Universidad de las Ciencias Informáticas

Facultad 5

Biblioteca multiplataforma para la integración del Wiimote.

Trabajo para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Antonio David Rubán Espinal.

Tutor: Ing. Orlay García Ducungé.

Ciudad de La Habana

Junio, 2011

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año _____.

_____	_____
Antonio David Rubán Espinal	Ing. Orlay García Ducungé.
Autor	Tutor.

Datos del Contacto

Tutor: Orlay García Ducungé**.**

Edad: 26 años.

Ciudadanía: cubana**.**

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas**.**

Categoría: Docente**.**

E-mail: oducunge@uci.cu.

Graduado en Ingeniería en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas (UCI), actualmente profesor de esta misma escuela, 5 años con experiencia en Gráfico por Computadoras, actualmente es el arquitecto del proyecto Laboratorios Virtuales en la Universidad de las Ciencias Informáticas.

A mi dos madres Ana Luisa y Juana, por estar siempre presente, por ser mis principal fuente de inspiración para avanzar en la vida. Por ser mis primeras maestras de la vida y por hacerme un hombre de bien. Gracias mis viejas.

A mi padre Rolando David, por ser mi modelo de hombre y ser mi principal orgullo. Gracias papá.

A mi abuelo Guancho, por ser ejemplo de modestia y desinterés personal, por ser mi paradigma de hombre incansable y trabajador. Gracias abue.

A mi linda hermanita Rosi, por ser mi segundo mayor orgullo, aunque el mayor tiempo de nuestras vidas nos la hemos pasado lejos, siempre te querré y te llevaré muy dentro de mí.

A mis tías queridas, por estar siempre pendientes de mí en los momentos buenos y malos de mi vida, por enseñarme el significado de la palabra familia. Muchas gracias.

A mi novia Solangel, por estar a mi lado en los momentos más difíciles. Por entregarme tu amor y cariño incondicional en todo momento posible.

A mi madrina Mireya, gracias por tu atención y dedicación.

A Caridad, gracias por aceptarme tal y como soy, por mostrarme el camino cuando ya no había luz.

A los integrantes del "Club", por ser una gran familia todo el tiempo, por señalar mis defectos y resaltar mis virtudes.

A mi tutor Orlay, por enseñarme el camino correcto para la realización del presente trabajo.

A mis padres, por su amor, comprensión y dedicación en todo momento.

A mis abuelos, por apoyarme toda mi vida.

A mi hermana Rosalba, gracias por estar siempre unidos en los buenos y malos tiempos.

A mi familia, gracias por darme el apoyo y la compresión necesaria para cumplir con mi sueño de convertirme en ingeniero.

A mi novia, por su amor incondicional.

A todas las personas que se preocuparon por el desarrollo de esta tesis.

Resumen

El mando de la videoconsola *Wii,* conocido como *Wiimote,* ha revolucionado el mundo de los videojuegos. Se trata de un dispositivo con una complejidad técnica considerable, pero con unas características muy especiales y novedosas que convierte su uso en algo intuitivo y llamativo.

Teniendo en cuenta las virtudes del *Wiimote* se ha considerado incluirlo como un elemento más en el desarrollo de aplicaciones gráficas, por lo que surge la necesidad de integrar este dispositivo con las aplicaciones gráficas que se están desarrollando en la Facultad 5 sobre los sistemas operativos *Windows* y *GNU/Linux*.

En el presente trabajo se propone desarrollar una biblioteca multiplataforma que brinde las funcionalidades básicas que se pueden obtener del *Wiimote* para así poder integrarlo con las aplicaciones gráficas que se desarrollan en el Centro de Informática Industrial de la Universidad de las Ciencias Informáticas (CEDIN), facilitando al programador el proceso de desarrollo de estas aplicaciones gráficas permitiendo que estas aplicaciones integren la tecnología del *Wiimote*.

Palabras Clave

Biblioteca, Multiplataforma, Wiimote.

Índice de Contenidos.

Índice de Figuras.

Índice de Tablas.

Introducción.

