzado al mercado. Playstation Move tiene una vida comercial mínima comparada con la de Wii.

- Wii es suficientemente conocido y probado por las personas ya que está disponible desde 2006. Los otros sistemas se conocen en profundidad desde hace meses y aun hay características por conocer con detalle.

- Existen varias herramientas (librerías software y drivers) para la programación de aplicaciones utilizando los periféricos de Wii. Todavía no se ha realizado ingeniería inversa para los otros dos sistemas.

- El precio de los dispositivos Wii es menor al de Kinect. Playstation Move se sitúa al mismo nivel que Wii en precio de sus controladores.

El sistema operativo en el que se ejecutarán las aplicaciones es Linux. Se elige este sistema operativo por las ventajas que aporta frente a otros, por ejemplo:

- Robustez, estabilidad y fiabilidad.
- Capacidad de funcionamiento en equipos que no son excesivamente potentes.
- Soporte de gran variedad de entornos gráficos (KDE, GNOME…)
- Existe gran cantidad de documentación libre.
- El sistema operativo es software libre, lo que implica no pagar nada por él.
- Es un sistema operativo muy seguro.
- Eficiencia de su código, lo cual hace que la velocidad de las aplicaciones sea superior.

El lenguaje de programación que se usará será Python, siendo las aplicaciones scripts escritos en este lenguaje. Se trata de un lenguaje de programación de alto nivel y multiparadigma ya que soporta orientación a objetos, programación imperativa y programación funcional [27]. Es un lenguaje interpretado, usa tipado dinámico, es fuertemente tipado y es multiplataforma. Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores. La versión que se utiliza para la elaboración y prueba de los programas es la 2.6.5.

Para el desarrollo de la interfaz gráfica de usuario se usará Pygame. Pygame es un conjunto de módulos en lenguaje Python que permite la creación de videojuegos en dos dimensiones de una manera sencilla [28]. Debido al lenguaje, se puede prototipar y desarrollar rápidamente pudiendo alcanzarse resultados profesionales. Se puede utilizar además otros programas multimedia o interfaces gráficas de usuario, modo que se utilizará en las aplicaciones.

La herramienta elegida para el manejo del hardware es la librería libcwiid. Se elige esta opción porque en su API encontramos cubiertas, casi en su totalidad, todas las funciones necesarias para el desarrollo. Además, libcwiid ofrece la capacidad de programar en Python, lo que incrementa la velocidad del proceso de desarrollo de aplicaciones.

2.8. RESUMEN

En este capítulo se nos han expuesto ejemplos de sistemas actuales de rehabilitación y como ayudan a estos procesos de recuperación. Se han presentado alternativas actuales basadas en periféricos de consolas por su mayor capacidad de atracción y entretenimiento manteniendo la idea de recuperación de las condiciones normales del paciente. Un ejemplo de estas alternativas es el uso que se da a Wii. Además se han presentado diferentes tipos de librerías que nos ayudan en el desarrollo de aplicaciones a medida.

Tras estas consideraciones se expone la elección tomada para realizar las aplicaciones deseadas en este proyecto. Se toma el hardware ofrecido por los periféricos de Nintendo, el sistema operativo Linux y la librería libcwiid por su capacidad para desarrollo en Python. Estos elementos formarán la base que queremos establecer y sobre ellos profundizaremos en posteriores apartados describiendo el hardware y desarrollando software diseñado específicamente.

Capítulo 3

Wii Remote

3.1. INTRODUCCIÓN

En este capítulo se describe el Wii Remote atendiendo a sus características técnicas y a los accesorios que complementan su uso. Además se expondrá el API para la programación de aplicaciones usando este componente. Por último se explica el funcionamiento de dos aplicaciones desarrolladas usando el Wii Remote como control de las mismas.

El Wii Remote es el mando principal de la consola Wii de Nintendo. También es conocido como Wiimote o control remoto. Sus características más destacables son su capacidad para detectar movimientos en el espacio y la posibilidad de apuntar objetos en la pantalla estableciendo una conexión inalámbrica con la consola.

Típicamente las videoconsolas se controlaban a través de joysticks o mandos con botones o una combinación de ambos. El diseño del Wiimote es innovador ya que permite nuevas posibilidades dentro del campo de los mandos. Implementa por primera vez un control gestual inalámbrico de manera que los usuarios pueden interaccionar con los juegos mediante movimientos de los brazos.

3.2. DESCRIPCIÓN TÉCNICA

3.2.1. DISEÑO

El Wiimote tiene una forma similar a un mando a distancia de un televisor o un reproductor de DVD. Sus dimensiones son 14,6x3,5x3,1 cm. Presenta un total de ocho botones en la superficie anterior y un botón en la superficie posterior. Éstos son por orden de situación de arriba abajo y de izquierda a derecha: “POWER”, “Cruceta direccional”,” A”, “-“, “HOME”, “+”, “1”, “2” y “B” en la parte posterior (véase Figura 3.1). Adicionalmente incluye en la parte frontal un altavoz y cuatro LEDs que indican el número de jugador al que corresponde el mando y el tiempo de vida de la batería. En la parte inferior se le une una correa de seguridad que se sujeta a la muñeca para evitar que se suelte el mando de forma accidental y se dañe.

Figura 3.1. Wiimote en sus diferentes vistas

3.2.2. DETECCIÓN DE MOVIMIENTO

El mando de Wii permite realizar acciones tales como mover un bate, pegar un golpe con una espada, girar una llave o conducir un vehículo moviendo un volante. Todas estas acciones, que se representan en tiempo real en la pantalla, son capaces de llevarse a cabo gracias a la detección de movimiento implementada mediante un acelerómetro incorporado. Con él se detectan los movimientos lineales, los giros y las inclinaciones (véase Figura 3.2).

El acelerómetro incorporado en el Wiimote es el chip MEMS ADXL330 de Analog Devices (véase Figura 3.3). Se trata de un acelerómetro de medidas 4x4x1.45 mm. Su rango típico de detección es de ±3.6G y tiene una sensibilidad máxima de 330 mV/g. Presenta un amplio rango de funcionamiento en temperatura y es capaz de medir la aceleración estática y dinámica.

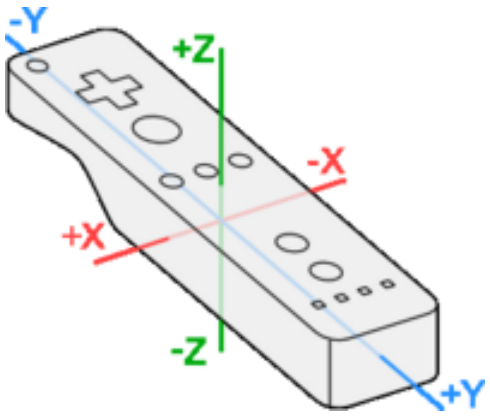

Figura 3.2. Orientación de los ejes del acelerómetro del Wiimote

Figura 3.3. Chip ADXL330 integrado en la placa del Wiimote

Al margen de especificaciones técnicas, el funcionamiento de los integrados MEMS (Sistemas micro electro-mecánicos) es similar al de una máquina aunque se trate de un diminuto chip de silicio. Al igual que éste, los dispositivos MEMS ganan gran importancia en montajes de todo tipo y dimensión hoy en día por las características que presentan. Si se estudia el funcionamiento de acelerómetros se puede comprobar que miden la aceleración y la dirección del movimiento comprobando los cambios en los electrones de su interior. Para una mejor comprensión se puede asimilar a una pareja de placas dentro del sensor, siendo una fija y otra móvil. Los electrones que rodean la placa se mueven con el movimiento de la placa y si medimos la capacitancia seremos capaces de capturar los datos requeridos. El dispositivo puede así enviar datos de movimiento relativos a todos los ejes de coordenadas además de los datos de aceleración detectando giros, movimientos en el aire o inclinaciones.

Nota histórica: Durante el desarrollo del Wiimote Nintendo invirtió en un acuerdo con Gyration, empresa dedicada a los giroscopios. De esta relación dio lugar a una de las funciones del mando actual. Pese a la investigación y el trabajo conjunto el componente de Gyration no se incluyó en la versión definitiva del mando ya que el acelerómetro elegido cumplía con la función de detección de giros y, para la función de apuntado, se optó por el sensor óptico [29].

3.2.3. FUNCIÓN PUNTERO

La función de puntero del Wiimote se implementa mediante el cálculo de distancia entre haces de luz infrarroja. Estas distancias son captadas por un sensor situado dentro del mando. Para más información, véase Capítulo 4: Wiimote+IR LEDs.

3.2.4. COMUNICACIÓN

El Wiimote se conecta de manera inalámbrica a la consola mediante el protocolo Bluetooth, especificado en la norma 802.15.1. Es posible conectarlo también a otros dispositivos que se comuniquen con este protocolo. Para la comunicación se utiliza el chip BCM2042 de Broadcom Technologies (véase Figura 3.5). Toda la información enviada por el mando (sensor, botones, cámara) se envía de manera continua al extremo receptor.

La comunicación se realiza en la banda de 2.4 GHz y proporciona una velocidad de transmisión de 2.1 Mbits/s. Esta velocidad es lo suficientemente alta para que la latencia sea comparable a la de los mandos cableados.

El protocolo Bluetooth sólo permite un servidor y siete clientes por lo que únicamente se podrán usar siete Wiimotes de forma simultánea. En un modo

multijugador con cuatro jugadores, éstos no podrían utilizar cada uno dos mandos debido a la limitación establecida por el protocolo.

Para realizar el emparejamiento entre mando y dispositivo se ha de pulsar el botón SYNCRO situado en el espacio dedicado a las pilas del mando o los botones 1 y 2 al mismo tiempo y, en el dispositivo a conectar, ha de estar operativa la opción de comunicación Bluetooth (véase Figura 3.4). En la consola Wii se ha de pulsar el botón SYNCRO situado tras la tapita frontal y, a continuación, se asigna un canal a cada Wiimote pulsando otro botón. El identificador obtenido se almacenará en la memoria del mando para mantener la asignación en futuros encendidos.

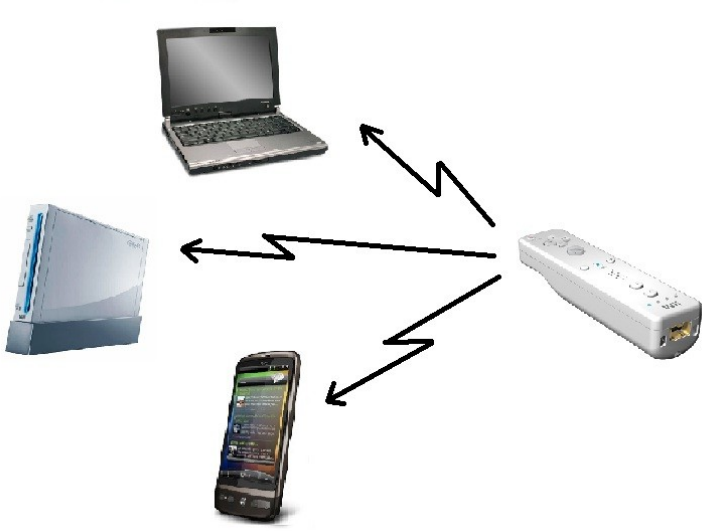

Figura 3.4. Conectividad Bluetooth del Wiimote

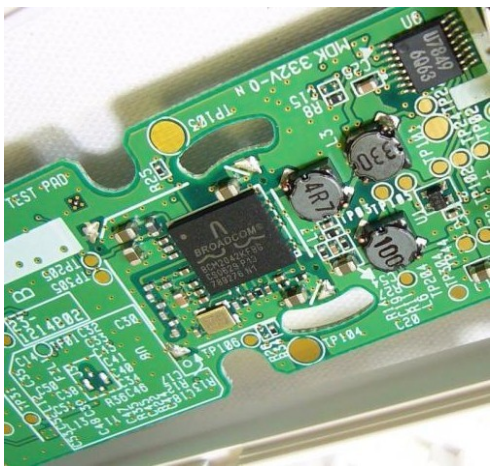

Figura 3.5. Chip BCM2042 integrado en la placa del Wiimote

3.2.5. FUNCIONES ADICIONALES

Además de los elementos ya introducidos, el Wiimote incorpora un vibrador, un altavoz y una memoria.

El sistema de vibración o “Rumble” proporciona una respuesta háptica, es decir, señales que no son auditivas o visuales. La fuerza de vibración es variable pero no se presenta ningún sistema de resistencia o “force feedback”. Este sistema no interfiere con los acelerómetros.

El Wii Remote incorpora un altavoz propio e independiente en su interior con salida por la cara anterior del mando. El altavoz es de un material piezo-eléctrico y su diámetro es de 21 mm. El uso habitual del altavoz hará que la sensación de inmersión se vea amplificada en gran medida.

El Wiimote contiene un chip de 16 Kb de memoria EEPROM. EL host puede usar hasta 6 Kb para leer y almacenar datos. Parte de esta memoria está disponible para almacenar hasta 10 avatares Mii y que pueden ser transportados para su uso en otra consola. En este caso, una vez almacenados los datos Mii, la memoria disponible sería de 4 Kb.

3.2.6. AUTONOMÍA

El Wiimote se alimenta con dos pilas AA comunes. La autonomía máxima son 60 horas pero cae hasta las 30 horas si se utiliza constantemente la función de puntero. El nivel de batería nos viene indicado por los cuatro LEDs de la parte frontal al encender el sistema. El funcionamiento es como sigue: un LED parpadea si queda menos de un cuarto de la duración, dos entre el cuarto y la media duración, tres hasta los tres cuartos de duración y los cuatro LEDS entre los tres cuartos y la totalidad de la duración.

Figura 3.6. Placa del Wiimote

El sistema de alimentación cuenta además con un condensador de 3'3 mF que proporciona una fuente temporal de energía con movimientos rápidos del Wiimote cuando la conexión a las pilas pudiera ser interrumpida de manera temporal.

3.3. ACCESORIOS

3.3.1. ACCESORIOS FUNCIONALES

Los accesorios funcionales son aquellos que complementan al Wiimote con alguna función añadida. Estos accesorios se conectan al mando principal por el puerto de expansión que este incorpora en la parte inferior. Se trata de un puerto propietario de Nintendo el cual manda datos serie de manera síncrona usando una interfaz a dos hilos de manera bidireccional. Utiliza para la transmisión un bus I^2C [30].

Figura 3.7. Accesorios funcionales para Wii, de izquierda a derecha y de arriba abajo: Nunchuk, Mando clásico, Wii Guitar y Wii Motion Plus.

Los accesorios funcionales actuales son (véase Figura 3.7):

- **Nunchuk**: Es un joystick analógico con dos botones más. Tiene capacidad para detectar movimientos ya que en su interior se localiza un acelerómetro. El chip elegido es el LIS3L02AL de STMicroelectronics. Tiene una sensibilidad de 660 mV/g y presenta un rango de detección de ±2G. Proporciona gran inmunidad ante vibraciones, golpes y temperaturas extremas.

- **Control clásico**: Se trata del mando que se utilizaba en los títulos de Nintendo. Contiene los mismos botones que están presentes en el Wiimote. Sirve para jugar a juegos bajados del Canal Tienda Wii y algunos juegos Wii compatibles.

- **Wii Guitar**: Es un accesorio con forma de guitarra en el que se inserta el Wiimote. Sirve para el juego Guitar Hero de Wii.

- **Wii Motion Plus**: Es un accesorio que añade precisión extra al mando (véase Capítulo 5: Wii Remote+Wii Motion Plus).

Figura 3.8. Accesorios soporte para Wii: Wii Zapper y Wii Wheel

3.3.2. ACCESORIOS SOPORTE

Los accesorios soporte son aquellos que no aportan más funcionalidad nueva sino que hacen de apoyo a los mandos actuales para un determinado juego o aplicación. Se tratan de carcasas en las que acoplar el Wiimote (véase Figura 3.8). Entre los muchos que se pueden encontrar en el mercado, los más populares son:

- **Wii Zapper**: Es un accesorio con forma de pistola en el que se da la posibilidad de acoplar el Wiimote y el Nunchuk. Sirve para los juegos de acción en primera persona y en los de puntería.
- **Wii Wheel**: Se trata de un accesorio con forma de volante. Se inserta dentro de él y permite un mayor control en los juegos de carreras.

3.4. API PARA LA PROGRAMACIÓN

El API de la librería libcwiid [31], la cual controla el Wiimote, es como sigue:

- `cwiid_open`
  - Sinopsis

    ```
    cwiid_wiimote_t *cwiid_open(bdaddr_t *bdaddr,
    int flags);
    ```

  - Parámetros

    `bdaddr`: puntero a la dirección del dispositivo Bluetooth del Wiimote.

    `flags`: indicadores de opciones a ser habilitadas.

- Descripción

  Establece una conexión con el Wiimote. Para conectarse a cualquier Wiimote disponible se ha de pasar `*BDADDR_ANY` en el campo `bdaddr`. En este caso, `bdaddr` se llenará con la dirección del Wiimote conectado.

- Valor de retorno

  `cwiid_wiimote` manejado, `NULL` o `error`.

- `cwiid_close`

  - Sinopsis

    ```
    int cwiid_close(cwiid_wiimote_t *wiimote);
    ```

  - Parámetros

    `cwiid_wiimote` manejado.

  - Descripción

    Desconecta el Wiimote.

  - Valor de retorno

    0 si ha resultado exitoso o un valor distinto si ha habido un error.

- `cwiid_get_id`

  - Sinopsis

    ```
    int cwiid_get_id(cwiid_wiimote_t *wiimote);
    ```

  - Parámetros

    `wiimote`: `cwiid_wiimote` manejado.

  - Descripción

    Función de acceso al identificador del Wiimote, el cual se garantiza que es único entre todos los Wiimotes en un único proceso.

  - Valor de retorno

    Identificador del Wiimote.

- `cwiid_set_data`
    - Sinopsis

      ```
      int cwiid_set_data(cwiid_wiimote_t *wiimote,
      const void *data);
      ```

    - Parámetros

      `wiimote`: `cwiid_wiimote` manejado.

      `data`: puntero a los datos de usuario.

    - Descripción

      Función modificadora del campo de datos del usuario.

    - Valor de retorno

      0 si ha resultado exitoso o un valor distinto si ha habido un error.

- `cwiid_get_data`
    - Sinopsis

      ```
      const void *cwiid_get_data(cwiid_wiimote_t
      *wiimote);
      ```

    - Parámetros

      `wiimote`: `cwiid_wiimote` manejado. .

    - Descripción

      Función de acceso a los datos de usuario.

    - Valor de retorno

      Puntero a los datos de usuario establecidos por `cwiid_set_data`.

- `cwiid_enable`
    - Sinopsis

      ```
      int cwiid_enable(cwiid_wiimote_t *wiimote, int
      flags);
      ```

- Parámetros

  `wiimote`: `cwiid_wiimote` manejado.

  `flags`: indicadores de opciones a ser habilitadas.

- Descripción

  Los indicadores de opciones se pueden habilitar con `cwiid_connect` o con `cwiid_enable`. Los indicadores de opciones y sus significados son los siguientes:

  - `CWIID_FLAG_MESG_IFC`: Habilita las interfaces basadas en mensajes.
  - `CWIID_FLAG_CONTINOUS`: Habilita los reportes continuos por parte del Wiimote.
  - `CWIID_FLAG_REPEAT_BTN`: Entrega un mensaje de botón para cada valor del botón recibido, incluso si no ha cambiado.
  - `CWIID_FLAG_NONBLOCK`: Hace que `cwiid_get_mesg` falle en vez de bloquearlo si no hay mensajes listos.

- Valor de retorno

  0 si ha resultado exitoso o un valor distinto si ha habido un error.

- `cwiid_disable`

  - Sinopsis

    ```
    int cwiid_disable(cwiid_wiimote_t *wiimote,
    int flags);
    ```

  - Parámetros

    `wiimote`: `cwiid_wiimote` manejado.

    `flags`: indicadores de opciones a ser habilitadas.

  - Descripción

    Deshabilita los indicadores de opciones habilitados por `cwiid_connect` o `cwiid_enable`.

- Valor de retorno

  0 si ha resultado exitoso o un valor distinto si ha habido un error.

- `cwiid_set_mesg_callback`

  - Sinopsis

    ```
    typedef void
    cwiid_mesg_callback_t(cwiid_wiimote_t *, int,
    union cwiid_mesg []);

    int cwiid_set_mesg_callback(cwiid_wiimote_t
    *wiimote, cwiid_mesg_callback_t *callback);
    ```

  - Parámetros

    `wiimote`: `cwiid_wiimote` manejado.