Desde su lanzamiento en noviembre del 2006, la videoconsola de sobremesa *Wii* de *Nintendo* ha sido una de las principales videoconsolas de séptima generación más utilizadas y distribuidas en el mundo junto al *Playstation 3* de *Sony* y *Xbox 360* de *Microsoft* [37] (ver Figura 1). Esta videoconsola, lejos de sostenerse únicamente en la pulsación de botones para el control de sus aplicaciones, también está basada en los movimientos del usuario, lo que permite una nueva forma de interactuar con el sistema, peculiaridad gracias a la cual ha tenido gran aceptación en la población mundial. Una de las características más atractivas de la videoconsola es su mando inalámbrico, el *Control Remote Wii* o *Wiimote* (ver Figura 9) como se le conoce, dicho mando ha ganado la popularidad rápidamente debido a la innovadora forma de interactuar con los videojuegos, llevando el tema de los videojuegos a un nivel superior, dicho mando es responsable de detectar los movimientos del usuario y gracias a su conectividad *Bluetooth* es posible utilizarlo como dispositivo de entrada para aplicaciones de PC.

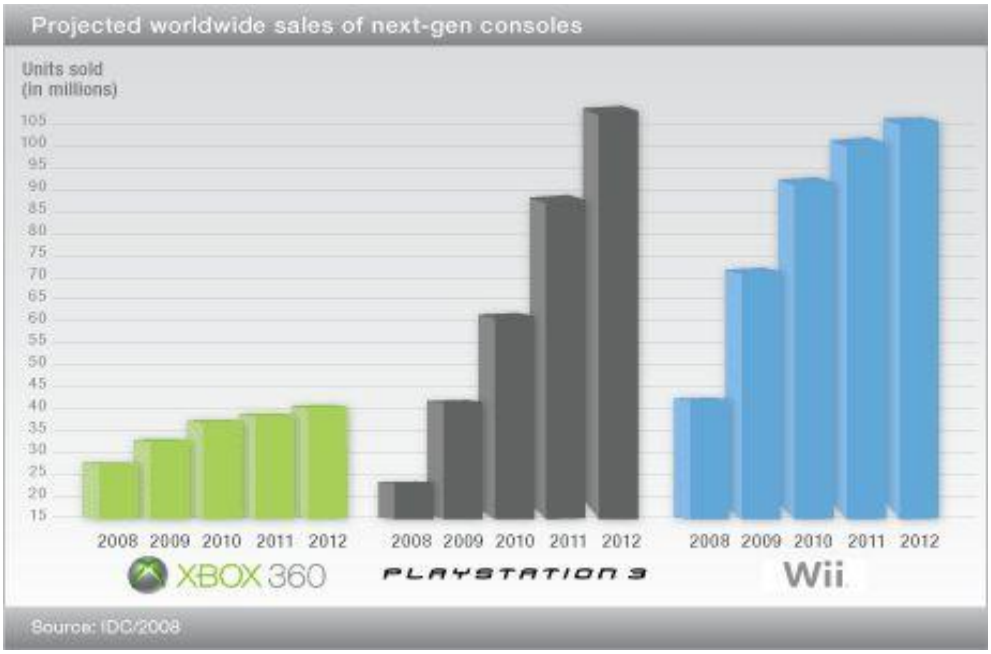

Figura 1. Videoconsolas más distribuidas.

En la actualidad se está haciendo uso de las bondades que brinda el *Wiimote* el cual no solo sirve para jugar con la consola, sino que tiene muchas aplicaciones y puede ser de ayuda en muchos aspectos del

día a día, por ejemplo el *Wiimote* nos ofrece la ventaja de poder controlar la televisión o la computadora de forma muy natural, gestual e intuitiva, otros de los campos que se pueden beneficiar de las virtudes de este dispositivo es el de la Medicina, especialmente en el área de la rehabilitación, entrenadores profesionales y otros tipos de aplicaciones gráficas.

Nuestro país no está ajeno a la utilización de estos tipos de tecnologías, en la Universidad de las Ciencias Informáticas (UCI), específicamente en la Facultad 5, se han realizado trabajos investigativos y de producción dirigido al uso del *Wiimote***.**

El Centro de Informática Industrial de la Universidad de las Ciencias Informáticas (CEDIN) adquirió a través de colaboraciones con universidades extranjeras la tecnología *Wii* pero no se puede integrar a las aplicaciones gráficas que se desarrollan, ya que estas son multiplataforma y no se dispone de una biblioteca multiplataforma escrita en el lenguaje C++ que permita integrar el *Wiimote*; esta necesidad conllevará a que las aplicaciones gráficas que se desarrollan en el centro están en desventaja con aplicaciones similares a nivel internacional.

Por lo que dada la situación existente surge como **problema científico** a resolver ¿Cómo lograr la integración del *Wiimote* con las aplicaciones gráficas en los sistemas operativos *Windows* y *GNU/Linux*?

A partir del problema científico se enmarca como **objeto de investigación**, el proceso de integración del *Wiimote*.