  - Descripción

    Establece la función de retorno, la cual es llamada cuando un conjunto de instrucciones se reciben del Wiimote. La opción `CWIID_FLAG_MESG_IFC` ha de estar activada para que la función de retorno sea llamada. Pasando `NULL` a la función de retorno se cancela la función actual en caso de existir.

  - Valor de retorno

    0 si ha resultado exitoso o un valor distinto si ha habido un error.

- `cwiid_get_mesg`

  - Sinopsis

    ```
    int cwiid_get_mesg(cwiid_wiimote_t *wiimote,
    int *mesg_count, unión cwiid_mesg *mesg[]);
    ```

  - Parámetros

    `wiimote`: `cwiid_wiimote` manejado.

    `mesg_count`: puntero al contador de mensajes.

    `mesg`: puntero al array de mensajes.

  - Descripción

Devuelve el último conjunto de instrucciones no entregado. Los mensajes son entregados a través de una función de retorno o a `cwiid_get_mesg`.

- Valor de retorno

  0 si ha resultado exitoso o un valor distinto si ha habido un error.

- `cwiid_get_state`

  - Sinopsis

    ```
    int cwiid_get_state(cwiid_wiimote_t *wiimote,
    struct cwiid_state *state);
    ```

  - Parámetros

    `wiimote`: `cwiid_wiimote` manejado.

    `state`: puntero a la estructura de estado del Wiimote.

  - Descripción

    Devuelve el estado actual del Wiimote.

  - Valor de retorno

    0 si ha resultado exitoso o un valor distinto si ha habido un error.

- `cwiid_command`

  - Sinopsis

    ```
    int cwiid_command(cwiid_wiimote_t *wiimote,
    enum cwiid_command command, int flags);
    ```

  - Parámetros

    `wiimote`: `cwiid_wiimote` manejado.

    `command`: orden a ser ejecutada.

    `flags` opciones específicas de la orden.

  - Descripción

    Emite una orden al Wiimote. Las órdenes disponibles y sus opciones asociadas son las siguientes:

- `CWIID_CMD_STATUS`: Pide un mensaje de estado
- `CWIID_CMD_LED`: Establece el estado de los LEDs. Los siguientes indicadores pueden ser a nivel de bit:
  - `CWIID_LED1_ON`
  - `CWIID_LED2_ON`
  - `CWIID_LED3_ON`
  - `CWIID_LED4_ON`
- `CWIID_CMD_RUMBLE`: Establece el estado del vibrador. Se ha de colocar el valor 0 para un estado apagado y otro valor para estar encendido.
- `CWIID_RPT_MODE`: Establece el modo de presentación de informes del Wiimote, el cual determina que periféricos están activados y que datos son recibidos por el ordenador. Las siguientes opciones se pueden considerar a nivel de bit:
  - `CWIID_RPT_STATUS`
  - `CWIID_RPT_BTN`
  - `CWIID_RPT_ACC`
  - `CWIID_RPT_IR`
  - `CWIID_RPT_NUNCHUK`
  - `CWIID_RPT_CLASSIC`
  - `CWIID_RPT_EXT`

- Valor de retorno

  0 si ha resultado exitoso o un valor distinto si ha habido un error.

- `cwiid_read`

  - Sinopsis

```
int cwiid_read(cwiid_wiimote_t *wiimote,
unit8_t flags, uint32_t offset, uint16_t len,
void *data);
```

- Parámetros

  `wiimote`: `cwiid_wiimote` manejado.

  `flags`: indicadores para la lectura.

  `offset`: dirección de comienzo.

  `len`: Número de bytes a leer.

  `data`: buffer de destino.

- Descripción

  Lee datos desde el Wiimote. Los indicadores `CWWID_RW_EEPROM` y `CWIID_RW_REG` seleccionan el espacio de direccionamiento de manera exclusiva entre ambos. El indicador `CWIID_RW_DECODE` realiza los algoritmos de decodificación necesarios para leer los datos desde el espacio de direccionamiento de los registros.

- Valor de retorno

  0 si ha resultado exitoso o un valor distinto si ha habido un error.

- `cwiid_write`

  - Sinopsis

    ```
    int cwiid_write(cwiid_wiimote_t *wiimote,
    unit8_t flags, uint32_t offset, uint16_t len,
    void *data);
    ```

  - Parámetros

    `wiimote`: `cwiid_wiimote` manejado.

    `flags`: Indicadores para la escritura.

    `offset`: dirección de comienzo.

    `len`: Número de bytes a leer.

    `data`: buffer de destino.

  - Descripción

    Escribe datos en el Wiimote. Los indicadores `CWWID_RW_EEPROM` Y `CWIID_RW_REG` seleccionan el espacio

de direccionamiento de manera exclusiva entre ambos. El indicador `CWIID_RW_DECODE` se ignora.

- Valor de retorno

  0 si ha resultado exitoso o un valor distinto si ha habido un error.

3.5. EJEMPLO DE PROGRAMACIÓN

En este apartado se muestra el uso de la API con un sencillo ejemplo que inicializa el Wiimote y muestra en pantalla el valor de los acelerómetros. El programa completo está en el Anexo B.

Primero se establece la comunicación con el Wiimote invocando al método `cwiid.Wiimote()`.

```
wiimote = cwiid.Wiimote()
```

En caso de error se lanzará la excepción RuntimeError. Una vez establecida la comunicación se enciende el led 1 del Wiimote. Se realiza esta operación para comprobar que existe conectividad.

```
wiimote.led = cwiid.LED1_ON
```

A continuación se activan los acelerómetros.

```
wiimote.rpt_mode = cwiid.RPT_ACC
```

A partir de este momento el Wiimote envía ininterrumpidamente el valor de los acelerómetros. Se tiene acceso a esta información de la siguiente manera:

```
acc = wiimote.state['acc']
```

Finalmente se imprime en pantalla su valor, disponibles a través de `acc[cwiid.X]`, `acc[cwiid.Y]` y `acc[cwiid.Z]`

3.6. APLICACIONES DESARROLLADAS

Se mostrarán a continuación dos aplicaciones de ejemplo que sirven para ilustrar el comportamiento del Wiimote como interfaz de control del ordenador.

Wiimote 2D

Wiimote2D se trata de una sencilla aplicación que traduce los movimientos del mando en un espacio en dos dimensiones. El programa da a elegir entre dos opciones:

- Mando en mano: Los movimientos detectados son las rotaciones sobre el eje X y las rotaciones sobre el eje Y, correspondientes ambos al movimiento de un punto en la dirección vertical y en la horizontal. Se podría hacer una correspondencia entre pitch y roll y movimiento vertical y horizontal respectivamente.

- Mando en gorra: Los movimientos que se detectan son los mismos que en la opción anterior. La diferencia es la correspondencia entre el pitch y roll siendo ahora con movimientos horizontales y verticales respectivamente. Se implementó está opción y se probó en una gorra por las posibilidades que puede ofrecer. Colocando el mando sobre la visera de la gorra o en algún otro soporte similar de manera que la parte más larga del mando se alinee con la frente del usuario se consigue el mismo efecto que el conseguido con la mano (véase Figura 3.10). Podría ser interesante a la hora de trasladar estos movimientos en los movimientos del cursor en la pantalla para personas con movilidad reducida o movilidad nula en las extremidades u otras aplicaciones que implicarán la intervención de estos usuarios.

La interfaz de la aplicación son dos ejes de coordenadas y un punto que se mueve según los ángulos anteriormente descritos. Se limita la representación a un intervalo de ángulos que son capaces de generar el movimiento de la mano o la cabeza respectivamente. Éstos son para la mano (-90º,90º) y para la cabeza (-22.5º,22-5º) pudiendo ser modificados por el usuario para lograr una mayor precisión en sus movimientos.

La aplicación comienza pidiendo la sincronización con el mando para lo cual se ha de pulsar 1+2. Tras finalizar con éxito, surge un menú en el que se puede elegir entre las dos opciones. Tras la elección surgirá la pantalla con los ejes de abcisas y ordenadas y la sensibilidad del mando ajustada a la opción elegida. Para finalizar la aplicación bastará con cerrar la ventana (véase Figura 3.9).

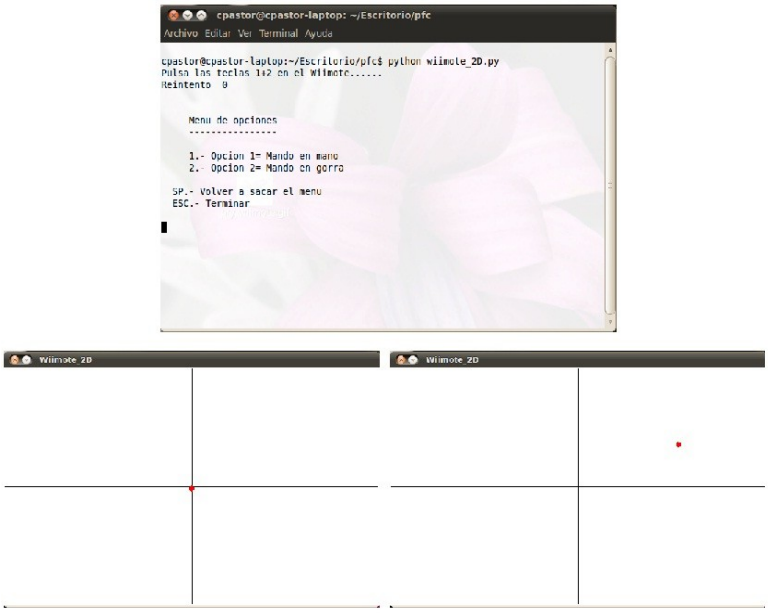

Figura 3.9. Vista del menú e imágenes de la aplicación Wiimote_2D

Figura 3.10. Ejecución de Wiimote_2D controlado mediante movimientos de la cabeza

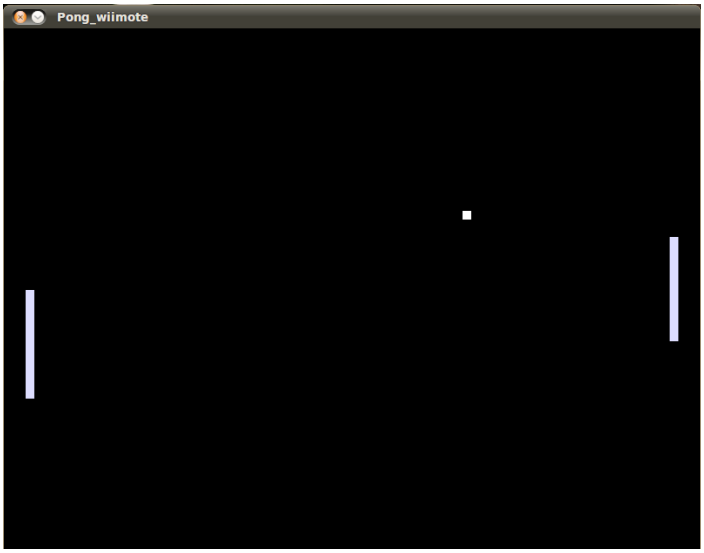

Figura 3.11. Vista de la aplicación Pong_wiimote

Pong_wiimote

Pong_wiimote es una aplicación que adapta el popular juego del pong [32] para que uno de los jugadores se controle mediante un Wiimote. Presenta una interfaz en la que se presenta el campo en negro y las palas con las que jugar y la bola en blanco. El juego consiste en pasar la bola y que toque el extremo contrario mientras que el jugador contrario intenta la misma acción en sentido contrario (véase Figura 3.10).

En esta versión falta la implementación de un "tanto" ya que nos centramos en la demostración de los módulos de *pygame* y de la librería que maneja el mando. Permite, sin embargo, el movimiento del jugador izquierdo mediante el Wiimote y el movimiento del jugador derecho mediante el pad direccional del ordenador. El movimiento para el segundo jugador es lineal mientras que en el del primero se ha modificado para que varíe con la velocidad. Se obtiene de esta manera un movimiento más suave y un control más preciso de la paleta. Para este movimiento vertical hacemos uso del pitch que se obtenga del movimiento del mando ya que se corresponde con la manera más natural de mover un objeto hacia arriba o abajo.

Pong_wiimote es una aplicación que combina varios elementos que definen perfectamente las aplicaciones objetivo en este proyecto: **uso del Wiimote como interfaz de control**, **entretenimiento** durante el uso y posible aplicación al campo de la **fisioterapia** o **rehabilitación**. Esto se debe a que además de usarlo con una mano se podría hacer una fácil adaptación a la cabeza o, incluso a la pierna, sin necesidad apenas de cambios en la implementación, solo en los parámetros que limiten el movimiento deseado.

3.7. RESUMEN

Se puede concluir que el Wiimote es un buen dispositivo para las aplicaciones a desarrollar por sus capacidades de detección de movimiento y captación de fuentes infrarrojas. Además, es ventajoso por su capacidad inalámbrica y por sus funciones adicionales que le pueden complementar en su uso. Nos encontramos ante un dispositivo que engloba varias tecnologías que nos permiten desarrollar aplicaciones adaptables a nuestras necesidades y sin un alto coste.

Capítulo 4

Wii Remote+IR LEDs

4.1. INTRODUCCIÓN

Este capítulo describe el sistema que utiliza Wii para su función de puntero basado en detección de fuentes infrarrojas. Se presenta la parte del API que se ha de utilizar para programar aplicaciones haciendo uso de esta característica del mando. Por último se ilustra con dos aplicaciones, el uso que se le puede dar a esta funcionalidad del Wiimote.

El sistema que utiliza Wii se puede presentar como el único que, hoy en día, es capaz de apuntar con precisión en una pantalla de cualquier tipo [29]. La solución que implementa Nintendo para la función de puntero en la pantalla está basada en un sensor óptico. Se descarta así la idea de la implementación de una lightgun (clásicas pistolas para videojuegos) en miniatura ya que éstas presentan dos graves problemas: funcionan solamente con televisores de tubo de rayos catódicos, en los cuales basan su tecnología, y una baja precisión, ya que pintaban grandes rectángulos en las zonas donde el disparo se consideraba acertado.

4.2. DESCRIPCIÓN TÉCNICA

En el interior del mando se encuentra el sensor MOT (Multi-Object Tracking) de Pixart Imaging (véase Figura 4.1). Se trata de una cámara monocromática de resolución 128x96 con un filtro de paso de infrarrojo. El procesador integrado utiliza un análisis subpíxel 8x para proporcionar una

resolución de 1024x768 para el seguimiento. La cámara es capaz de seguir hasta cuatro objetos en movimiento [33].

La cámara de infrarrojos tiene un campo de visión efectivo de 33 grados en el plano horizontal y de 23 grados en el plano vertical (véase Figura 4.2). Con el filtro IR intacto, las fuente de 940 nm de longitud de onda se detectan con el doble de intensidad que una fuente equivalente de 850 nm pero no se resuelve tan bien en distancias cercanas. La longitud focal de la cámara es de 1320 [34]. La distancia mínima a la que se detecta una fuente de luz de manera nítida y constante es de 8.5 cm y la distancia máxima depende de la potencia que tenga dicha fuente.

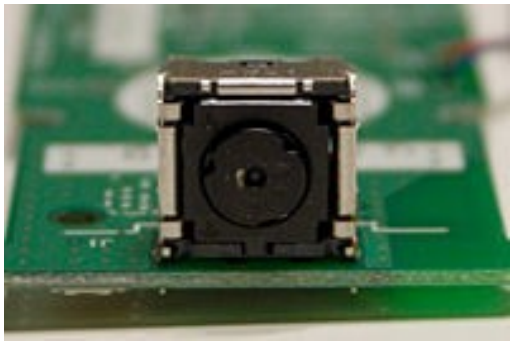

Figura 4.1. Vista frontal del sensor de movimiento

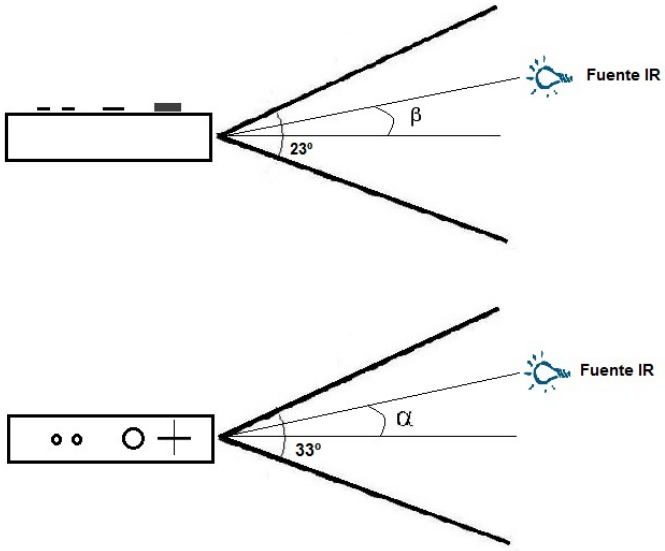

Figura 4.2. Esquema de los ángulos de visión máximos de la cámara del Wiimote

El otro elemento que permite el funcionamiento de la función de puntero es la barra sensora (véase Figura 4.3). Es un elemento que contiene en cada extremo cinco LEDs infrarrojos y cuyas medidas son 24x1 cm. La distribución dentro de cada grupo de LEDs es así:

- El LED situado en el extremo de la barra apunta ligeramente hacia el exterior de la barra.
- El LED situado en la posición más cercana al medio de la barra apunta ligeramente hacia el interior de la barra.
- Los tres LEDs restantes apuntan de manera perpendicular y se encuentran agrupados.

La barra tiene un cable de 3 m de largo que se conecta a un puerto propietario de la consola, encargado de alimentar y poner en funcionamiento las luces de los extremos.

4.3. MODOS DE FUNCIONAMIENTO

4.3.1. FUNCIONAMIENTO ESTÁNDAR

El denominado funcionamiento estándar de este sistema es el utilizado por Nintendo. Esta característica del mando se usa para implementar una función de puntero en la pantalla que se utiliza en juegos y en los menús de estos.

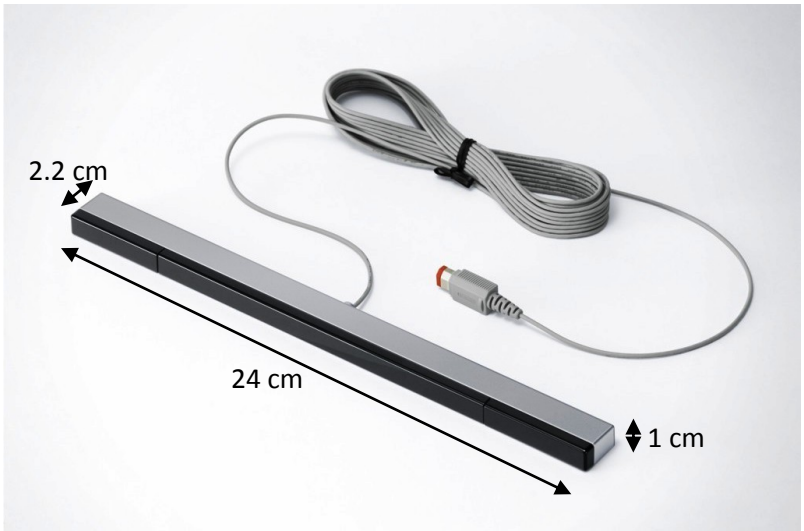

Figura 4.3. Barra de sensores

El sensor óptico del Wiimote localiza los LEDs conformando los extremos de una “pantalla virtual”, la cual es un campo de acción relativo generado muy cerca de la pantalla real del televisor o proyector. Debido a este fenómeno, cuando se configura la consola por primera vez, se ha de indicar la posición de la barra con respecto a la zona de visualización para que ese campo se considere aproximadamente donde se encuentra la pantalla real. Si la barra está colocada en la parte superior del televisor, el sensor debe estar alineado con la parte delantera de la televisión y si se coloca en la parte inferior se ha de alinear con la parte delantera de la superficie de ésta última (vese Figura 4.4).

A partir de la pantalla virtual generada se envían al sistema datos de movimiento del puntero en un espacio similar al que ocupa la pantalla real simulando que estamos apuntando directamente hacia ella. Según sea la relación de aspecto del televisor, el campo de acción será ensanchado si nos encontramos con una relación 16:9 o, por el contrario, permanecerá inalterado si la relación es 4:3. La distancia del mando respecto a la barra se calcula basándose en la recepción por parte del mando de la luz emitida por la barra, lo que permite el uso coherente independientemente del tipo o tamaño del televisor. La luz que se emite desde cada extremo de la barra se centra en el sensor de imagen como dos puntos separados por una distancia “di”. La segunda distancia “d” entre los dos grupos de emisores de luz es una distancia fija (véase Figura 4.5). A partir de estas dos distancias el procesador de la consola calcula la distancia entre el Wiimote y la barra de sensores utilizando triangulación. Además, la rotación del mando con respecto al suelo se calcula a partir del ángulo relativo de los dos puntos de luz en el sensor de imagen. Con estos datos (posición de la barra, relación de aspecto de la pantalla y distancia del mando respecto a la barra) y basándose en la sensibilidad, el puntero se puede mover con precisión dentro de la pantalla.

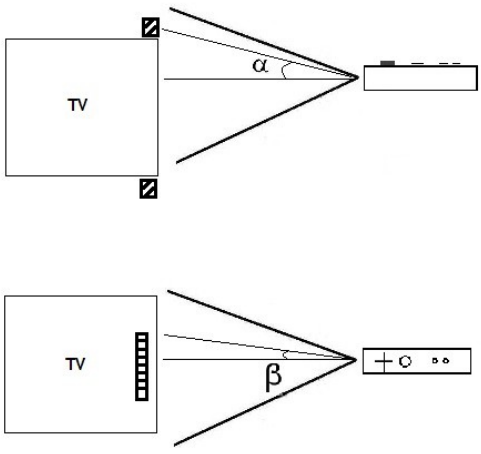

Figura 4.4. Vista lateral y superior de posible colocación de barra de sensores y su relación con el mando

Se plantea la opción del apuntado directo que fue la que dio vida al periférico Wii Zapper (apartado 3.3.2.) con el que cualquiera intentará apuntar a través de la mirilla. Esta posibilidad se mantiene al margen por la complejidad que puede suponer la implementación ya que se debería conocer a la perfección el tamaño de la pantalla y mediante una calibración al inicio el movimiento podría corresponderse con ella. Pero esta solución plantea el problema de que la distancia entre mando y barra puede variar durante el juego y, por tanto, la perspectiva inicial.

Otro añadido al sistema es una pequeña amortiguación del movimiento del puntero realizada por software y que depende de la aplicación a la que se esté dedicando el puntero y de la configuración. Se incluye ya que no se podría aceptar una total precisión a la hora de acertar en el apuntado a un objeto de pequeñas dimensiones.

Además se incluye una compensación al posible error producido por la falta de lectura de las coordenadas de los LEDs. Si se gira tanto el mando que éste pierde de vista los LEDs de la barra los sensores de aceleración y giro miden la aceleración lineal y los rangos de rotación de manera que se ofrecen valores aproximados de la orientación y posición del Wiimote. Estos valores se van descompensando a medida que el mando no ve nuevamente los emisores de luz infrarroja. Una vez entre en su campo de visión, los datos de posición y rotación del sensor óptico recalibran automáticamente la posición y orientación del mando.

La distancia máxima de uso de esta función es de cinco metros, ya que a partir de ahí, el mando no puede distinguir la posición de los LEDs. La distancia que se ha de guardar a la hora de jugar será la citada como máximo ya que todos los sistemas de menús de la consola como de los juegos se aprovechan de ella.

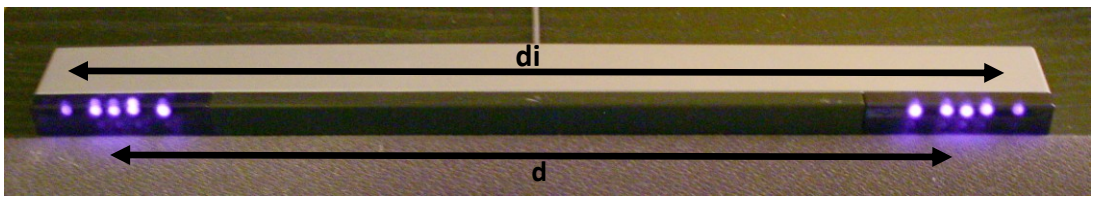

Figura 4.5. Vista de los LEDs de la barra sensora mediante una cámara convencional

Un problema que puede surgir en el sensor infrarrojo a la hora de detectar posición es la existencia de otras fuentes de infrarrojos alrededor, tales como velas o bombillas incandescentes. Una solución para mitigar este efecto es el uso de luces fluorescentes alrededor de la Wii ya que emiten poca o ninguna luz infrarroja. A raíz de este problema, usuarios innovadores han utilizado otras fuentes de luz IR como sustitutos de la barra de sensores usando un par de linternas y un par de velas.

4.3.2. FUNCIONAMIENTO NO ESTÁNDAR

El funcionamiento no estándar se corresponde con el uso que se puede dar a la cámara incorporada en el mando sin la barra de sensores y la consola. De esta manera se pueden desarrollar aplicaciones independientes de Nintendo con el uso exclusivo del mando. Existen tres aplicaciones desarrolladas por Johny Chung Lee [35], el primero en utilizar estas herramientas y darles difusión.

- **Seguimiento de los dedos con el Wiimote**

  El fundamento de esta aplicación es el mismo que para la detección de una o varias fuentes de infrarrojo (véase Figura 4.2), pero usando un array de LEDs.

  El Wiimote se coloca sobre un soporte apuntando hacia el usuario acoplando el array de LEDs alrededor de la cámara. En los dedos se colocan unos trozos de material reflectante de un tamaño equivalente a la yema de éstos. Una vez iniciada la aplicación, establecida la conexión con el mando y encendido el array, podremos mover los dedos visualizando en pantalla los puntos que se corresponden con ellos. El funcionamiento varía en cuanto a que la posición que se calcula es la del material reflectante y no del led. Es posible hacer seguimiento ya que lo que se refleja se filtra en el mando y la señal es similar a la que generaría un LED IR directamente.

- **Pizarras interactivas multi-touch de bajo coste**

  Esta aplicación consiste en una pizarra en la que se pueda interactuar con hasta cuatro lápices emisores de IR. El esquema que se ha de seguir para poder ejecutar la aplicación satisfactoriamente es el siguiente:

  El mando ha de estar apuntando a la pantalla teniendo en cuenta el ángulo de visión de éste y la distancia a la pantalla (véase Figura 4.6). Una vez iniciada la aplicación, y establecida la conexión con el mando, se procede a una calibración situando puntos en las cuatro esquinas de la pantalla. Si la calibración es satisfactoria se podrá usar un lápiz emisor de infrarrojo como ratón del ordenador. Para comprobar las capacidades

multi-touch se ha de instalar un programa desarrollado por el autor y utilizar más de un lápiz IR.

Esta aplicación se puede utilizar también sobre el material que se utiliza en proyecciones. Proyectando la imagen del ordenador en una pared o en una tela se puede proceder de la misma manera que con una pantalla y conseguir idéntico resultado.

- **Seguimiento de la cabeza en entornos de realidad virtual**

Esta aplicación nos sirve para hacer un seguimiento de la cabeza en un entorno de realidad virtual. El mando se ha de colocar con ángulo suficiente para que sea capaz de captar los movimientos de la cabeza de una persona dentro de su campo de visión. La persona se ha colocar unas gafas que tienen en cada extremo un par de LEDs emisores de infrarrojos u otro objeto similar sujeto a la cabeza y con los diodos instalados.

El programa funciona de tal manera que la imagen que se ve en pantalla se adapta a la posición del usuario. De esta manera se consigue un efecto de realidad virtual ya que según se acerque o aleje el usuario, la imagen se adaptará para que parezca que se acerque o se aleje a ella. También se mostrarán las partes laterales de una imagen que quedarían ocultas si de una imagen estática se tratase. Se consigue un efecto parecido al de asomarse a una ventana, siendo su marco el marco de la pantalla de visualización.

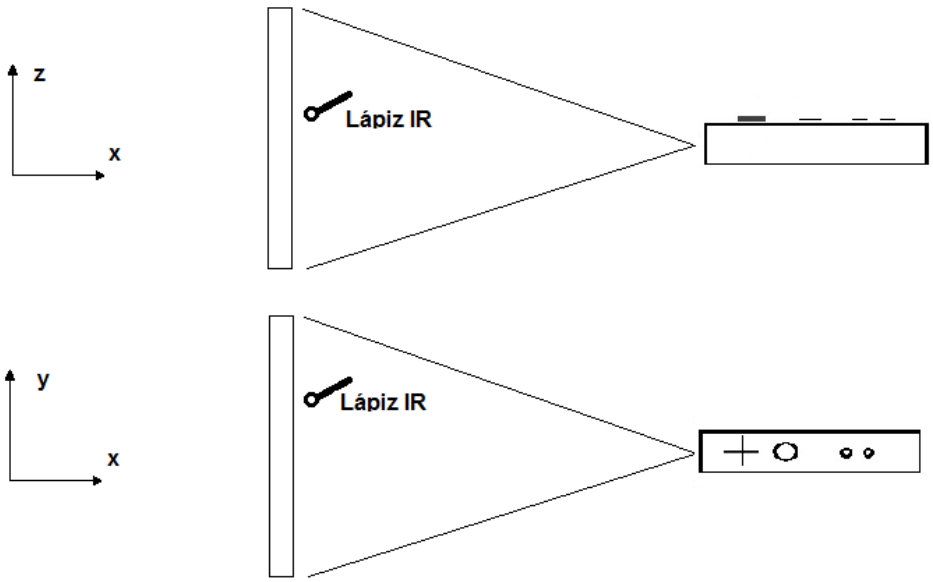

Figura 4.6. Vista superior y lateral de la interacción entre un lápiz emisor de IR y el Wiimote

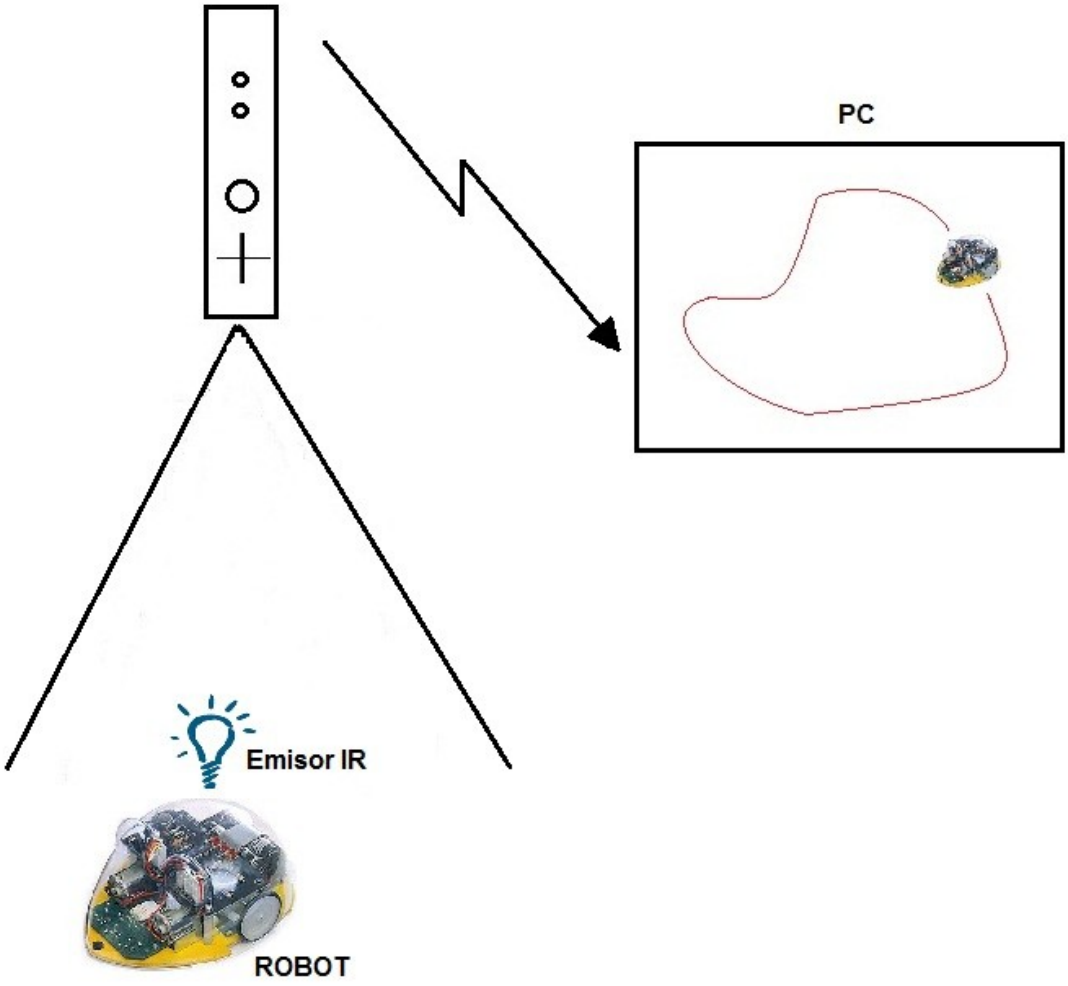

Figura 4.7. Vista esquemática del seguimiento de un robot móvil

Otro posible uso de esta tecnología sería la del **seguimiento de objetos**. El objeto ha de tener incorporado un emisor de infrarrojos y se ha de colocar el mando en unas posición que abarque todo el posible campo de movimiento del objeto. Mediante este montaje y un software podemos seguir la ruta que sigue el objeto móvil en la pantalla del ordenador. Por ejemplo, podemos seguir el recorrido de un robot móvil al que incorporaremos un LED IR en su parte superior.

4.4. CIRCUITO EMISOR DE INFRARROJOS

El circuito que se ha de usar para el montaje de un emisor de infrarrojos se muestra en la Figura 4.8.

Consta de una pila para alimentar el LED, una resistencia para regular la corriente, el LED emisor y un interruptor de encendido/apagado. Los valores de la pila y la resistencia vienen dados por el valor de intensidad o voltaje que soporte el LED IR.

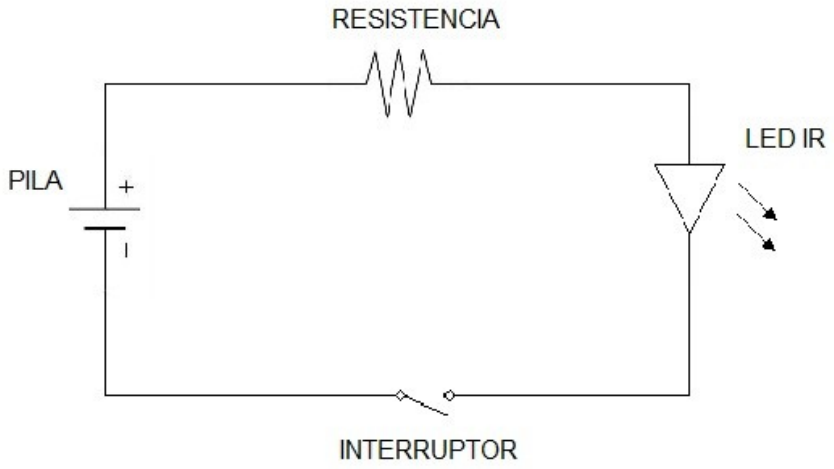

Figura 4.8. Esquema eléctrico de un emisor infrarrojo

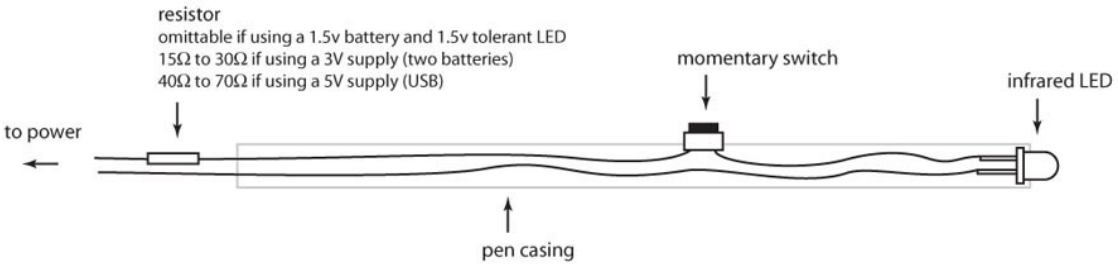

Figura 4.9. Esquema de montaje de lápiz infrarrojo

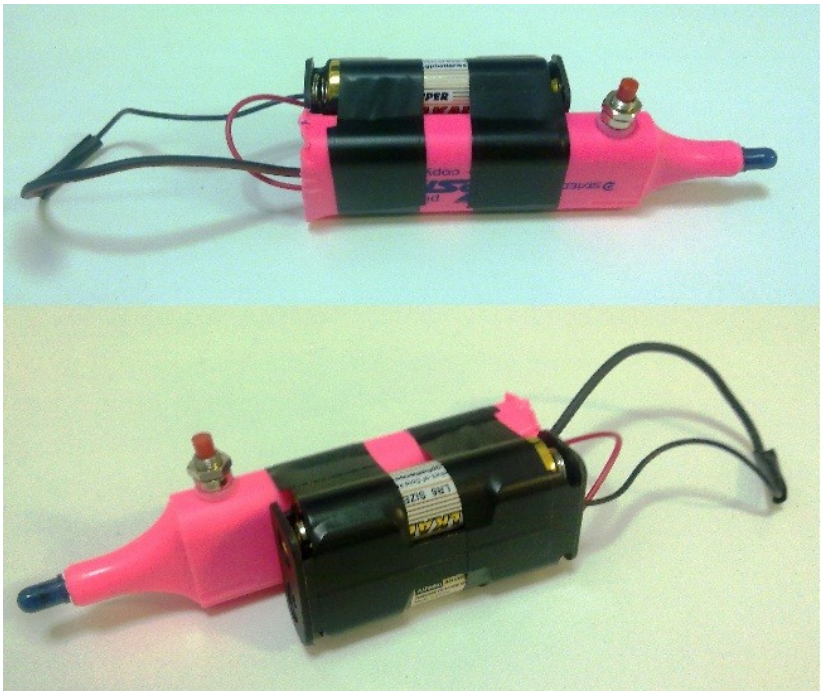

Figura 4.10. Vistas laterales del montaje utilizado en las pruebas de aplicaciones

Para realizar las pruebas se ha montado un emisor siguiendo el esquema indicado en la Figura 4.9 [35].

En este caso se utiliza como soporte un rotulador. Acoplado a éste hay un portapilas con dos pilas de 1.5V. En el extremo que correspondería a la tinta se encuentra el LED emisor y en un lateral el interruptor que utilizaremos para encender y apagar la emisión (véase Figura 4.10). En el interior se encuentra una resistencia de 23Ω, además del cableado de conexión necesario.

4.5. API PARA LA PROGRAMACIÓN

El API que se utilizará para la programación de aplicaciones será el mismo que se utiliza para el Wiimote(véase apartado 3.4). Para hacer uso de la funcionalidad de detección de LEDs infrarrojos se ha de activar en el mando el envío de informes acerca de esa característica. Posteriormente se ha de detectar si existe alguna fuente de luz IR y tratarla en la aplicación como se desee.

4.6. EJEMPLO DE PROGRAMACIÓN

En este apartado se muestra el uso de la API con un ejemplo que monitoriza las coordenadas de una fuente de luz infrarroja captada por el Wiimote. El programa completo está en el Anexo B.

En primer lugar, establecemos la comunicación con el Wiimote invocando al método `cwiid.Wiimote()`.

```
wiimote = cwiid.Wiimote()
```

Si resulta errónea la operación, se lanzará la excepción RuntimeError. Si la conexión ha sido exitosa se puede activar un led del Wiimote para tener confirmación visual.

```
wiimote.led = cwiid.LED1_ON
```

A continuación se activa la función de captación de fuentes infrarrojas.

```
wiimote.rpt_mode = cwiid.RPT_IR
```

El Wiimote mandará de manera continua el valor de la posición de la fuente o fuentes de luz infrarroja que capte su cámara. Se comprueba que existen fuentes infrarrojas en el estado del Wiimote mediante:

```
src = wiimote.state['ir_src']
```

Se imprimirá por pantalla la posición de la fuente de luz accediendo a su valor a través de `src['pos']`.

4.7. APLICACIONES DESARROLLADAS

Se presentarán dos aplicaciones desarrolladas que ilustran el comportamiento del Wiimote con un LED infrarrojo:

Wii_led

Wii_led hace un seguimiento de las luces infrarrojas detectadas. Representa la posición que ocupan respecto a la cámara mediante un punto rojo dentro de un plano de coordenadas cartesianas (véase Figura 4.10). Nos puede servir para ajustar los límites de visión del mando de manera empírica si manejamos una fuente de luz infrarroja en nuestra mano y calibrar para futuras aplicaciones.