El **objetivo general de investigación** está dirigido a elaborar una biblioteca para integrar el *Wiimote* a las aplicaciones gráficas en *Windows* y *GNU*/*Linux*.

Se delimita como **campo de acción,** los procesos de integración del *Wiimote* para el desarrollo de aplicaciones gráficas en el campo de la Realidad Virtual.

Dado el objetivo planteado anteriormente se hace necesario trazar las siguientes **tareas investigativas:**

- ✓ Elaboración del marco teórico de la investigación a partir del estado del arte existente actualmente sobre el tema.
- ✓ Análisis de las bibliotecas existentes que son utilizadas para integrar el *Wiimote*, para realizar una selección de las que respondan a las necesidades del sistema.
- ✓ Selección de las herramientas y bibliotecas de integración a utilizar, para la implementación del sistema.
- ✓ Diseño y modelado arquitectónico de las funcionalidades definidas para lograr una arquitectura que se adapte a las necesidades del sistema.

- ✓ Implementación de las soluciones técnicas para alcanzar los objetivos propuestos.

- ✓ Validación del sistema elaborado, para encontrar deficiencias en la aplicación.

Con la utilización de la biblioteca propuesta se podrá integrar el *Wiimote* con las aplicaciones gráficas que se desarrollan en el centro en diferentes sistemas operativos ya sea *Windows* o *GNU/Linux*.

Para el desarrollo de esta investigación de emplearon diferentes **métodos científicos** tanto **teórico** como **empíricos** los cuales quedan planteado a continuación**:**

Métodos Teóricos:

Analítico-Sintético: Con el uso de este método de investigación se analizarán las informaciones obtenidas y se hará un desglose de las mismas, para después sintetizarlas y arribar a ideas esenciales.

Histórico-Lógico: Este método teórico se utilizará en la investigación para el análisis de la trayectoria, la evolución y el desarrollo del *Wiimote* en aplicaciones gráficas.

Métodos Empíricos:

Consultas de las fuentes de información: Para seleccionar la información necesaria, que permitirá construir el marco teórico.

Consultas de especialistas: Para recibir los criterios de validación sobre la aplicabilidad y utilización de lo realizado.

Observación: Se utilizará para reunir información visual de como la solución propuesta se comporta y cómo influirá en el rendimiento de la computadora.

Fundamentación Teórica

En el presente capítulo se analizará algunos conceptos y características de las bibliotecas para la integración del *Wiimote*. Se introducirá las principales propiedades del *Wiimote* las cuales lo han convertido en un dispositivo útil para el desarrollo de aplicaciones en todas las esferas sociales, también se expondrán algunas de sus aplicaciones en las diferentes ramas, como la Medicina y la Educación.

1.1. Evolución de Nintendo.

Nintendo Company Limited es una empresa multinacional de videojuegos. Fue fundada el 23 de septiembre de 1889 por Fusajiro Yamauchi como fabricante de barajas hanafuda (tradicionales naipes japoneses). Desde 1975 se ha dedicado a la producción de software y hardware para videojuegos, creciendo progresivamente hasta convertirse en la compañía más exitosa en la industria de los videojuegos.

Videoconsolas de sobremesa.

A finales de los 70, tras la proliferación de los ordenadores domésticos, la industria del videojuego entró en una importante crisis, puesto que los jugadores preferían piratear o comprar los juegos que habían en las zonas recreativas, en vez de gastarse el dinero en ellas (en aquella época la forma tradicional de jugar era asistiendo a centros recreativos).

La solución al problema fue la creación por parte de las empresas de sus propias consolas caseras, o desarrollar software para éstas. En ese contexto nace la ***Nintendo Entertainment System*** o **NES** una consola de 8-bits lanzada por *Nintendo* en 1983, esta fue la videoconsola más exitosa de su tiempo en Asia y América del Norte, convirtiéndose en la primera videoconsola que dio ganancias a sus fabricantes. Según datos de Nintendo la videoconsola vendió más de 60 millones de unidades a nivel mundial [34], sin contar periféricos y juegos que se vendieron posteriormente.

Figura 2. NES con su controlador.

Super Nintendo Entertainment System (SNES) (1991-1998).

La *SNES* fue la segunda videoconsola hogareña de *Nintendo.* Mientras que en un principio tuvo que luchar en las regiones de sistema *PAL* y gran parte de ASIA, la *SNES* demostró ser un éxito mundialmente rotundo, en parte gracias a que tenía millones de seguidores desde la época de *NES*. *Nintendo* vendió el doble de videoconsolas *Super Nintendo* que sus competidoras convirtiéndose así en la videoconsola más exitosa de la era de los 16-bits, vendiendo más de 49 millones de unidades a nivel mundial [35].