Antes de ejecutar la aplicación se ha de colocar el mando en la posición que deseemos. Una vez arrancada la aplicación y establecida la conexión entre mando y ordenador nos aparecerá la interfaz antes citada. Si posteriormente ejecutaremos otra aplicación en la que se haga un seguimiento, podemos

comprobar con esta si todo el movimiento está abarcado en el ángulo de visión. Si esto no es así, se ha de mover el mando hasta una posición donde se cumpla la condición de visión.

Wii_mancuerna

Wii_macuerna es una aplicación para la monitorización del levantamiento de una mancuerna. El ejercicio al que podría acompañar sería el consistente en levantar una mancuerna u otro objeto similar desde una superficie estable. La aplicación nos muestra el ángulo que mantiene el brazo respecto a la superficie y una representación gráfica simplificada (véase Figura 4.13).

Para su ejecución se han de seguir estas indicaciones:

- Colocar el mando de manera adecuada para que el movimiento del objeto sujetado en la mano esté dentro de su campo de visión. Para esta calibración se puede usar la aplicación desarrollada Wii_led.
- Montar en la mancuerna u objeto a sujetar un LED infrarrojo para que se pueda realizar el seguimiento de su movimiento. El montaje se ha de realizar según el esquema visto en la sección cuatro del capítulo adaptándolo a la forma del objeto en cuestión.
- Calibrar la aplicación cada vez que se ejecute para que el seguimiento sea lo más preciso posible. La calibración consiste en la colocación del brazo en la posición inicial y la emisión de luz IR para tomar la referencia inicial del movimiento.

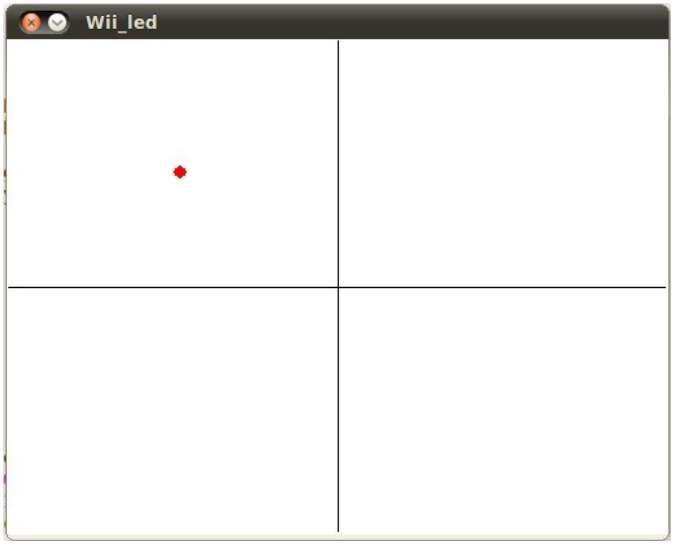

Figura 4.11. Vista de la aplicación Wii_led

La aplicación se ha desarrollado y probado en un entorno tal y como describe la Figura 4.11. La superficie sobre la que se trabaja es una mesa sobre la cual colocamos unas cajas que nos servirán de soporte al mando en uno de los extremos. El brazo lo hemos de colocar apoyado lo más cerca del borde de la mesa a la distancia indicada. En la mano sujetaremos el soporte construido (ver apartado 4.4) para la localización de la fuente de infrarrojo.

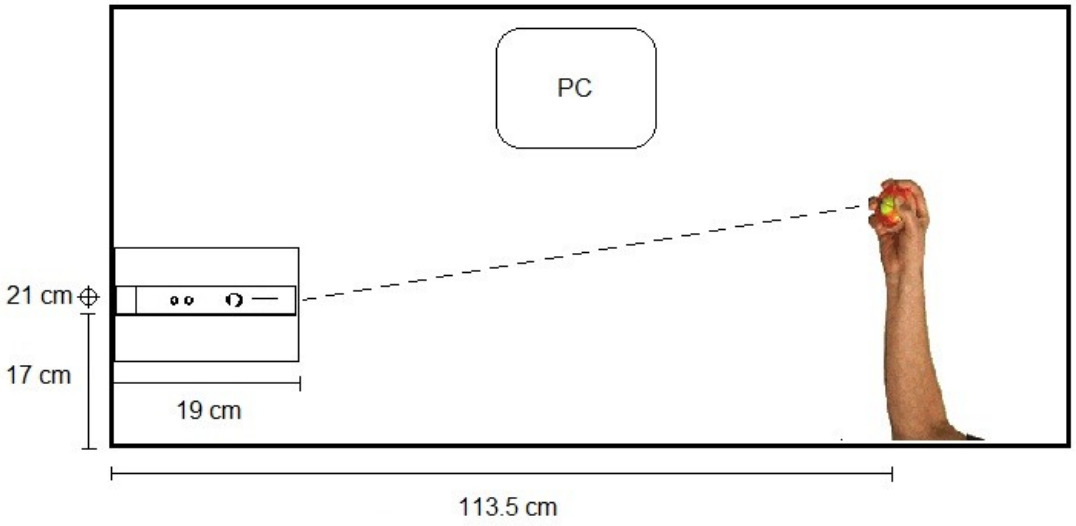

Figura 4.12. Esquema de entorno de prueba de la aplicación Wii_mancuerna

Figura 4.13. Vista del entorno de prueba de la aplicación Wii_mancuerna

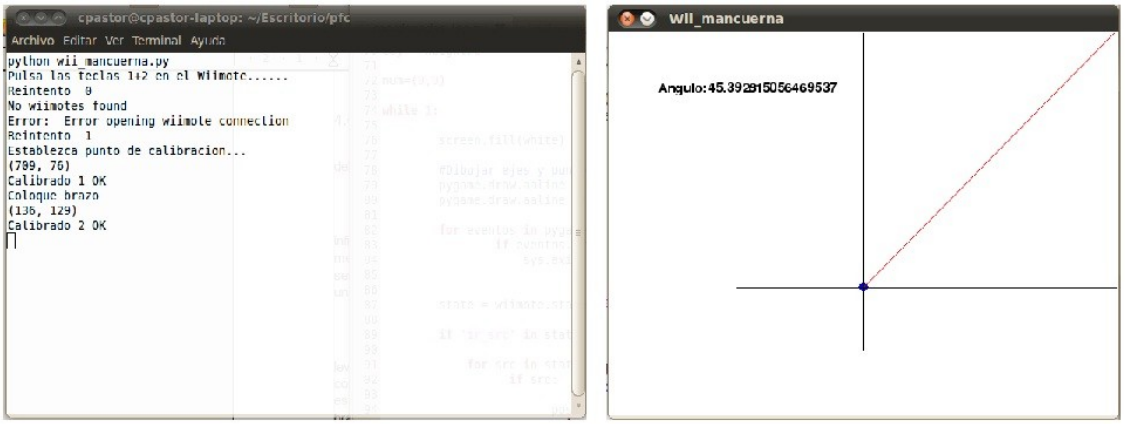

Figura 4.14. Vista de la aplicación Wii_mancuerna

4.8. RESUMEN

La consola Wii y el sistema de apuntado de su mando nos ofrecen una buena solución para movernos en una pantalla basándonos en la posición de luz ofrecida por los LEDs. La forma de apuntar no es la misma que utilizaríamos con un puntero láser o con un ratón, ya este sistema junto con el acelerómetro ofrece mayores posibilidades. Esto combinado con el bajo coste de adquisición de mandos y LEDs permite ofrecer soluciones económicas y fáciles de usar para cualquiera.

Se pueden diseñar aplicaciones basándonos en esta funcionalidad en las que se interactúe con el mando teniendo unas luces fijas o con algún dispositivo en el que integremos diodos luminosos. Esta última opción es muy interesante para pacientes de rehabilitación con limitación en los movimientos de las extremidades o pérdida de éstos ya que **se podrían adaptar interfaces específicas a su limitación**.

Capítulo 5

Wii Remote+Wii Motion Plus

5.1. INTRODUCCIÓN

En este capítulo se describe el Wii Motion Plus. Se detallan sus características técnicas y el API que se usará en el desarrollo de aplicaciones usando el Wiimote con este accesorio. Se incluye además un ejemplo de aplicación que hace uso de la funcionalidad añadida por este elemento.

El objetivo que Nintendo persiguió con el desarrollo de este producto fue traspasar las fronteras actuales y lograr un mayor realismo en el control basado en movimientos. Las sensaciones que transmite el control mejorado son gratificantes y se percibe una diferencia notable entre el uso con el accesorio y el uso sin él. Se puede decir que la sensibilidad en los movimientos con el anterior mando de Wii estaba limitada ya que existe una mejora en la interactividad gracias a una gran precisión y realismo. El juego se convierte en más participativo y accesible debido a las demandas de los jugadores más exigentes que pedían una extensión de este tipo o una mejora en el mando [36].

5.2. DESCRIPCIÓN TÉCNICA

El Wii Motion Plus es uno de los periféricos que se pueden acoplar al Wiimote (véase Figura 5.1). Se trata de un nuevo sistema que mejora la precisión y que puede reflejar en la pantalla hasta los movimientos más pequeños realizados por el jugador complementando al acelerómetro incluido en el mando y a la barra de sensores (si su se usa junto con la consola).

Figura 5.1 Wiimote con Wii Motion Plus

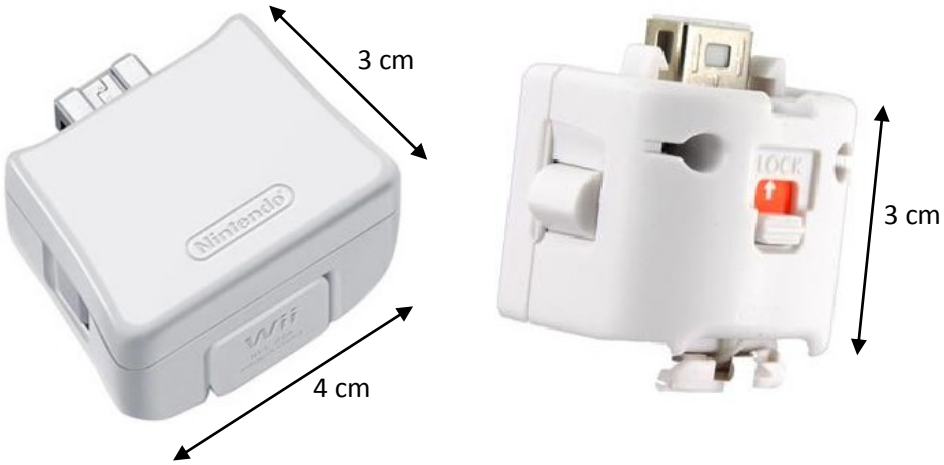

Figura 5.2. Vista frontal y trasera del Wii Motion Plus

El Wii Motion Plus es un pequeño dispositivo rectangular de 3x4x3 cm sin botones. Se conecta por el puerto de expansión que está situado en la parte inferior del Wiimote. Permite la conexión asimismo de otro accesorio de los antes citados como, por ejemplo, el Nunchuk. La unión entre el mando original y el accesorio no es rígida en su totalidad pero cuenta con un sistema de bloqueo para que éste tenga más seguridad en su inserción o extracción (véase Figura 5.2).

El fundamento físico del Wii Motion Plus es el funcionamiento de un giroscopio. Este giroscopio está implementado mediante dos sensores giroscópicos:

- Un giroscopio de dos ejes IDG-600 de InvenSense. Es el encargado de la lectura del pitch y del roll.
- Un giroscopio de un eje X3500W de EPSON TOYOCOM. Se encarga de la lectura del yaw.

El uso combinado de estos dos giroscopios es el que permite el envío por parte del Motion Plus de la velocidad angular que se experimenta en los tres ejes. El pitch se corresponde con los giros experimentados sobre el eje X (cabeceo), el roll se corresponde con los giros experimentados sobre el eje Y (alabeo) y el yaw mide los giros realizados sobre el eje Z (guiñada) (véase Figura 5.3).

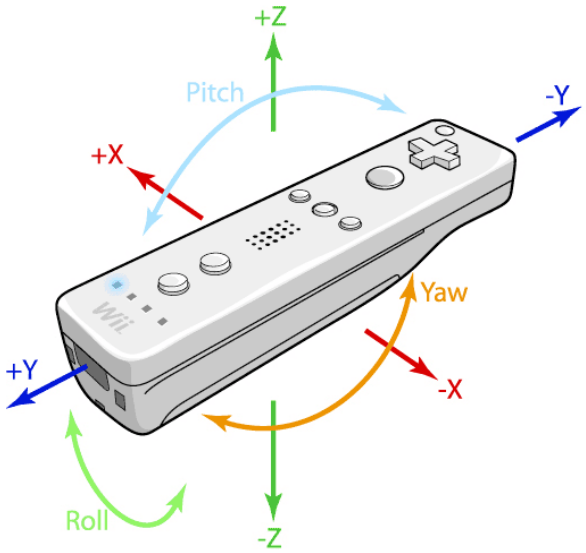

Figura 5.3. Orientación del acelerómetro del Wiimote y de las magnitudes medidas por el Wii Motion Plus

El sensor detecta los movimientos de giro y rotación debido a que capta 1600 grados por segundo, lo que equivaldría a cuatro giros y medio de la mano. Esto supone aumentar la sensibilidad del sensor multiplicando por cinco la sensibilidad de sensores giroscópicos habituales en ese tamaño. De esta manera se garantiza que se captan los movimientos más rápidos realizados por los jugadores.

El sensor giroscópico del mando está diseñado para tener dos modos de funcionamiento: uno para movimientos lentos y otro para movimientos rápidos. De esta manera no perdería sensibilidad a la hora de detectar los movimientos lentos pese a estar diseñado para detectar los más rápidos. Para lograr esto, teniendo en cuenta que los datos se envían de forma inalámbrica y su resolución está predeterminada, se miden las velocidades en distintas escalas. Para los movimientos lentos se envían datos midiendo hasta una determinada resolución en el número fijo de bandas de transmisión y para los movimientos rápidos se envían los datos midiendo al doble de resolución en el mismo número de bandas [37].

Los giroscopios antes citados se tratan de dos chips que utilizan la tecnología MEMS. La mayoría de estos giroscopios se basan en el efecto Coriolis para su funcionamiento. Para la comprensión de este efecto supondremos el siguiente ejemplo: un niño se encuentra en el centro de un tiovivo y quiere andar hacia su madre que se encuentra fuera del tiovivo. El niño empieza a ir hacia ella pero para poder avanzar en línea recta además de seguir su movimiento hacia adelante deberá mantener otro también de forma lateral para contrarrestar la rotación del tiovivo. El efecto Coriolis consiste en que la velocidad lateral que deberá mantener será mayor cuanto más cerca del borde se encuentre. El aumento de velocidad produce una aceleración y el decremento una deceleración.

Un giroscopio electrónico está formado por dos partes: un elemento de polisilicato resonante que realiza un movimiento de vibración hacia fuera o hacia dentro respecto al eje de rotación del giroscopio y otra parte que se encuentra fija y perpendicular al movimiento vibratorio de la primera. Estas dos partes juntas forman una estructura capacitiva que es capaz de contener una carga eléctrica. Cuando el giroscopio no está girando la distancia entre todos los elementos se mantiene y por tanto la capacitancia se mantiene constante. Si el giroscopio está girando se produce el siguiente efecto: el elemento resonante se encuentra desplazándose hacia dentro o fuera del eje de rotación debido a su movimiento de vibración, experimentando una aceleración o una deceleración producida por el efecto Coriolis. Estas aceleraciones se traducirán en fuerzas en sentidos contrarios que afectarán a la masa resonante. Debido al movimiento experimentado por la masa, la capacitancia varía de forma proporcional a la velocidad de rotación de ésta. Estos cambios de capacitancia se detectan por sensores que determinan la velocidad de rotación del giroscopio y la expresan en un voltaje de salida [38].

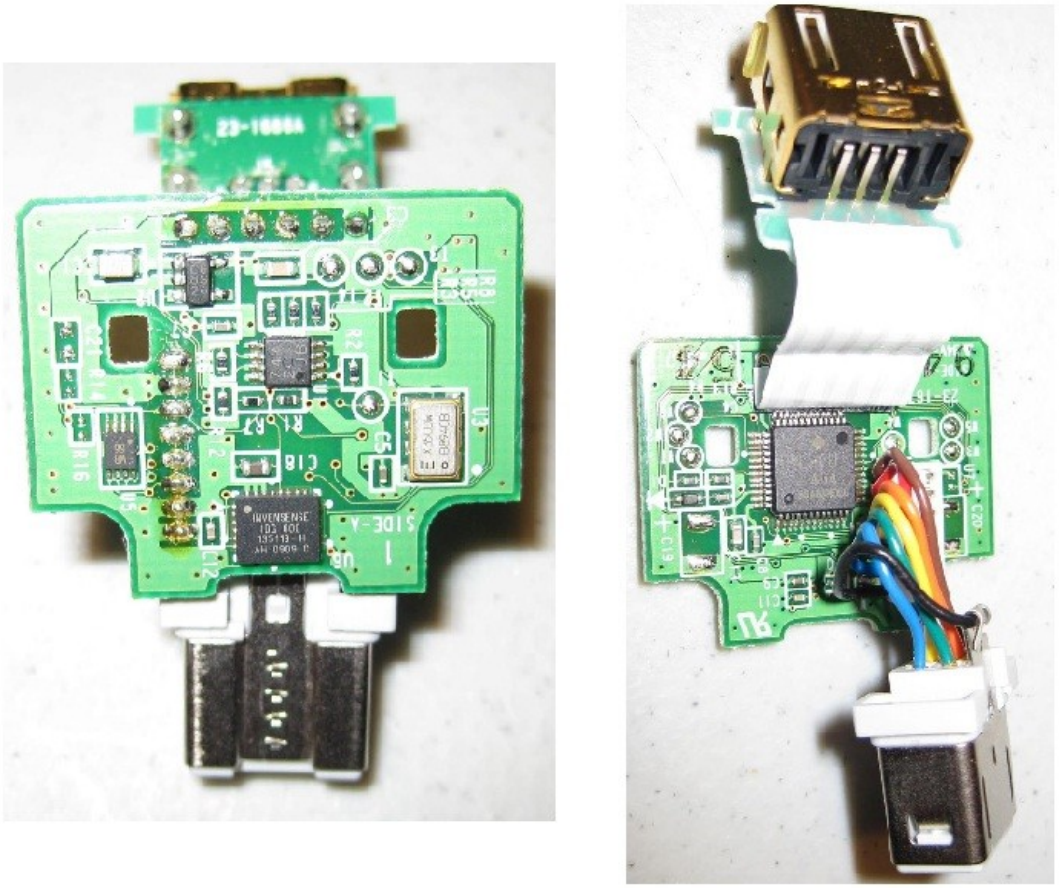

Figura 5.4. Placa del Wii Motion Plus

El Wiimote con Wii Motion Plus consigue un seguimiento del movimiento 1:1 ya que se alcanzan seis grados de libertad. Se ha de tener en cuenta la limitación representada por la conversión de aceleraciones y velocidades en posiciones mediante integración. Además, en la entrega de la lectura del yaw, los giroscopios permiten al software diferenciar entre orientación y aceleración lo cual dificulta la lectura diferenciada respecto al acelerómetro que entrega una mezcla de ambas.

5.3. API PARA LA PROGRAMACIÓN

El API que se utilizará para la programación de aplicaciones es el mismo que se utiliza para el Wiimote (véase apartado 3.4). Para utilizar el accesorio se ha de habilitar el flag indicado a tal efecto (`FLAG_MOTIONPLUS`). Posteriormente se ha de activar el modo de reporte de informes de Wii Motion Plus. Para obtener los valores de velocidad angular en los tres ejes, se ha de acceder al

estado del Wiimote y extraerlos mediantes las claves `'motionplus'` y `'angle_rate'`.

5.4. EJEMPLO DE PROGRAMACIÓN

En este apartado mostramos el uso del API de programación con un ejemplo que muestra los valores de las velocidades angulares experimentadas en los tres ejes del accesorio. El código completo está en el Anexo B.

En primer lugar se establece la comunicación con el Wiimote mediante el método que nos devuelve el objeto Wiimote a manejar.

```
wiimote = cwiid.Wiimote()
```

En el caso de que esta operación resulte errónea, se lanzará la excepción RuntimeError. Confirmamos visualmente que la conexión se ha establecido correctamente encendiendo uno de los LEDs del mando.

```
wiimote.led = cwiid.LED1_ON
```

A continuación habilitaremos la opción correspondiente a la conexión del Wii Motion Plus al mando.

```
wiimote.enable(cwiid.FLAG_MOTIONPLUS)
```

Posteriormente se activa la función de recepción de mensajes del Wii Motion Plus.

```
wiimote.rpt_mode = cwiid.RPT_MOTIONPLUS
```

El Wiimote mandará continuamente los valores de velocidad angular a la aplicación. Para el posterior tratamiento se comprueba si existen valores correspondientes al Wii Motion Plus accediendo al estado.

```
if  wiimote.state.has_key('motionplus')
```

Por último se imprimen los valores de velocidad angular en los tres ejes accediendo a su valor mediante

```
vel_ang = wiimote.state['motionplus']['angle_rate']
```

5.5. APLICACIÓN DESARROLLADA

Wii_motionPlus

Wii_motionPlus es una aplicación que nos muestra visualmente los valores que nos ofrece el Wii Motion Plus. La aplicación presenta una interfaz en la que se observan tres ejes correspondientes a las tres componentes de velocidad angular detectada por el accesorio. Para cada una de estas componentes, y conforme el movimiento del mando con el accesorio incorporado, se nos mostrará un punto coloreado que varía en función de la velocidad experimentada (véase Figura 5.5).

Para reflejar los valores de velocidad angular se han tomado como referencia los valores que nos devuelve el Wii Motion Plus cuando se sitúa horizontalmente sobre una superficie estable. Los desplazamientos de los puntos respecto a la referencia se muestran de manera que se corresponda con el movimiento de la mano siendo:

- Pitch. El desplazamiento a la derecha se corresponde con un movimiento en sentido horario en el plano YZ y el desplazamiento a la izquierda con un movimiento en sentido antihorario en el mismo plano.
- Roll. El desplazamiento a la derecha se corresponde con un movimiento en sentido horario en el plano XZ y el desplazamiento a la izquierda con un movimiento en sentido antihorario en el mismo plano.
- Yaw. El desplazamiento a la derecha se corresponde con un movimiento en sentido horario en el plano XY y el desplazamiento a la izquierda con un movimiento en sentido antihorario en el mismo plano.

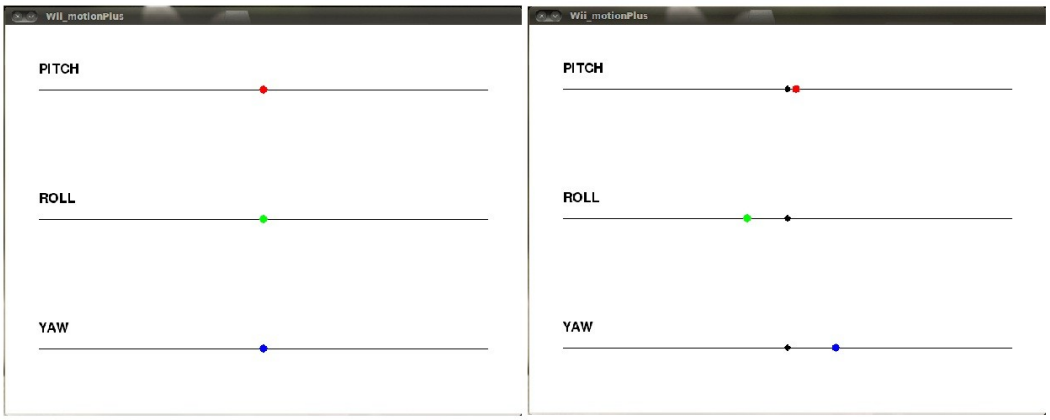

Figura 5.5. Vista de la aplicación Wii_motionPlus con el mando en reposo y con el mando en movimiento

5.6. RESUMEN

El Wii Motion Plus es un accesorio que ofrece unas características que complementan al Wiimote en cuanto a detección de movimiento se refiere. Se puede conseguir el seguimiento natural de los movimientos ya que con este accesorio se consiguen unos datos acerca del movimiento lineal y angular.

Para las aplicaciones objetivo se trata de un accesorio que mejora la detección de movimiento ofrecida por el mando actual pudiendo tener más precisión en éstos. Además se ofrecen los valores de pitch, roll y yaw los cuales no se podían ofrecer con el mando sin accesorio si no existía una fuerza de gravedad que actuara sobre él.

Capítulo 6

Wii Balance Board

6.1. INTRODUCCIÓN

En este capítulo se describe la Wii Balance Board. La Wii Balance Board (WiiBoard) es un accesorio para la consola Wii de Nintendo que consiste en una tabla capaz de calcular la presión ejercida sobre ella. Se presenta el API para desarrollar aplicaciones con ella y dos programas de ejemplo.

El objetivo que se persiguió con la Wii Balance Board fue el de conseguir un juego que permitiera a los usuarios pesarse a diario. Se comenzó pensando en construir algo similar a una báscula pero sus componentes encarecerían el producto final que está compuesto de software y hardware. De la primera idea basada en dos básculas capaces de enviar datos al ordenador surgió la idea de medición del equilibrio. Posteriormente se modificó el diseño para que enviara 60 señales por segundo ya que la medición con las básculas tradicionales haría que la respuesta se ralentizara y fuera imposible desarrollar un título basado en las variaciones del equilibrio.

El primer prototipo era circular y contaba en su interior con dos codificadores giratorios. Éstos transformaban la presión ejercida sobre ellos en rotaciones calculadas por los sensores ópticos y, a partir de estas rotaciones, poder calcular peso y equilibrio a ambos lados. Para poder medir mejo el equilibrio surgieron otros dos prototipos que fueron descartados por su complejo diseño, llegando en el proceso de desarrollo a la forma actual [39].

6.2. DESCRIPCIÓN TÉCNICA

La tabla tiene unas dimensiones de 51.1x31.6x3.2 cm y de un peso de 3.5 kg. El ancho se corresponde con la longitud media de un pie mientras que el largo se corresponde con la anchura de los hombros. Se diseño de esta manera ya que la posición más natural al subir a la tabla sería con los pies separados a la altura de los hombros. El aspecto es similar al de un step en la que se realizan ejercicios físicos (por ejemplo aeróbic) [40]. Cuenta con un solo botón en la parte inferior y con un led (véase Figura 6.1).

La tabla tiene cuatro patas en las esquinas cubiertas con gomas antideslizantes y resistentes a radiación de fuentes de calor. La tabla tiene una altura de 3.2 cm para minimizar el riesgo de caída accidental o por movimientos que pudieran producir ésta. Además tiene un cierto desnivel para que el usuario perciba en que parte de la tabla está colocando los pies aunque haya una zona preferentemente habilitada para ello. Para su transporte existen dos asas en la parte inferior.

Utiliza cuatro sensores de deformación para la obtención de las presiones ejercidas por los usuarios (véase Figura 6.2). Estos se sitúan en cada una de las patas de la tabla simplificando así el diseño que trasladó el dispositivo sensor de peso del interior al exterior. Al realizar la pisada se dobla el metal del sensor aunque no se aprecie a simple vista. Con la inclusión de estos componentes se gana en fiabilidad ya que se dobla muy poco y apenas requiere de componentes móviles. El número de sensores viene determinado por la estabilidad del sistema. Siendo cuatro el número de éstos se puede detectar una variación en el equilibrio en todas las direcciones y medir el peso con gran precisión. La Wiiboard soporta mecánicamente pesos de hasta 300 kilogramos pero los sensores solo pueden medir hasta los 150 kilogramos.

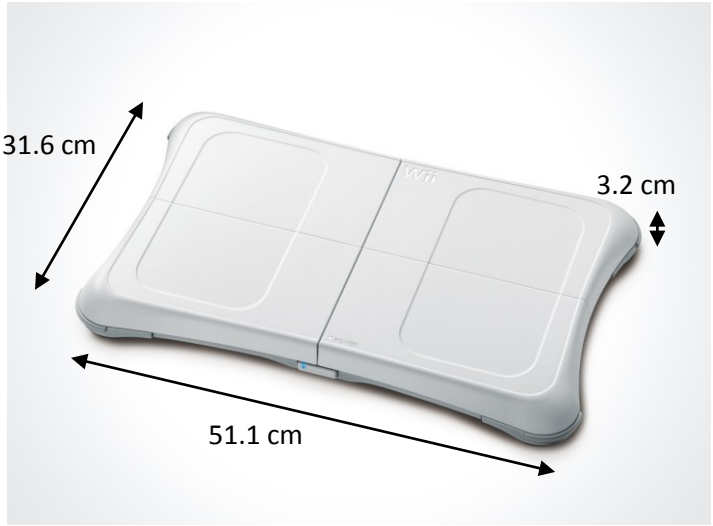

Figura 6.1. Vista de la Wii Balance Board

Figura 6.2. Vista de los sensores de deformación utilizados en la Wii Balance Board

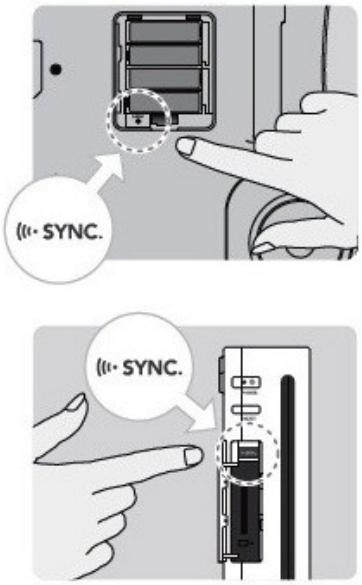

Figura 6.3. Sincronización de Wii Balance Board y consola Wii [3]

La comunicación es inalámbrica y se realiza mediante el protocolo Bluetooth. El diseño de la electrónica asociada se realizó de manera modular, pudiendo reutilizar este módulo de comunicaciones en futuros accesorios. Para su sincronización con la consola se ha de pulsar el botón SYNC situado dentro del compartimento dedicado a las pilas (véase Figura 6.3). Solo se puede utilizar una única WiiBoard con la consola, siendo compatible con el uso simultaneo de 3 Wiimotes. Usa la conexión del cuarto jugador desconectando los posibles Wiimotes que existieran en esa posición.

Para su alimentación son necesarias cuatro pilas AA. La autonomía media es de 60 horas de duración, similar a la del Wiimote cuando no se utiliza su función de puntero.

La mayoría de las actividades que se realizan con la WiiBoard se basan en las siguientes cuatro posiciones [41]:

- Equilibrarse sobre una pierna.
- Inclinar el cuerpo.
- Girar las caderas.
- Mover el cuerpo al ritmo.

6.3. API PARA LA PROGRAMACIÓN

IMPORTANTE: Para la programación de aplicaciones para la WiiBoard se ha realizar la instalación de la librería como se detalla en el Anexo D.

El API que se utilizará para la programación de aplicaciones es el mismo que se utiliza para el Wiimote y sus funcionalidades (apartado 3.4). El objeto que representa a la WiiBoard se trata de igual manera que el que representaba al Wiimote. Para identificar que el dispositivo conectado es la WiiBoard se indicará en el modo de reporte de mensajes. Posteriormente podremos manejar los valores de los sensores de presión accediendo al estado de la WiiBoard y seleccionando el sensor deseado.

Figura 6.4. Movimientos básicos en Wii Balance Board

6.4. EJEMPLO DE PROGRAMACIÓN

En este apartado se muestra el uso del API con un ejemplo que nos muestra por pantalla los valores de los cuatro sensores incorporados en la tabla. El programa completo está en el Anexo B.

En primer lugar creamos un objeto que represente la WiiBoard llamando al método `cwiid.Wiimote()`.

```
wiiboard = cwiid.Wiimote()
```

En caso de resultar errónea la operación, se lanzará la excepción RuntimeError. Si la conexión es exitosa, activamos el LED de la WiiBoard para confirmar la conexión.

```
wiiboard.led = cwiid.LED1_ON
```

A continuación, activamos el modo de envío de la WiiBoard.

```
wiiboard.rpt_mode = cwiid.RPT_BALANCE
```

La tabla enviará de manera continua el valor obtenido de la presión en los cuatro sensores internos. Se accederá a los valores mediante el estado de la tabla en cada momento.

```
state = wiiboard.state
```

Cada uno de los sensores viene identificado por su posición dentro de la tabla accediendo mediante ésta a su valor para su posterior muestra por pantalla.

```
a = state['balance']['left_top']

b = state['balance']['right_top']

c = state['balance']['left_bottom']

d = state['balance']['right_bottom']
```

6.5. APLICACIONES DESARROLLADAS

Grawiity_center

Grawiity_center es una aplicación que muestra el centro de gravedad en pantalla sobre una imagen de la WiiBoard. Muestra además con un dato numérico la presión ejercida en cada sensor. Además se nos presenta en pantalla el nivel de batería de la tabla y el peso del usuario que se coloca encima de la tabla (véase Figura 6.5).

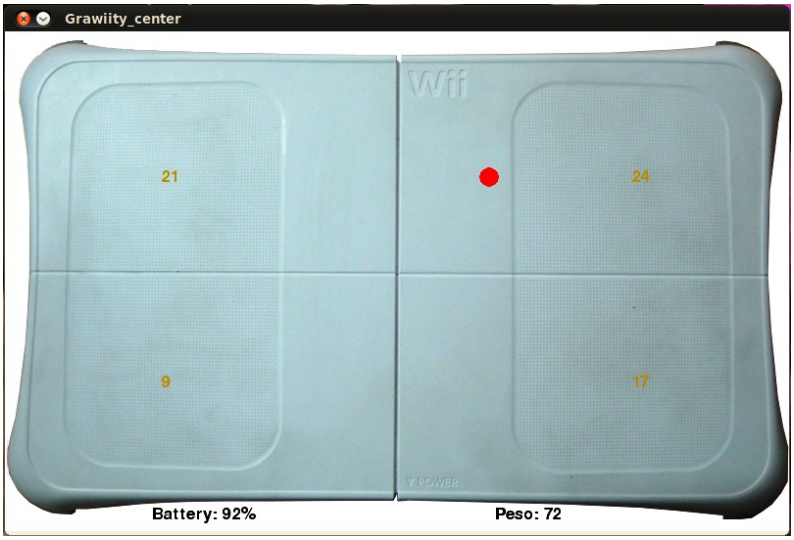

Figura 6.5. Vista de la aplicación Grawiity_center

Esta aplicación sirve de calibración permitiendo ajustar la sensibilidad y la posición. Estos parámetros varían en función de si el usuario se encuentra en posición erguida sobre la tabla o si se está sentado.

El proceso a seguir es el siguiente:

1. Lanzamiento de la aplicación
2. Sincronización con el ordenador. Esta se realiza pulsando el botón de SYNCRO situado tras la tapa para las pilas.
3. Posicionamiento del usuario.
4. Movimiento del usuario. Tras este paso comprobaremos el movimiento del centro de gravedad y ajustaremos nuestra posición para una correcta utilización.

Pong_wiiboard

Pong_wiiboard es una aplicación que adapta el popular juego del pong para ser controlado mediante la WiiBoard (véase Figura 3.7). Presenta la misma interfaz que la mostrada en la aplicación desarrollada Pong_wiimote (apartado 3.6).

Nuevamente falta la implementación de un “tanto” ya que nos centramos en la capacidad para manejar las capacidades ofrecidas por el dispositivo. La aplicación está enteramente controlada por el usuario que se coloca encima de la tabla. El movimiento de la pala que correspondería al jugador 1 se controla

situando levemente el centro de gravedad a la izquierda del usuario. Posteriormente el desplazamiento vertical se logra desplazando el centro de gravedad hacia adelante o hacia atrás según corresponda. Para el movimiento que correspondería al jugador 2, el usuario ha de desplazar levemente su centro de gravedad hacia la derecha. De igual manera que en el caso anterior, los movimientos de su centro de gravedad hacia adelante o hacia atrás se corresponderán con movimientos de la pala en sentido ascendente o descendente respectivamente.

Pong_wiiboard es una aplicación que se correspondería con los objetivos fijados para éstas: uso de la WiiBoard como control, entretenimiento durante el uso y **posibilidad de usos en fisioterapia o rehabilitación**. Esto es así ya que el control de la bola sin que toque las paredes laterales hace moverse al usuario guardando el equilibrio subido en la tabla. Además se ejercita la capacidad de concentración al variar continuamente la orientación y sentido del movimiento dentro de la pantalla. Adicionalmente se puede usar sentado sobre la tabla lo que beneficiaría al control del tronco, efecto deseado en varios pacientes de rehabilitación (véase Figura 6.6).

Figura 6.6. Vista de las posibles posiciones a adoptar en el uso de la aplicación Pong_wiiboard

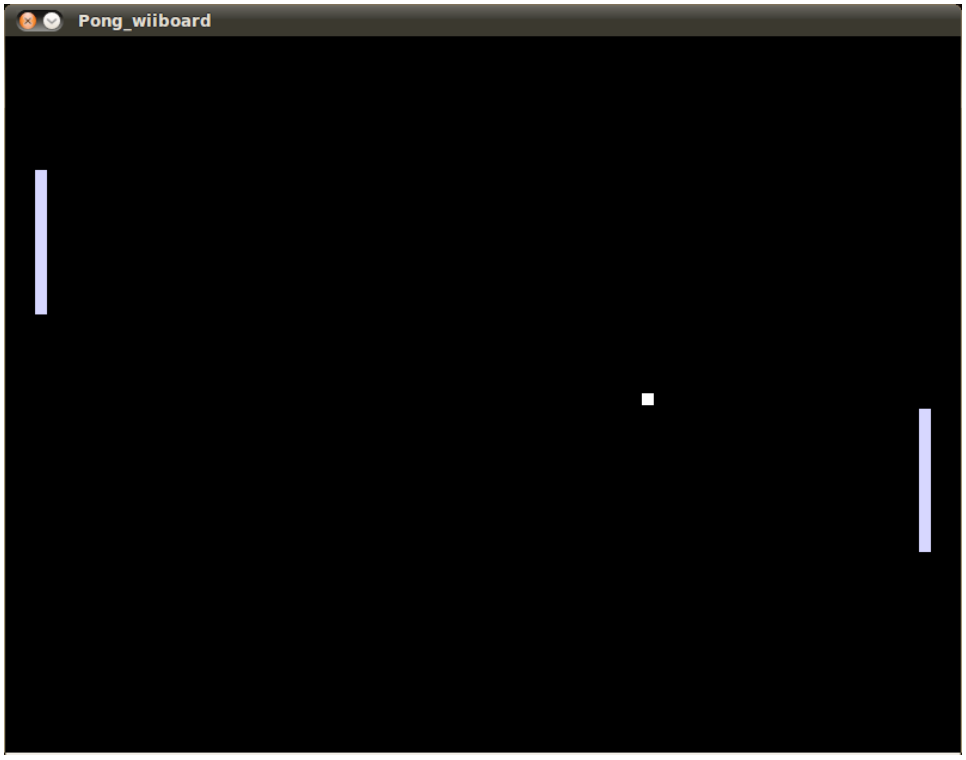

Figura 6.7. Vista de la aplicación Pong_wiiboard

6.6. RESUMEN

La WiiBoard se puede utilizar como dispositivo de control de aplicaciones en el ordenador. La detección de presión se puede realizar usando la tabla de pie sobre ella, sentados sobre su superficie o apoyando ambas manos. Además incluye una mayor versatilidad al disponer de una interfaz Bluetooth que hace que sea posible conectarla a un ordenador con facilidad.

Aunque su uso está pensado para que los usuarios se pongan de pie encima de ella, hemos comprobado que también funciona cuando los usuarios se sientan encima. Esto nos permite crear aplicaciones para personas que precisen de silla de ruedas o con un cierto grado de deficiencia en el tren inferior.

Capítulo 7
Conclusiones y trabajos futuros

7.1. CONCLUSIONES

El proyecto aborda el problema del prototipado rápido de aplicaciones Opensource para rehabilitación usando dispositivos inalámbricos. Se han estudiado los diferentes dispositivos inalámbricos existentes en el mercado presentado cuatro alternativas. Se ha analizado la manera en que se usa actualmente la consola Wii para la rehabilitación y las soluciones software que hay disponibles para desarrollar aplicaciones específicas.

La plataforma elegida está formada por elementos hardware y software. Los primeros son los periféricos de la Wii: Wiimote, Wii Motion Plus y Wii Balance Board. Los segundos son el sistema operativo Linux (Ubuntu 10.04), Cwiid, lenguaje Python y Pygame.

Todos los elementos hardware se han explicado y documentado detalladamente para que otros ingenieros puedan desarrollar aplicaciones para rehabilitación rápidamente. Se han analizado las posibilidades de los acelerómetros y cámara de detección de IR del Wiimote, el giróscopo del Wii Motion Plus y los sensores de deformación de la Wii Balance Board. Se ha documentado la API de programación y se han escrito programas de ejemplo para cada periférico.