Figura 3. Super Nintendo Entertaiment System (versión norteamericana).

La era 3D y la nueva generación de consolas.

En 1995, *Nintendo* lanzó *Vitrual Boy*, marcando así el primer fracaso de la compañía, si bien en si era innovador tenía tantas contras que le impidieron el éxito. Entre las que se destacan por ejemplo a mucha gente le resultaba incómodo los juegos en rojo y negro y a más de uno le causaba mareos, además se encontraba constantemente con obstáculos mercantiles. Llegaron a lanzar solo 22 títulos, sin embargo la consola es digna de ser apreciada, contaba con un procesador de 32-bits y realmente se podía ver los juegos 3D con excelentes efectos [36].

Figura 4. Virtual Boy.

Nintendo 64 (N64) (1996 - 2002)

Tras el rotundo éxito de la SNES, *Nintendo* lanzó una consola de 64-bits al mercado, *Nintendo 64* en 1996 con Super Mario 64 como juego estrella. Si bien la *Nintendo 64* fue tecnológicamente superior en casi todos los aspectos a sus competidoras por su calidad técnica y unos accesorios innovadores (que hasta el día de hoy son fundamentales en cualquier videoconsola) como los mandos con vibración, sticks de un dedo para juegos en entornos tridimensionales y 4 puertos de mandos para que pudiesen jugar 4 jugadores [35].

Figura 5. Nintendo 64.

Nintendo GameCube (GCN) (2001 - 2008)

En noviembre del 2001 se lanza la *Nintendo GameCube*, una consola de nueva generación que tendría que rivalizar con *PlayStation* 2 de *Sony* y *Xbox* de *Microsoft.* Buscando la lucha por la piratería, *Nintendo* desarrolla el GOD (*GameCube Optical Disc,* un formato de MiniDVD exclusivo para *Nintendo* con una capacidad aproximada de 1.8 *GygaBytes* de datos) para sus juegos. *GameCube* fue junto a la videoconsola de *Microsoft,* la más potente de su momento y tuvo su cuota de mercado en torno al 20%, parecido al de *Xbox*, pero muy inferior a *PlayStation 2.* La cantidad de videoconsolas cubicas vendidas a nivel mundial es acerca de 22 millones, siendo superada por *Xbox* y *PlayStation 2. Nintendo* fue demasiado conservadora con esta consola porque su ideología es basada en hacer consolas para jugar solamente, por lo que eliminó las capacidades de multimedia (no reproducía ni películas ni música, a diferencia de sus competidoras).

Figura 6. Nintendo GameCube.

Wii (2006 – Fecha de descontinuación aún no determinada).

Wii es el nombre de la videoconsola de séptima generación de *Nintendo*, que es la sucesora de *Nintendo GameCube.* Fue conocida anteriormente con el nombre clave *Revolution,* el cual según *Nintendo* dictaba "la dirección en la que la compañía se dirigía". Gráficamente inferior a las consolas de su generación, cabe destacar que *Wii* ha sido la primera videoconsola de *Nintendo* que no ha llevado la palabra "Nintendo" en el nombre. Su principal característica es el mando inalámbrico bautizado como *Control Remote Wii,* desde su lanzamiento hasta la actualidad (2011), *Wii* ha sido la videoconsola más vendida de su generación haciendo que *Nintendo* se mantenga el lo alto del podio hasta la fecha [35].

Figura 7. Nintendo Wii.

1.2. Wiimote.

Wii Remote (comúnmente conocido como, *Control Remoto Wii* en Latinoamérica, *Mando* de *Wii* en España o *Wiimote* en el mundo *gamer)* es el mando principal de la videoconsola de sobremesa *Wii* producida y distribuida por *Nintendo* muy similar al control remoto de televisión, que además de los botones que posee, contiene un acelerómetro en los tres ejes de posición, una cámara de infrarrojos de alta resolución, un altavoz, un motor de vibración, cuatro luces tipo *LED* y conectividad inalámbrica vía *bluetooth*.

El *Wiimote* no siempre fue como el que conocemos hoy en día.

Figura 8. Predecesores del Wiimote.

1.3. Análisis del Wiimote.

En esta sección se describen de forma breve los componentes (ver Figura 10) y procedimientos principales involucrados en el funcionamiento del *Wiimote*. A continuación se expone una vista global de las características del dispositivo. Para ver más detalles referirse a [20].