Para comprobar la viabilidad de la plataforma se han programado aplicaciones de demostración para rehabilitación para cada uno de los periféricos: adaptación del juego del Pong para el Wiimote y Wiiboard, monitorización del levantamiento de una mancuerna, monitorización de velocidades angulares para el Wii Montion Plus y diversas aplicaciones de calibrado.

No sólo se han alcanzado satisfactoriamente los objetivos fijados inicialmente sino que además se han conseguido otras aportaciones:

- Se ha propuesto un interfaz nuevo que permite a los pacientes usar los movimientos del cuello para controlar las aplicaciones. Se usa un Wiimote colocado en una gorra. Es de gran utilidad para personas con discapacidad parcial o total en las manos o brazos.

- Se ha propuesto otro interfaz novedoso para que los usuarios puedan ejercer el control mediante movimientos del torso en una posición sentada. Se ha comprobado la viabilidad de utilizar la Wiiboard para que los pacientes se sienten sobre ella. Aunque no es un uso planeado inicialmente por Nintendo, los experimentos demuestran su viabilidad. Es especialmente interesante para personas con pérdida de movilidad en el tren inferior.

7.2. TRABAJOS FUTUROS

Algunos de los trabajos que dan continuidad a este proyecto son:

- **Control del Wiimote con diferentes partes del cuerpo.** Además del uso tradicional se podría controlar el Wiimote con una pierna si se sujeta a ella con un soporte específico. Se podría controlar con los antebrazos o sujeto al tronco si los movimientos a realizar implican un desplazamiento de éste.

- **Localización del Wiimote ajeno al cuerpo.** Se podrían desarrollar aplicaciones teniendo en cuenta que el Wiimote está acoplado a una de las máquinas de rehabilitación ya existentes. Por ejemplo, se podría monitorizar el giro de la muñeca de un paciente si colocamos el Wiimote en el eje de giro de la máqiuna rotatoria de muñeca.

- **Seguimiento avanzado de parte del cuerpo.** Se puede realizar un seguimiento avanzado si adherimos al cuerpo el cuatro emisores de IR o materiales reflectantes. De esta manera tendríamos una caracterización más completa de los movimientos durante el ejercicio. Se podría utilizar para extremidades o para personas con muy reducida movilidad gracias a la resolución de la cámara.

- **Control de móviles mediante movimiento del cuerpo.** Si adaptamos la salida que nos ofrece la WiiBoard se puede realizar control de elementos móviles que permitan desplazamiento. Una posibilidad sería la implementación del movimiento de una silla de ruedas mediante el movimiento del usuario sentado encima de ella siguiendo un esquema: silla, Wii Balance Board, usuario.

Otro uso no destinado a rehabilitación y fisioterapia sería el destinado a robótica asistencial. Se puede adaptar las salidas que nos ofrecen los mandos y sus accesorios para controlar robots asistenciales como ASIBOT.

Los trabajos futuros que extiendan a este proyecto irán dirigidos a la creación de aplicaciones más complejas. Además se ha de contar con la ayuda de uno o varios especialistas en fisioterapia o rehabilitación para que las aplicaciones estén hechas totalmente a la medida de la necesidad. Se recomienda desarrollar las aplicaciones mediante diseño centrado en el usuario y diseño participativo.

Anexo A

Instalación del entorno de trabajo

Elementos necesarios:

- Ordenador personal con sistema operativo Linux. Se usa la distribución Ubuntu 10.04. Se han probado los códigos desarrollados en su versión anterior (9.10) siendo las pruebas satisfactorias y garantizando su funcionamiento en ordenadores con esta versión instalada.
- Interfaz Bluetooth. Bluetooth interno o Bluetooth USB.

Adicionalmente se han de instalar en el ordenador tres paquetes correspondientes a los drivers del mando (lswm), a la librería de desarrollo de los programas (libcwiid) y a la interfaz bluetooth (bluez).

```
sudo apt-get install bluez lswm wmgui wminput libcwiid1-dev
```

Wmgui

Wmgui es una aplicación implementada por el equipo encargado de la librería de desarrollo Cwiid. Se trata de una aplicación que nos muestra una interfaz gráfica en la que se nos muestran diversas partes del Wiimote o de sus funciones. La parte superior se divide en tres: un área que muestra los diferentes

botones del Wiimote, un área que muestra los sensores de movimiento en los tres ejes y un área en el que se muestra las fuentes de infrarrojo captadas por la cámara del mando en forma de punto o puntos (hasta cuatro). La parte inferior presenta una parte dedicada a los controles del Nunchuk y otra parte dedicada a los controles del mando clásico.

Una vez lanzada la aplicación se ha de conectar el Wiimote a ella pulsando CTRL+C, la cual nos presentará una ventana que nos pedirá que pulsemos 1+2 en el mando. Una vez realizado el proceso el mando se encontrará vinculado a la aplicación y si pulsamos cualquier botón, éste aparecerá coloreado en la parte correspondiente a los botones. Para activar el sensor de movimiento se ha de pulsar CTRL+A, el cual mandará las señales que se mostrarán en su espacio dedicado en forma numérica. Para la visualización de las fuentes de infrarrojo mediante la cámara se ha de pulsar CTRL+I. Las fuentes aparecerán como puntos dentro del rango visible del Wiimote mapeando en dos dimensiones su movimiento y haciendo que el punto sea más grande o más pequeño dependiendo de su distancia al mando.

Permite además encender o apagar los LEDs correspondientes a los juagadores pulsando CTRL+1, CTRL+2, CTRL+3 o CTRL+4 respectivamente, y encender o apagar el vibrador interno pulsando CTRL+R. La parte inferior correspondiente a las extensiones se habilita pulsando CTRL+E. Si los periféricos correspondientes están conectados aparecerá en una barra de estado inferior en la que además se muestra si el Wiimote se encuentra vinculado al programa y el estado de la batería.

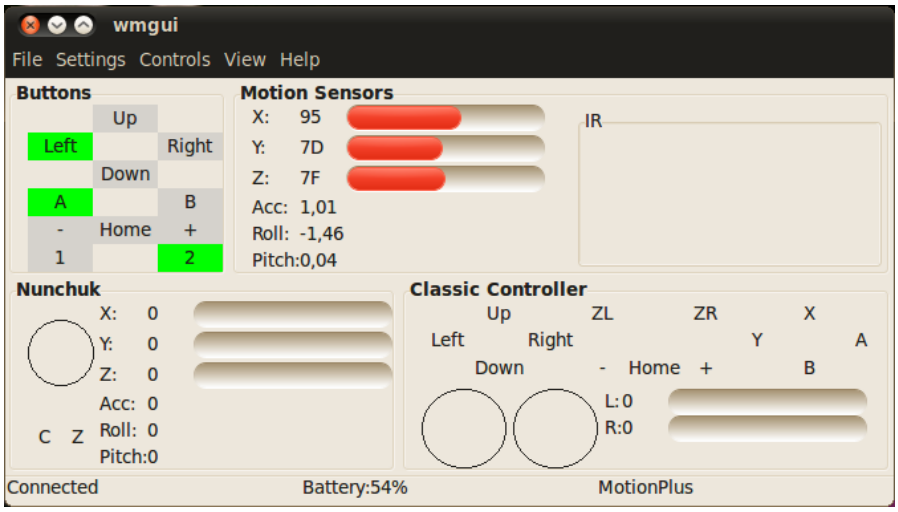

Figura A.1. Vista de la aplicación wmgui interactuando con los botones y con el acelerómetro

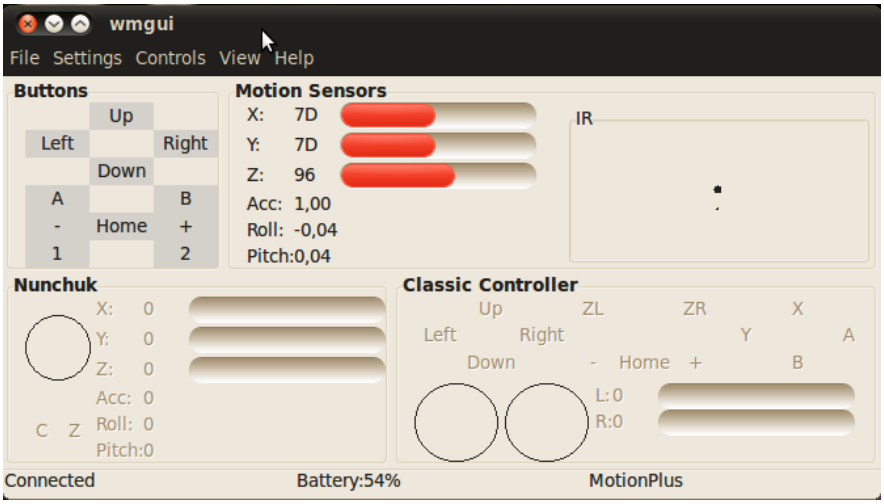

Figura A.2. Vista de la aplicación wmgui interactuando con la cámara de IR y con el acelerómetro

IMPORTANTE: Para la programación de aplicaciones para la WiiBoard se ha realizar la instalación de la librería como se detalla en el Anexo D.

Anexo B

Ejemplos de programación

1. LECTURA DE ACELERÓMETRO

```
#-------------------------------------------------------------------
#--  ej_libcwiid_acc.py
#--  Ejemplo lectura acelerometro
#-------------------------------------------------------------------
#-- Carlos Pastor
#-------------------------------------------------------------------

#Importar modulos
import sys
import cwiid
import time

#Variables
led=0;
```

```
reintentos=0;
ok=False;

print 'Pulsa las teclas 1+2 en el Wiimote...'

#Establecer conexion con Wiimote
while reintentos<3 and not ok:
  try:
    print "Reintento ",reintentos
    if len(sys.argv) > 1:
      wiimote = cwiid.Wiimote(sys.argv[1])
    else:
      wiimote = cwiid.Wiimote()
    ok=True
  except RuntimeError, msg:
    reintentos+=1;

#Si se superan los reintentos, terminar
if reintentos==3:
  print "Sin conexion con wiimote", reintentos
  sys.exit(-1)

#Conexion establecida
#Encendemos el led 1 para saber que la conexion esta establecida
wiimote.led = cwiid.LED1_ON

#Activamos los acelerometros
wiimote.rpt_mode = cwiid.RPT_ACC

#Bucle principal
while 1:
    # Leer las aceleraciones
    acc = wiimote.state['acc']
```

```
    # Sacar por pantalla

    str = "X: %.1f, Y: %.1f, Z: %.1f " %
acc[cwiid.X],acc[cwiid.Y],acc[cwiid.Z])

    print str


    #Introducimos una pausa para mejorar la visualizacion

    time.sleep(0.5)
```

2. COORDENADAS DE LED IR

```
#-------------------------------------------------------------------
#--  ej_posicion_ir.py
#--  Ejemplo coordenadas IR LED
#-------------------------------------------------------------------
#-- Carlos Pastor
#-------------------------------------------------------------------

#Importar modulos
import sys
import cwiid
import time

#Variables
led=0;
reintentos=0;
ok=False;

print 'Pulsa las teclas 1+2 en el Wiimote......'

#Establecer conexion con Wiimote
while reintentos<3 and not ok:
  try:
    print "Reintento ",reintentos
    if len(sys.argv) > 1:
      wiimote = cwiid.Wiimote(sys.argv[1])
    else:
      wiimote = cwiid.Wiimote()
    ok=True
  except RuntimeError, msg:
    reintentos+=1;
```

```
#Si se superan los reintentos, terminar

if reintentos==3:

  print "Sin conexion con wiimote despues de "+repr(reintentos)+"
reintentos"

  sys.exit(1)


#Conexion establecida

#Encender el led 1 del wiimote

wiimote.led = cwiid.LED1_ON


#Activamos ir

wiimote.rpt_mode = cwiid.RPT_IR


#Bucle principal

while 1:

      #Acceder al estado del wiimote

      state = wiimote.state


      if 'ir_src' in state:

            for src in state['ir_src']:

                  if src:

                        posicion=src['pos']

                  print posicion


      #Introducimos una pausa para mejorar la visualizacion

      time.sleep(0.1)
```

3. VALORES WII MOTION PLUS

```
#-------------------------------------------------------------------
#--  ej_wiimotionplus.py
#--  Ejemplo valores ofrecidos por Wii Motion Plus
#-------------------------------------------------------------------
#-- Carlos Pastor
#-------------------------------------------------------------------

#Importar modulos
import sys
import cwiid
import time

#Variables
led=0;
reintentos=0;
ok=False;

print 'Pulsa las teclas 1+2 en el Wiimote......'

#Establecer conexion con Wiimote
while reintentos<3 and not ok:
  try:
    print "Reintento ",reintentos
    if len(sys.argv) > 1:
      wiimote = cwiid.Wiimote(sys.argv[1])
    else:
      wiimote = cwiid.Wiimote()
    ok=True
  except RuntimeError, msg:
    reintentos+=1;
```

```
#Si se superan los reintentos, terminar

if reintentos==3:

  print "Sin conexion con wiimote despues de "+repr(reintentos)+"
reintentos"

  sys.exit(1)


#Conexion establecida

#Encendemos el led 1 del wiimote

wiimote.led = cwiid.LED1_ON


#Activar Wii Motion Plus

wiimote.enable(cwiid.FLAG_MOTIONPLUS)

wiimote.rpt_mode = cwiid.RPT_MOTIONPLUS


#Bucle principal

while 1:

      if wiimote.state.has_key('motionplus'):

            #Leemos velocidades angulares del estado del wiimote

            vel_ang=wiimote.state['motionplus']['angle_rate']


            print vel_ang


      #Introducimos una pausa para mejorar la visualizacion

      time.sleep(0.1)
```

4. VALORES WII BALANCE BOARD

```
#-----------------------------------------------------------------

#--  ej_wiiboard.py

#--  Ejemplo valores ofrecidos por WiiBoard

#-----------------------------------------------------------------

#-- Carlos Pastor

#-----------------------------------------------------------------


#Importar modulos

import sys

import cwiid

import time

#Variables

led=0;

reintentos=0;

ok=False;

print 'Pulsa el boton de sync de la wiiboard (donde estan las
pilas)...'

#Establecer conexion con Wiimote

while reintentos<3 and not ok:

  try:

    print "Reintento ",reintentos

    if len(sys.argv) > 1:

      wiimote = cwiid.Wiimote(sys.argv[1])

    else:

      wiimote = cwiid.Wiimote()

    ok=True

  except RuntimeError, msg:

    reintentos+=1;
```

```
#Si se superan los reintentos, terminar
if reintentos==3:
  print "Sin conexion con wiimote", reintentos
  sys.exit(-1)

#Conexion establecida
#Encender el led de la wii-board
wiimote.led = cwiid.LED1_ON

#Activar la wii-board
wiimote.rpt_mode = cwiid.RPT_BALANCE

#Bucle principal
while 1:
       #Leer el estado de la wiiboard
       state = wiimote.state

       #Leer los 4 sensores de presion
       a = state['balance']['left_top']
       b = state['balance']['right_top']
       c = state['balance']['left_bottom']
       d = state['balance']['right_bottom']

       #Imprimir en pantalla
       str = "A=%d, B=%d, C=%d, D=%d" % (a,b,c,d)
       print str

       #Introducimos una pausa para mejorar la visualizacion
       time.sleep(0.1)
```

Anexo C

Aplicaciones desarrolladas

1. WIIMOTE_2D

```
#------------------------------------------------------------------
#--  wiimote_2D.py
#--  Movimiento de punto en 2D
#------------------------------------------------------------------
#-- Carlos Pastor
#------------------------------------------------------------------

# Importar modulos
import sys
import cwiid
import math
import time
import decimal
```

```
import consola_io

import pygame


def menu():

  print """

     Menu de opciones

     ----------------

     1.- Opcion 1= Mando en mano

     2.- Opcion 2= Mando en gorra

  SP.- Volver a sacar el menu

  ESC.- Terminar

  """


def lee_pitch_roll(wiimote):

      # Leer el calibrado

      ext_type = wiimote.state['ext_type']

      cal = wiimote.get_acc_cal(ext_type)


      # Leer las aceleraciones

      acc = wiimote.state['acc']


      #Calcular las aceleraciones normalizadas

      a_x = float( acc[cwiid.X] - cal[0][cwiid.X] ) / float(
cal[1][cwiid.X] - cal[0][cwiid.X] )

      a_y = float( acc[cwiid.Y] - cal[0][cwiid.Y] ) / float(
cal[1][cwiid.Y] - cal[0][cwiid.Y] )

      a_z = float( acc[cwiid.Z] - cal[0][cwiid.Z] ) / float(
cal[1][cwiid.Z] - cal[0][cwiid.Z] )


      #Calcular pitch y roll

      if a_z!=0:

          roll = math.atan(a_x/a_z);
```

```
      else:
          roll = math.pi/2;
      if (a_z<=0.0):
          if (a_x!=0):
            signo = (a_x/a_x)
            roll += math.pi * signo
          else:
            roll += math.pi
      roll *= -1
      divisor = a_z*math.cos(roll)

      if divisor!=0:
            pitch = math.atan(a_y/a_z*math.cos(roll))

      else:
            pitch=0

      #Pasar pitch y roll a grados
      roll = roll * 180.0/math.pi
      pitch = pitch * 180.0/math.pi

      return (pitch, roll)

#Comienzo programa
led=0;
reintentos=0;
ok=False;
blue = 0, 0, 255
black = 0, 0, 0
white = 255, 255, 255
red= 255, 0, 0
```

```
mano=0

gorra=0


print 'Pulsa las teclas 1+2 en el Wiimote......'


#Establecer conexion con Wiimote

while reintentos<3 and not ok:

  try:

    print "Reintento ",reintentos

    if len(sys.argv) > 1:

      wiimote = cwiid.Wiimote(sys.argv[1])

    else:

      wiimote = cwiid.Wiimote()

    ok=True

  except RuntimeError, msg:
    reintentos+=1;


#Si se superan los reintentos, terminar

if reintentos==3:

  print "Sin conexion con wiimote despues de "+repr(reintentos)+"
reintentos"

  sys.exit(1)


#Conexion establecida

#Encender el led 1 del wiimote

wiimote.led = cwiid.LED1_ON


#Activar acelerometros

wiimote.rpt_mode = cwiid.RPT_ACC


#Sacar menu, leer tecla y procesarla

menu()
```

```
c = consola_io.getkey()

#Procesar tecla
if   c=='1':
      mano=1
elif c=='2':
      gorra=1
elif c==' ':
      menu()

#Configurar pygame
pygame.init()
size = width, height = 640, 400
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Wiimote_2D")

#Coordenadas para el centro del control
cox = width/2
coy = height/2

#Bucle principal
while 1:
      for eventos in pygame.event.get():
            if eventos.type == pygame.QUIT :
                sys.exit(0)

      (pitch,roll) = lee_pitch_roll(wiimote)

      #Limitando accion de pitch y roll a giros de 90
      if mano==1:
            if pitch>90:
```

```
                pitch=90
          if pitch<-90:
                pitch=-90
          if roll>90:
                roll=90
          if roll<-90:
                roll=-90

    #Limitando accion de pitch y roll a giros de 22.5
    if gorra==1:
          if pitch>22.5:
                pitch=22.5
          if pitch<-22.5:
                pitch=-22.5
          if roll>22.5:
                roll=22.5
          if roll<-22.5:
                roll=-22.5



    #Borrar lo que habia antes
    screen.fill(white)


    #Dibujar ejes y punto
      pygame.draw.aaline (screen, black, (cox-cox, coy),
(cox+cox, coy),5)
      pygame.draw.aaline (screen, black, (cox, coy-coy), (cox,
coy+coy),5)



    #Pasar a decimal y cuantizar numero de cifras decimales
    TWOPLACES = decimal.Decimal(repr(0.01))
```

```
      rn=decimal.Decimal(repr(roll)).quantize(TWOPLACES)

      pn=decimal.Decimal(repr(pitch)).quantize(TWOPLACES)


      #Para mando en posicion perpendicular a la pantalla(mano)
cox-rn y coy+pn

      #Para mando en posicion paralela a la pantalla(gorra) cox-pn
y coy+rn

      if mano==1:

            dox=cox-(rn*width/180)

            doy=coy+(pn*height/180)

      if gorra==1:

            dox=cox-(pn*width/45)

            doy=coy-(rn*height/45)


      pygame.draw.circle(screen, red, (dox,doy),5,0)

      #Refrescar pantalla

      pygame.display.flip()

      time.sleep(0.1)
```

2. PONG_WIIMOTE

```
#------------------------------------------------------------------

#--  pong_wiimote.py

#--  Clasico juego pong adaptado para que el jugador 1 use un
wiimote como interfaz de control

#------------------------------------------------------------------

#-- Carlos Pastor

#------------------------------------------------------------------


#Importar modulos

import pygame

from pygame.locals import Rect, K_ESCAPE, QUIT, KEYDOWN, KEYUP,
K_UP, K_DOWN, K_q, K_a

import sys

import math

import cwiid


#Variables

SCREEN_DIMENSIONS = (800, 600)

screen = None

clock = None

paddle1 = None

paddle2 = None

ball = None


#Datos de wiimote

reintentos=0

ok=False

pitchPrev=0


#Establecer conexion con Wiimote
```

```
while reintentos<3 and not ok:
      try:
          print "Reintento ",reintentos
            if len(sys.argv) > 1:
              wiimote = cwiid.Wiimote(sys.argv[1])
            else:
              wiimote = cwiid.Wiimote()
            ok=True
        except RuntimeError, msg:
          print "Error: ",msg
          reintentos+=1;

      #Si se superan los reintentos, terminar
      if reintentos==3:
        print "Sin conexion con wiimote despues de
"+repr(reintentos)+" reintentos"
        sys.exit(1)

      #Encender el led 1 del wiimote
      wiimote.led = cwiid.LED1_ON

      #Activar acelerometros
      wiimote.rpt_mode = cwiid.RPT_ACC

#Rutina de inicio
def main():
    global screen
    global clock
    global SCREEN_DIMENSIONS
```

```
    #Inicializando todo lo relacionado con Pygame
    pygame.init()
    screen = pygame.display.set_mode( SCREEN_DIMENSIONS )
    pygame.display.set_caption("Pong_wiimote")
    clock = pygame.time.Clock()
    initData()
    mainLoop()

#Inicializacion de estructuras de datos
def initData():
    global paddle1
    global paddle2
    global ball


    paddle1 = Paddle(30, 300)
    paddle2 = Paddle(770, 300)
    ball = Ball(400, 300, 3, 3)

#Bucle principal
def mainLoop():
    global clock

    while True:
      renderScreen()
      readInput()
      updateState()
      clock.tick(120)

#Funcion que dibuja la pantalla
def renderScreen():
```

```
    global screen


    s = screen

    s.lock()

    s.fill(0)

    s.fill(0xffffff, Rect(ball.left(), ball.top(), ball.width,
ball.height))

    s.fill(0xccccff, Rect(paddle1.left(), paddle1.top(),
paddle1.width, paddle1.height))

    s.fill(0xccccff, Rect(paddle2.left(), paddle2.top(),
paddle2.width, paddle2.height))

    s.unlock()

    pygame.display.update()


#Lectura de la entrada de teclado y eventos del wiimote

def readInput():

    global paddle1

    global paddle2

    global pitchPrev


    pitch = lee_pitch_roll(wiimote)


    paddle1.dy=0.2*abs(pitch)

    paddle2.dy=8


    for event in pygame.event.get():

      if event.type == QUIT:

         quit()

      elif event.type == KEYDOWN:

         if event.key == K_ESCAPE:

            quit()

         elif event.key == K_UP:
```

```
            paddle2.joystick.up = True
        elif event.key == K_DOWN:
            paddle2.joystick.down = True

    elif event.type == KEYUP:
        if event.key == K_UP:
            paddle2.joystick.up = False
        elif event.key == K_DOWN:
            paddle2.joystick.down = False

    if abs(pitch-pitchPrev)> 0.5:
        if pitch < 0:
            paddle1.joystick.up = True
        else:
            paddle1.joystick.up= False
        if pitch > 0:
            paddle1.joystick.down= True
        else:
            paddle1.joystick.down = False
    else:
        paddle1.joystick.up= False
        paddle1.joystick.down= False

    pitchPrev=pitch

#Actualizar estado del juego
def updateState():
    global paddle1
    global paddle2
    global ball
```

```
    if (paddle1.joystick.up):
        paddle1.moveUp()
    elif (paddle1.joystick.down):
        paddle1.moveDown()

     if (paddle2.joystick.up):
        paddle2.moveUp()
    elif (paddle2.joystick.down):
        paddle2.moveDown()

    ball.update(paddle1, paddle2)

#Obtener pitch y roll a partir de aceleraciones
def lee_pitch_roll(wiimote):
      # Leer el calibrado
      ext_type = wiimote.state['ext_type']
      cal = wiimote.get_acc_cal(ext_type)

      # Leer las aceleraciones
      acc = wiimote.state['acc']

      # Calcular las aceleraciones normalizadas
      a_x = float( acc[cwiid.X] - cal[0][cwiid.X] ) / float(
cal[1][cwiid.X] - cal[0][cwiid.X] )
      a_y = float( acc[cwiid.Y] - cal[0][cwiid.Y] ) / float(
cal[1][cwiid.Y] - cal[0][cwiid.Y] )
      a_z = float( acc[cwiid.Z] - cal[0][cwiid.Z] ) / float(
cal[1][cwiid.Z] - cal[0][cwiid.Z] )

      #Calcular pitch y roll
      if a_z!=0:
          roll = math.atan(a_x/a_z);
      else:
```

```
            roll = math.pi/2;
        if (a_z<=0.0):
            if (a_x!=0):
              signo = (a_x/a_x)
              roll += math.pi * signo
            else:
              roll += math.pi

        roll *= -1
        divisor = a_z*math.cos(roll)

        if divisor!=0:
              pitch = math.atan(a_y/a_z*math.cos(roll))
        else:
              pitch=0

        #Pasar pitch a grados
        pitch = pitch * 180.0/math.pi

        return (pitch)

#Clase joystick
class Joystick:
    up = False
    down = False

#Clase bola
class Ball:
    x = 0
    y = 0
    dx = 0
```

```
dy = 0
width = 10
height = 10

def __init__(self, x, y, dx, dy):
    self.x = x
    self.y = y

    self.dx = dx
    self.dy = dy

def top(self):
    return self.y - self.height/2

def left(self):
    return self.x - self.width/2


def update(self, paddle1, paddle2):
    #Actualizo posicion
    self.x += self.dx
    self.y += self.dy

    #Deteccion de colision con las paletas
    if paddle1.collides((self.x, self.y)):
        self.dx = -self.dx
    if paddle2.collides((self.x, self.y)):
        self.dx = -self.dx

    #Deteccion de colision con los bordes de la ventana
    if (self.x < 0):
        self.x = 1
```

```
            self.dx = -self.dx
        if (self.x > SCREEN_DIMENSIONS[0]):
            self.x = SCREEN_DIMENSIONS[0]-1
            self.dx = -self.dx
        if (self.y < 0):
            self.y = 1
            self.dy = -self.dy
        if (self.y > SCREEN_DIMENSIONS[1]):
            self.y = SCREEN_DIMENSIONS[1]-1
            self.dy = -self.dy

#Clase paleta
class Paddle:
    x = 0
    y = 0
    #Velocidad vertical de la paleta
    dy = 0

    width = 10
    height = 120
    joystick = None

    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.joystick = Joystick()

    def top(self):
        return self.y - self.height/2

    def bottom(self):
```

```
        return self.y + self.height/2

    def left(self):
        return self.x - self.width/2

    def right(self):
        return self.x + self.width/2

    def moveUp(self):
        self.y -= self.dy
        self.y = max([self.height/2, self.y])

    def moveDown(self):
        self.y += self.dy
        self.y = min([self.y, SCREEN_DIMENSIONS[1]-self.height/2])

    def collides(self, point):
        return (point[0] >= self.left() and point[0] <=
self.right()) and (point[1] >= self.top() and point[1] <=
self.bottom())


#Inicio del juego
main()
```

3. WII_LED

```
#-----------------------------------------------------------------

#--  wii_led.py

#--  Aplicacion de seguimiento de puntero led y representacion en
2D

#-----------------------------------------------------------------

#-- Carlos Pastor

#-----------------------------------------------------------------


# Importar modulos

import sys

import cwiid

import math

import time

import decimal

import pygame

import wave

import os


# Variables

led=0;

reintentos=0;

ok=False;

blue = 0, 0, 255

black = 0, 0, 0

white = 255, 255, 255

red= 255, 0, 0


print 'Pulsa las teclas 1+2 en el Wiimote......'
```

```
#Establecer conexion con Wiimote
while reintentos<3 and not ok:

  try:
    print "Reintento ",reintentos
    if len(sys.argv) > 1:
      wiimote = cwiid.Wiimote(sys.argv[1])
    else:
      wiimote = cwiid.Wiimote()
    ok=True
  except RuntimeError, msg:
    reintentos+=1;

#Si se superan los reintentos, terminar
if reintentos==3:
  print "Sin conexion con wiimote despues de "+repr(reintentos)+"
reintentos"
  sys.exit(1)

#Conexion establecida
#Encender el led 1 del wiimote
wiimote.led = cwiid.LED1_ON

#Activar acelerometros y ir
wiimote.rpt_mode = cwiid.RPT_IR

#Configurar pygame
pygame.init()
size = width, height = 512, 384
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Wii_led")
```

```
#Coordenadas para el centro del control

cox = width/2

coy = height/2


#Bucle principal

while 1:

      screen.fill(white)


      #Dibujar ejes y punto

        pygame.draw.aaline (screen, black, (cox-cox, coy),
(cox+cox, coy),5)

        pygame.draw.aaline (screen, black, (cox, coy-coy), (cox,
coy+coy),5)


      for eventos in pygame.event.get():

            if eventos.type == pygame.QUIT :

                sys.exit(0)


      state = wiimote.state


      if 'ir_src' in state:

            for src in state['ir_src']:

                  if src:

                        posicion=src['pos']


                  cx=posicion[0]/2

                  cy=posicion[1]/2

                  pygame.draw.circle(screen, red, (cox+cox-
cx,coy+coy-cy),5,0)


      pygame.display.flip()
```

4. WII_MANCUERNA

```
#-----------------------------------------------------------------

#--  wii_mancuerna.py

#--  Aplicacion de seguimiento de brazo con mancuerna IR

#-----------------------------------------------------------------

#-- Carlos Pastor

#-----------------------------------------------------------------


#Importar modulos

import sys

import cwiid

import math

import time

import pygame

import wave

import os


#Variables

led=0;

reintentos=0;

ok=False;

blue = 0, 0, 255

black = 0, 0, 0

white = 255, 255, 255

red= 255, 0, 0

ceroX2=0

ceroY2=0


print 'Pulsa las teclas 1+2 en el Wiimote......'
```

```
#Establecer conexion con Wiimote
while reintentos<3 and not ok:
  try:
    print "Reintento ",reintentos
    if len(sys.argv) > 1:
      wiimote = cwiid.Wiimote(sys.argv[1])
    else:
      wiimote = cwiid.Wiimote()
    ok=True
  except RuntimeError, msg:
    reintentos+=1;

#Si se superan los reintentos, terminar
if reintentos==3:
  print "Sin conexion con wiimote despues de "+repr(reintentos)+"
reintentos"
  sys.exit(1)

#Conexion establecida
#Encender el led 1 del wiimote
wiimote.led = cwiid.LED1_ON

#Activar ir
wiimote.rpt_mode = cwiid.RPT_IR

#Configurar pygame
pygame.init()
size = width, height = 512, 384
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Wii_mancuerna")
```

```
#Coordenadas para el centro del control
cox = width/2
coy = 2*height/3

#Calibracion de la aplicacion
print 'Coloque brazo en posicion inicial'

time.sleep( 10 )

state = wiimote.state

if 'ir_src' in state:
    src= state['ir_src'][0]
    if src:
        posCal2=src['pos']

      ceroX2=posCal2[0]
      ceroY2=posCal2[1]

print 'Calibrado OK'

time.sleep(5)

#Comienzo de la aplicacion
print 'Go!'

while 1:
      screen.fill(white)
      pygame.draw.circle(screen,blue, (cox,coy),5,0)
      font = pygame.font.Font(None, 20)

      #Dibujar ejes y punto
```

```
      pygame.draw.aaline (screen, black, (cox-(width/4), coy),
(cox+cox, coy),5)


        pygame.draw.aaline (screen, black, (cox, coy-coy), (cox,
coy+(height/6)),5)


      for eventos in pygame.event.get():

            if eventos.type == pygame.QUIT :

                sys.exit(0)


      state = wiimote.state


      if 'ir_src' in state:

           for src in state['ir_src']:

               if src:

                        posicion=src['pos']


                  cx=posicion[0]

                  cy=posicion[1]


                  xp=1023-cx

                  yp=cy-ceroY2


                  #Calculo del angulo del brazo respecto a la
horizontal

                  angulo=math.atan2(yp,xp)

                  anguloGrados= math.degrees(angulo)


                  #Representacion en pantalla de valores

                  text_angulo = font.render('Angulo: ', True,
black, white)

                  text_anguloN = font.render(repr(anguloGrados),
True, black, white)

                  screen.blit(text_angulo, (50, 50))
```

```
                    screen.blit(text_anguloN, (100, 50))

                    pygame.draw.aaline (screen, red, (cox, coy),
(cox+xp, coy-yp))



      pygame.display.flip()


      time.sleep(0.1)
```

5. WII_MOTIONPLUS

```
#-----------------------------------------------------------------

#--  Wii_motionPlus.py

#--  Representacion de las velocidades captadas por el WMP

#-----------------------------------------------------------------

#-- Carlos Pastor

#-----------------------------------------------------------------


#Importar modulos

import cwiid

import sys

import time

import pygame


#Variables

size = width, height = 768, 576

reintentos=0

ok=False

positivoP, negativoP= False,False

positivoR, negativoR= False,False

positivoY, negativoY= False,False

dP,dR,dY=0,0,0

xP,xR,xY=0,0,0

yaw,pitch,roll=0,0,0

white=255,255,255

red=255,0,0

black=0,0,0

blue=0,0,255

green=0,255,0
```

```
vel_ang=0

def comprobar_orientacion(yaw,pitch,roll):
      if yaw>8000:
            positivoY= True
            negativoY= False
      else:
            negativoY= True
            positivoY= False
      if pitch>7860:
            positivoP= True
            negativoP= False
      else:
            negativoP= True
            positivoP= False
      if roll>7980:
            positivoR= True
            negativoR= False
      else:
            negativoR= True
            positivoR= False

      return
(positivoY,negativoY,positivoR,negativoR,positivoP,negativoP)

print 'Pulsa las teclas 1+2 en el Wiimote......'

#Establecer conexion con Wiimote
while reintentos<3 and not ok:
  try:
    print "Reintento ",reintentos
```

```
    if len(sys.argv) > 1:
      wiimote = cwiid.Wiimote(sys.argv[1])

    else:
      wiimote = cwiid.Wiimote()
    ok=True
  except RuntimeError, msg:
    reintentos+=1;

#Si se superan los reintentos, terminar
if reintentos==3:
  print "Sin conexion con wiimote despues de "+repr(reintentos)+"
reintentos"
  sys.exit(1)

#Coordenadas para los centros del control
cox = width/2
coyP = height/6
coyR = height/2
coyY = 5*height/6

#Activar MotionPlus
wiimote.enable(cwiid.FLAG_MOTIONPLUS)
wiimote.rpt_mode = cwiid.RPT_MOTIONPLUS

#Configurar pygame
pygame.init()
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Wii_motionPlus")
font = pygame.font.Font(None, 30)

while 1:
```

```
      for eventos in pygame.event.get():
            if eventos.type == pygame.QUIT :
                sys.exit(0)

      if wiimote.state.has_key('motionplus'):
            vel_ang=wiimote.state['motionplus']['angle_rate']
            pitch=vel_ang[0]
            roll=vel_ang[1]
            yaw=vel_ang[2]

            xP=(pitch/100)*(width-100)/(16383/100)
            xR=(roll/100)*(width-100)/(16383/100)
            xY=(yaw/100)*(width-100)/(16825/100)

      (positivoY,negativoY,positivoR,negativoR,positivoP,negativoP
)=comprobar_orientacion(yaw,pitch,roll)

            if positivoP==True:
                  dP=-(xP-((7860/100)*(width-100)/(16383/100)))
            else:
                  dP=((7860/100)*(width-100)/(16383/100))-xP
            if positivoR==True:
                  dR=-(xR-((7980/100)*(width-100)/(16383/100)))
            else:
                  dR=((7980/100)*(width-100)/(16383/100))-xR
            if positivoY==True:
                  dY=-(xY-((8000/100)*(width-100)/(16825/100)))
            else:
                  dY=((8000/100)*(width-100)/(16825/100))-xY

      #Borrar lo que habia antes
```

```
      screen.fill(white)


      #Dibujar elementos graficos

      text_pitch = font.render('PITCH', True, black, white)

      text_roll = font.render('ROLL', True, black, white)


      text_yaw = font.render('YAW',True,black,white)

      screen.blit(text_pitch, (50, height/6-40))

      screen.blit(text_roll, (50, height/2-40))

      screen.blit(text_yaw, (50,5*height/6-40))

      pygame.draw.aaline (screen, black, (50, height/6), (width-
50, height/6),5)

      pygame.draw.aaline (screen, black, (50, height/2), (width-
50, height/2),5)

      pygame.draw.aaline (screen, black, (50, 5*height/6), (width-
50, 5*height/6),5)

      pygame.draw.circle(screen, black, (cox,coyP),5,0)

      pygame.draw.circle(screen, black, (cox,coyR),5,0)

      pygame.draw.circle(screen, black, (cox,coyY),5,0)

      pygame.draw.circle(screen, red, (cox+dP,coyP),6,0)

      pygame.draw.circle(screen, green, (cox+dR,coyR),6,0)

      pygame.draw.circle(screen, blue, (cox+dY,coyY),6,0)

      pygame.display.flip()

      time.sleep(0.1)
```