Figura 9. Wiimote actual.

El *Wiimote* está formado por los siguientes elementos:

- **Botones** (POWER, HOME, A, -, +, 1, 2 y B como gatillo en la parte inferior).
- **Cruz direccional**.
- **4 *Leds***: indican el número de jugador o el nivel de batería.
- **2 pilas AA**: con una duración de entre 30 y 60 horas según el uso.
- **Acelerómetro ADLX330 de 3 ejes**: para detección de movimientos de usuarios.
- **Sensor óptico de infrarrojos**: para detección de movimientos de usuarios.
- **Puerto de expansión**: para la conexión con otros dispositivos como el Nunchuck (*Figura 13.A*).
- **Conexión Bluetooth**: para comunicación entre la videoconsola o PC y el propio Wiimote.
- **Altavoz de baja calidad:** Puede reproducir efectos sonoros.
- **Motor de vibración**: Similar a los motores de vibración de los celulares, el cual vibrará en caso de que ocurra un acontecimiento.
- **Chip de memoria EPROM de 16 Kib**: destinado a guardar información del juego de ejecución y hasta 10 perfiles de usuarios (denominados *Mii*) [1].

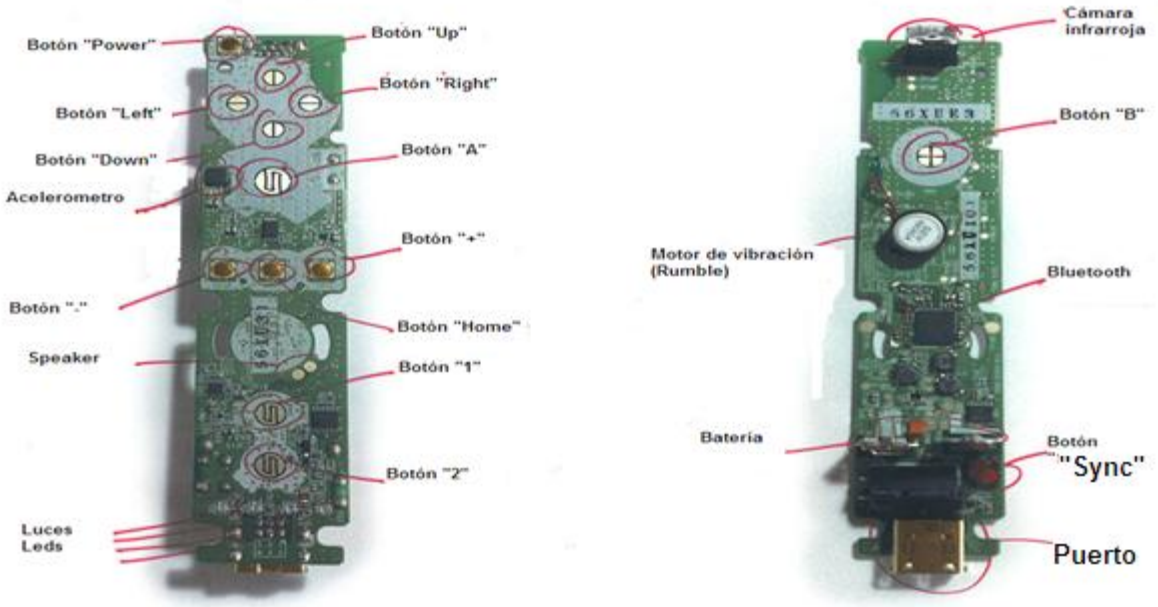

Figura 10. Componentes del Wiimote.

1.3.1. Funcionamiento.

El *Wiimote* se basa aparte de la clásica pulsación de botones, en los movimientos del usuario para el control de las aplicaciones y los juegos de la videoconsola *Wii*. Estos movimientos se detectan utilizando dos elementos del dispositivo.

- Acelerómetro de 3 ejes.
- Sensor de infrarrojos.

El primero de ellos mide la inclinación del mando en 3 ejes y transmite a la videoconsola o la PC a través del enlace *Bluetooth*. Esta forma de detección de movimiento se usa por ejemplo en juegos de conducción.

El segundo elemento, el sensor de infrarrojos, es similar al de la cámara de visión infrarroja y se utiliza para detectar hacia donde apunta el usuario con el mando de manera que use el mando como puntero para desplazarse por pantallas e interactuar con juegos de puntería.