6. GRAWIITY CENTER

```
#-------------------------------------------------------------------

#--  grawiity_center.py

#--  Mapeo de los movimientos sobre la tabla(centro de gravedad)
en un   espacio en 2 dimensiones

#-------------------------------------------------------------------

#-- Carlos Pastor

#-------------------------------------------------------------------


#Importar modulos

import sys

import cwiid

import math

import time

import pygame

#Devolver una cadena con el estado de la bateria

def estado_bateria (state):

      bateria = state['battery']

      porcentaje = (100.0 * bateria / cwiid.BATTERY_MAX)

      return 'Battery: ' + str(int(porcentaje)) + '%'

#Leer los valores normalizados de los 4 sensores

def leer_sensores(wiimote,calibration):

  state = wiimote.state

  sensor = 'left_top'

  valor = state['balance'][sensor]

  valor0 = calibration[sensor][0]

  valor1 = calibration[sensor][1]
```

```
  a = 1700 * float((valor - valor0)) / float((valor1-valor0))

  sensor = 'right_top'
  valor = state['balance'][sensor]
  valor0 = calibration[sensor][0]
  valor1 = calibration[sensor][1]

  b = 1700 * float((valor - valor0)) / float((valor1-valor0))
  sensor = 'left_bottom'
  valor = state['balance'][sensor]
  valor0 = calibration[sensor][0]
  valor1 = calibration[sensor][1]

  c = 1700 * float((valor - valor0)) / float((valor1-valor0))
  sensor = 'right_bottom'
  valor = state['balance'][sensor]
  valor0 = calibration[sensor][0]
  valor1 = calibration[sensor][1]

  d = 1700 * float((valor - valor0)) / float((valor1-valor0))

  return (a,b,c,d)

#Comienzo programa
reintentos=0;
ok=False;

print 'Pulsa el boton de sync de la wiiboard (donde estan las
pilas)...'

while reintentos<3 and not ok:
  try:
    print "Reintento ",reintentos
```

```
    if len(sys.argv) > 1:
      wiimote = cwiid.Wiimote(sys.argv[1])

    else:
      wiimote = cwiid.Wiimote()
    ok=True
  except RuntimeError, msg:
    reintentos+=1;

#Si se superan los reintentos, terminar
if reintentos==3:
  print "Sin conexion con wiimote", reintentos
  sys.exit(-1)

#Conexion establecida
#Encender el led de la wii-board
wiimote.led = cwiid.LED1_ON

#Activar la wii-board
wiimote.rpt_mode = cwiid.RPT_BALANCE

print "CONEXION ESTABLECIDA"

#Obtener los valores de calibracion, usados para calcular
#los valores normalizados de los sensores de presion
balance_calibration = wiimote.get_balance_cal()

calibration = {'right_top'   : balance_calibration[0],
               'right_bottom': balance_calibration[1],
               'left_top'    : balance_calibration[2],
               'left_bottom' : balance_calibration[3]   }
```

```
pygame.init()

size = width, height = 800, 511
black = 0, 0, 0

white = 255,255,255
color1 = 196, 160, 15
font = pygame.font.Font(None, 25)

screen = pygame.display.set_mode(size)

pygame.display.set_caption("Grawiity_center")

#Cargamos la imagen de la wiiboard
fondo = pygame.image.load("wiiboard-top-view-1.jpg")
fondorect = fondo.get_rect()

time.sleep(0.2)

#Variable para mostrar/ocultar el peso
view_peso = True

try:
    while 1:
        #Leer los valores nomalizados de los sensores
        lt,rt,lb,rb = leer_sensores(wiimote,calibration)

        #Calcular el peso en kilos
        peso = (lt+rt+lb+rb)/100.0

        #Se usa un umbral de peso para determinar si hay
        #alguien subido. El peso debe ser mayor de 2Kg
        if peso>2:
```

```
    #Calcular las posiciones x,y del centro de gravedad
    try:
      x_balance = (rt+rb) / (lt+lb)
      if x_balance > 1:
        x_balance = -1.0*(lt+lb) / (rt+rb) + 1.0

      else:
        x_balance = x_balance -1.0

      y_balance = (lb+rb) / (lt+rt)
      if y_balance > 1:
        y_balance = -1.0*(lt+rt)/(lb+rb) +1.0
      else:
        y_balance = y_balance -1.0
    except:
       x_balance = 1.0
       y_balance = 1.0

#Peso menor de dos kilos: no hay nadie subido
else:
  x_balance=0.0;
  y_balance=0.0;

#Escalar las coordenadas y establecer el criterio de signos
xpos = 400*x_balance + 400
ypos = 248*y_balance + 248

#Comprobar si programa terminado
for event in pygame.event.get():
  if event.type == pygame.QUIT:
    wiimote.close()
```

```
            sys.exit()
          elif event.type == pygame.KEYDOWN:
            if view_peso:
              view_peso=False
            else:
              view_peso=True

        screen.blit(fondo, fondorect)
        text_bateria =
font.render(str(estado_bateria(wiimote.state)), True, black)
        screen.blit(text_bateria, (150,485))

        if view_peso:
          text_peso = font.render("Peso: %d" % peso, True, black)
          screen.blit(text_peso, (500,485))

        #Estado sensores de peso
        text_lt = font.render("%d" % (lt/100),True,color1)
        screen.blit(text_lt, (160,140))

        text_lb = font.render("%d" % (lb/100),True,color1)
        screen.blit(text_lb, (160,350))

        text_rt = font.render("%d" % (rt/100),True,color1)
        screen.blit(text_rt, (640,140))

        text_rb = font.render("%d" % (rb/100),True,color1)
        screen.blit(text_rb, (640,350))

        #Circulo para indicar la posicion del centro de gravedad
        pygame.draw.circle(screen, (255,0,0), (int(xpos),
int(ypos)), 10)
```

```
        pygame.display.flip()

#Terminar con Control-C
except KeyboardInterrupt:
  wiimote.close()
  print
  print "-- FIN --"
```

7. PONG_WIIBOARD

```
#------------------------------------------------------------------

#--  pong_wiiboard.py

#--  Clasico juego pong adaptado para que el jugador use la
wiiboard como interfaz de control

#------------------------------------------------------------------

#-- Carlos Pastor

#------------------------------------------------------------------


# Importar modulos

import pygame

from pygame.locals import Rect, K_ESCAPE, QUIT, KEYDOWN, KEYUP,
K_UP, K_DOWN, K_q, K_a

import sys

import math

import cwiid

import time


#Variables

SCREEN_DIMENSIONS = (800, 600)

screen = None

clock = None

paddle1 = None

paddle2 = None

ball = None


#Datos de WiiBoard

reintentos=0

ok=False

yPrevi,yPrevd=0,0
```

```
print 'Pulsa el boton de sync de la wiiboard (donde estan las
pilas)...'


#Establecer conexion con la WiiBoard

while reintentos<3 and not ok:

      try:

          print "Reintento ",reintentos

            if len(sys.argv) > 1:

              wiimote = cwiid.Wiimote(sys.argv[1])

          else:

                  wiimote = cwiid.Wiimote()

            ok=True

        except RuntimeError, msg:

          reintentos+=1;


      #Si se superan los reintentos, terminar

      if reintentos==3:

        print "Sin conexion con wiimote despues de
"+repr(reintentos)+" reintentos"

        sys.exit(1)


      #Encender el led 1 de la WiiBoard

      wiimote.led = cwiid.LED1_ON


      #Activar la WiiBoard

      wiimote.rpt_mode = cwiid.RPT_BALANCE


#Rutina de inicio

def main():

    global screen

    global clock

    global SCREEN_DIMENSIONS
```

```
#Inicializando todo lo relacionado con Pygame
    pygame.init()
    screen = pygame.display.set_mode( SCREEN_DIMENSIONS )
    pygame.display.set_caption("Pong_wiiboard")
    clock = pygame.time.Clock()
    initData()
    mainLoop()

#Inicializacion de estructuras de datos
def initData():
    global paddle1
    global paddle2
    global ball

    paddle1 = Paddle(30, 300)
    paddle2 = Paddle(770, 300)
    ball = Ball(400, 300, 3, 3)

#Bucle principal
def mainLoop():
    global clock

    while True:
      renderScreen()
      readInput()
      updateState()
      clock.tick(120)

#Funcion que dibuja la pantalla
def renderScreen():
```

```
    global screen


    s = screen

    s.lock()

    s.fill(0)

    s.fill(0xffffff, Rect(ball.left(), ball.top(), ball.width,
ball.height))

    s.fill(0xccccff, Rect(paddle1.left(), paddle1.top(),
paddle1.width, paddle1.height))

    s.fill(0xccccff, Rect(paddle2.left(), paddle2.top(),
paddle2.width, paddle2.height))

    s.unlock()

    pygame.display.update()


#Lectura de la entrada de teclado y eventos del wiiboard

def readInput():

   global paddle1

   global paddle2

   global yPrevi

   global yPrevd


   balance_calibration = wiimote.get_balance_cal()

   calibration = {'right_top'   : balance_calibration[0],

                  'right_bottom': balance_calibration[1],

                  'left_top'    : balance_calibration[2],

                   'left_bottom' : balance_calibration[3]   }


   view_peso = True


   #Leer los valores nomalizados de los sensores

   lt,rt,lb,rb = leer_sensores(wiimote,calibration)


   #Calcular el peso en kilos
```

```
peso = (lt+rt+lb+rb)/100.0

#Se usa un umbral de peso para determinar si hay
#alguien subido. El peso debe ser mayor de 2Kg
if peso>2:
   #Calcular las posiciones x,y del centro de gravedad
     try:
          x_balance = (rt+rb) / (lt+lb)
          if x_balance > 1:
            x_balance = -1.0*(lt+lb) / (rt+rb) + 1.0
          else:
            x_balance = x_balance -1.0

          y_balance = (lb+rb) / (lt+rt)
          if y_balance > 1:
            y_balance = -1.0*(lt+rt)/(lb+rb) +1.0
          else:
            y_balance = y_balance -1.0
     except:
           x_balance = 1.0
           y_balance = 1.0

 #Peso menor de dos kilos: no hay nadie subido
else:
      x_balance=0.0;
      y_balance=0.0;

paddle1.dy=10*abs(y_balance)
paddle2.dy=10*abs(y_balance)

for event in pygame.event.get():
```

```
   if event.type == QUIT:

      quit()


if x_balance<0:
      paddle2.joystick.up = False
      paddle2.joystick.down = False

      if y_balance>0:
         paddle1.joystick.up = False
         paddle1.joystick.down = True
      else:
         paddle1.joystick.up = True
         paddle1.joystick.down = False

      if abs(y_balance-yPrevi)<0.0009:
         paddle1.joystick.up = False
         paddle1.joystick.down = False

yPrevi=y_balance

if x_balance>0:
      paddle1.joystick.up = False
      paddle1.joystick.down = False

      if y_balance>0:
         paddle2.joystick.up = False
         paddle2.joystick.down = True
      else:
         paddle2.joystick.up = True
         paddle2.joystick.down = False
```

```
            if abs(y_balance-yPrevd)<0.0009:
                paddle2.joystick.up = False

                paddle2.joystick.down = False

    yPrevid=y_balance

#Actualizar estado del juego
def updateState():
    global paddle1
    global paddle2
    global ball

    if (paddle1.joystick.up):
        paddle1.moveUp()
    elif (paddle1.joystick.down):
        paddle1.moveDown()

    if (paddle2.joystick.up):
        paddle2.moveUp()
    elif (paddle2.joystick.down):
        paddle2.moveDown()

    ball.update(paddle1, paddle2)

#Leer los valores normalizados de los 4 sensores
def leer_sensores(wiimote,calibration):
  state = wiimote.state

  sensor = 'left_top'
  valor = state['balance'][sensor]
  valor0 = calibration[sensor][0]
```

```
  valor1 = calibration[sensor][1]
  a = 1700 * float((valor - valor0)) / float((valor1-valor0))

  sensor = 'right_top'
  valor = state['balance'][sensor]
  valor0 = calibration[sensor][0]
  valor1 = calibration[sensor][1]
  b = 1700 * float((valor - valor0)) / float((valor1-valor0))

  sensor = 'left_bottom'
  valor = state['balance'][sensor]
  valor0 = calibration[sensor][0]
  valor1 = calibration[sensor][1]
  c = 1700 * float((valor - valor0)) / float((valor1-valor0))

  sensor = 'right_bottom'
  valor = state['balance'][sensor]
  valor0 = calibration[sensor][0]
  valor1 = calibration[sensor][1]
  d = 1700 * float((valor - valor0)) / float((valor1-valor0))


  return (a,b,c,d)

#Clase joystick
class Joystick:
    up = False
    down = False

#Clase bola
class Ball:
    x = 0
```

```
y = 0
dx = 0
dy = 0

width = 10
height = 10

def __init__(self, x, y, dx, dy):
    self.x = x
    self.y = y
    self.dx = dx
    self.dy = dy

def top(self):
    return self.y - self.height/2

def left(self):
    return self.x - self.width/2

def update(self, paddle1, paddle2):
    #Actualizo posicion
    self.x += self.dx
    self.y += self.dy

    #Deteccion de colision con las paletas
    if paddle1.collides((self.x, self.y)):
        self.dx = -self.dx

    if paddle2.collides((self.x, self.y)):
        self.dx = -self.dx

    #Deteccion de colision con los bordes de la ventana
```

```
        if (self.x < 0):
            self.x = 1
            self.dx = -self.dx

        if (self.x > SCREEN_DIMENSIONS[0]):
            self.x = SCREEN_DIMENSIONS[0]-1
            self.dx = -self.dx

        if (self.y < 0):
            self.y = 1
            self.dy = -self.dy

        if (self.y > SCREEN_DIMENSIONS[1]):
            self.y = SCREEN_DIMENSIONS[1]-1
            self.dy = -self.dy

#Clase paleta
class Paddle:
    x = 0
    y = 0

    #Velocidad vertical de la paleta
    dy = 0
    width = 10
    height = 120
    joystick = None

    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.joystick = Joystick()
```

```
    def top(self):
        return self.y - self.height/2

    def bottom(self):
        return self.y + self.height/2

    def left(self):
        return self.x - self.width/2
    def right(self):
        return self.x + self.width/2

    def moveUp(self):
        self.y -= self.dy
        self.y = max([self.height/2, self.y])

    def moveDown(self):
        self.y += self.dy
        self.y = min([self.y, SCREEN_DIMENSIONS[1]-self.height/2])

    def collides(self, point):
        return (point[0] >= self.left() and point[0] <=
self.right()) and (point[1] >= self.top() and point[1] <=
self.bottom())

#Inicio del juego
main()
```

Anexo D

Instalación de la librería para Wii Balance Board

Para la programación de aplicaciones usando la Wii Balance Board se ha de instalar una versión específica de la librería libcwiid. Esto se debe a un error producido tras la inclusión en la librería del módulo encargado de manejar el Wii Motion Plus.

Los pasos a seguir para la instalación son los siguientes:

1. Instalar los paquetes necesarios para la instalación de las librerías (en caso de no estar instalados).

   ```
   sudo aptitude install autoconf autogen gcc bluetooth
   libbluetooth3-dev libgtk2.0-dev pkg-config python-dev flex
   bison

   sudo aptitude install bluez libbluetooth-dev
   ```

2. Descargar las librerías del Wiimote y WiiBoard (revisión 183) de la dirección http://users.prevenciokft.hu/sam/files/libcwiid/libcwiid-0.60.00-2.tar.gz y descomprimirlo.
3. Descargar el parche para la WiiBoard balance.diff de la dirección http://abstrakraft.org/cwiid/raw-attachment/ticket/63/balance.diff en la ruta ./libcwiid/src.

4. Ejecutar las siguientes instrucciones:

```
$HOME/libcwiid/src$ aclocal

$HOME/libcwiid/src$ autoconf

$HOME/libcwiid/src$ ./configure –libdir=/usr/lib

$HOME/libcwiid/src$ make

$HOME/libcwiid/src$ patch –b –p0 < balance.diff

$HOME/libcwiid/src$ make

$HOME/libcwiid/src$ sudo make install

$HOME/libcwiid/src/python/build/lib.linux-i686-2.6$ sudo cp
./cwiid.so /usr/local/lib/python2.6/dist-packages
```

Anexo E

Manual de operaciones de Wii Balance Board

A continuación se presenta las páginas 32 y 33 del manual de operaciones de Wii Balance Board.

Cómo Sincronizar el Accesorio de Wii Balance Board con su consola Wii

5. Oprima el Botón de SYNCHRO (SINC.) en el accesorio de Wii Balance Board. Las LED de Energía en la tabla comenzarán a parpadear (hasta por alrededor de 20 segundos).

6. Oprima el botón de SYNCHRO (SINC.) en la consola **mientras que las LEDs de Energía de la tabla estén parpadeando.**

IMPORTANTE: No oprima y mantenga oprimiendo el Botón de SYNCHRO de la consola Wii por más de 10 segundos. Esto borrará toda la información de Sincronización Estándar para todos los Controles Remoto Wii que haya usado con esa consola.

7. Las LED de Energía dejarán de parpadear y se mantendrán encendidas una vez que la sincronización se haya completado. Ahora ya está listo para jugar con un Disco de Juegos compatible con el Wii Balance Board. Por favor consulte al manual de instrucciones del Disco de Juegos que esté usando para información adicional del juego.

NOTA:

- **Solamente se puede sincronizar un accesorio de Wii Balance Board con una consola Wii a la vez. Un total de 10 Controles Remoto Wii y accesorios pueden estar sincronizados a una consola Wii en cualquier momento utilizando el procedimiento de Sincronización Estándar. Si se ejecutan sincronizaciones adicionales, esta sincronización sobreescribirá a la sincronización más vieja.**
- **El accesorio de Wii Balance Board solamente puede ser sincronizado a una sola consola a la vez siguiendo el procedimiento de Sincronización Estándar. Si cambia la consola con la cual esté usando el accesorio de Wii Balance Board, tendrá que repetir el procedimiento de sincronización cada vez que cambie la consola.**
- **El accesorio de Wii Balance Board utiliza la conexión del Jugador 4 (J4). Si un Control Remoto Wii está usando la conexión del J4, ese control remoto se desconectará mientras que se esté usando la tabla.**

Operaciones Básicas

La mayoría de las actividades básicas que se ejecutan con el accesorio de Wii Balance Board se llevan a cabo al usar la posiciones que se enseñan abajo. Puede que existan otras maneras de juego dependiendo en el tipo de juego que esté jugando. Revise el folleto de instrucciones para el juego que esté jugando para información adicional.

También consulte la Información sobre Salud y Seguridad en la página 26 de éste manual y la Información sobre Salud y Seguridad e instrucciones de operación en el Manual De Operaciones del Wii – Configuración del Sistema que viene incluido con su sistema Wii.

Bibliografía

[1] Asibot en RoboticsLab

http://roboticslab.uc3m.es/roboticslab/robot.php?id_robot=3

[2] Campo de acción de la rehabilitación

http://www.neurorehabilitacion.com/areascampo.htm

[3] Medicina física y rehabilitación

http://es.wikipedia.org/wiki/Medicina_f%C3%ADsica_y_rehabilitaci%C3%B3n

[4] Mecanoterapia

http://www.sld.cu/galerias/pdf/sitios/rehabilitacion/mecanoterapia.pdf

[5] Departamento de terapia ocupacional

http://www.neurorehabilitacion.com/areasterapiaocupacional.htm

[6] Fisioterapia

http://es.wikipedia.org/wiki/Fisioterapia

[7] Equipa BUAP laboratorios de fisioterapia con tecnología de punta

http://cmas.siu.buap.mx/portal_pprd/wb/comunic/equipa_buap_laboratorios_de_fisioterapia_con_1318

[8] Un equipo multidisciplinar español en el Journal of Neuroengineering and Rehabilitation

http://rehabilitacionymedicinafisica.blogspot.com/2010/08/un-equipo-multidisciplinar-espanol-en.html

[9] Silla de ruedas que se controla con la nariz para personas severamente discapacitadas

http://www.medgadget.com/spanish/2010/07/silla_de_ruedas_que_se_control_1.html

[10] Rehabilitación: Tecnología robótica vence la parálisis

http://axxon.com.ar/not/138/c-1380036.htm

[11] Lokomat

http://www.samarit.com/pdf/hocoma_lokomat_s.pdf

[12] SEGA anuncia Sega Zone su nuevo dispositivo de juegos

http://www.3djuegos.com/noticias-ver/109590/sega-anuncia-sega-zone-su-nuevo-dispositivo-de-juegos/

[13] Playstation Move

http://en.wikipedia.org/wiki/PlayStation_Move

[14] Playstation Eye

http://en.wikipedia.org/wiki/PlayStation_Eye

[15] Playstation move, ¿podría aplicarse a rehabilitación?

http://rehabilitacionymedicinafisica.blogspot.com/2010/06/play-station-move-podra-aplicarse.html

[16] Kinect

http://es.wikipedia.org/wiki/Kinect

[17] Kinect

http://en.wikipedia.org/wiki/Kinect

[18] Más Allá del 2000 – Natural User Interface

http://expressionlab.net/tag/interfaz-natural-de-usuario/

[19] Wii

http://es.wikipedia.org/wiki/Wii

[20] Wii-Habilitation

http://www.wiihabilitation.co.uk/home.shtml

[21] motej

http://motej.sourceforge.net/

[22] Wiimote libraries

http://wiibrew.org/wiki/Wiimote/Library

[23] WiiYourself!

http://wiiyourself.gl.tter.org/

[24] Wiim

http://digitalretrograde.com/projects/wiim/

[25] Wiiuse

http://www.wiiuse.net/

[26] Cwiid

http://abstrakraft.org/cwiid/

[27] Python

http://es.wikipedia.org/wiki/Python

[28] Pygame

http://www.pygame.org/wiki/about

[29] Cómo funciona el Wiimote.

http://www.taringa.net/posts/info/1843549/Cómo-funciona-el-Wiimote.html

[30] Extension Controllers

http://wiibrew.org/wiki/Wiimote/Extension_Controllers

[31] API libcwiid

http://abstrakraft.org/cwiid/wiki/libcwiid

[32] Pong

http://es.wikipedia.org/wiki/Pong

[33] Wiimote

http://wiibrew.org/wiki/Wiimote

[34] giimote. A Wii Remote extension for Game Maker and MATLAB

http://code.google.com/p/giimote/wiki/Camera

[35] Johnny Chung Lee > Projects > Wii

http://johnnylee.net/projects/wii/

[36] Especial Wii Motion Plus
http://www.meristation.com/v3/des_articulo.php?id=cw4a3e9e107070c&pic=GEN#

[37] Iwata pregunta: Wii Motion Plus

http://es.wii.com/wii/es_ES/software/iwata_pregunta_wii_motionplus_volume_1_2162.html

[38] Como funciona Wii

http://laconchadetuhermana.taringa.net/posts/apuntes-y-monografias/4682115/como-funciona-el-wii.html

[39] Iwata pregunta: Wii Fit

http://latam.wii.com/wii-fit/iwata_asks/vol2_page1.jsp

[40] El Step

http://www.hispagimnasios.com/a_culturismo/step.php

[41] Manual de operaciones Wii Balance Board

Anexo E