Dado que el *Wiimote* utiliza *Bluetooth* para la transmisión de datos, es una de las principales ventajas de utilizar este dispositivo, ya que permite la comunicación con la computadora o cualquier dispositivo compatible a esta tecnología.

1.3.2. Accesorios.

Además del propio *Wiimote*, se pueden utilizar accesorios conjuntamente con él, u otros de forma independiente, cabe destacar que estos dispositivos son bastantes novedosos y compatibles con la computadora.

Nunchuck.

Este periférico se conecta por la parte inferior del *Wiimote*, provee de cuatro elementos para el control de los juegos y de aplicaciones de *Wii* ver en la *figura 13 A*.

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

Classic Controller.

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

Figura 11. Wii Classic Controller.

Guitarra.

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

La tercera entrega del juego fue lanzada entre otras videoconsolas para *Wii*, lo que obligo a los desarrolladores a crear una guitarra específica para esta videoconsola. La guitarra para *Wii* está pensada para albergar en su interior al *Wiimote* y comunicarse con él a través del puerto de extensiones, dotando a la guitarra de un sensor de movimiento y vibración.

Figura 12. Guitarra para Wii.

1.4. Aplicaciones del Wiimote.

En la actualidad el *Wiimote* ha alcanzado gran reputación en el mundo, por lo que la comunidad investigadora ha propuesto este mando como solución a diferentes proyectos utilizando en parte sus capacidades.

Medicina.

El Centro Tecnológico para el Desarrollo de las Telecomunicaciones de Castilla y León (CEDETEL) [18] ha estado estudiando las posibles aplicaciones que se le puede dar al *Wiimote* en la esfera de la medicina principalmente en el área de la rehabilitación.

Se están haciendo estudios de investigación sobre la rehabilitación de los dedos y la muñeca donde gracias al pequeño tamaño del *Wiimote* se emplea para detectar los movimientos de la muñeca. Se ha desarrollado un robot conocido como *CRAMMER* que se observa en la *figura 13 B*, para ayudar a los pacientes a realizar movimientos de tres grados de libertad en la muñeca y el antebrazo.

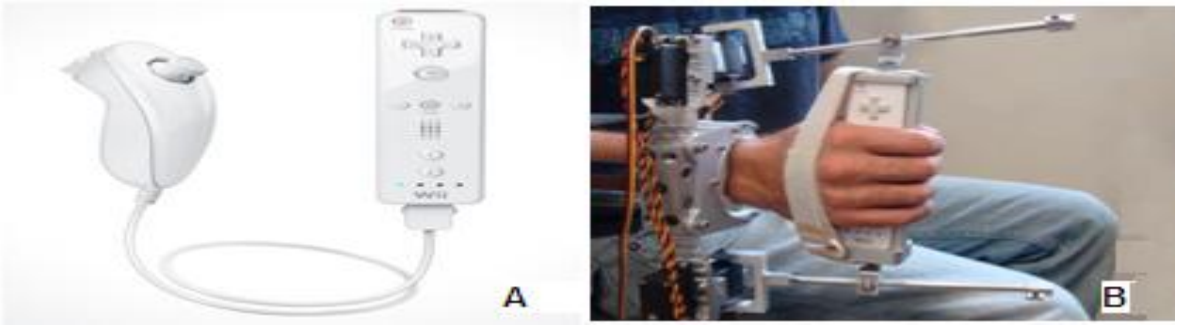

Figura 13. Wiimote y Nunchuck (A) y Robot CRAMMER (B).

Aunque solo se trata de pruebas piloto ya se están dando los primeros pasos para ayudar a las personas con problemas neurológicos, como la falta de concentración o la hiperactividad. También el CEDETEL está investigando como el *Wiimote* podría ayudar a las personas que padecen de fobia, donde se espera usar la cámara del *Wiimote* y unas gafas que va a llevar la persona en cuestión. Con este proyecto se espera tratar la fobia de forma segura y virtualmente simulando un entorno real que obliguen al paciente a enfrentarse a sus miedos sin poner en peligro su seguridad, por ejemplo una persona que le tendría miedo a las alturas ya no haría falta subirla a un quinto piso [18].

Educación.

En las aulas pueden acabar albergando un *Wiimote*, pues como está investigando el mismo CEDETEL la cámara del *Wiimote*, un lápiz infrarrojo, un data show y una computadora son suficientes para convertir una pizarra normal en una pizarra electrónica con todas las oportunidades que esta pueda ofrecer. Esto beneficiaria el trabajo docente grandemente debido a que la pizarra electrónica permite innovar en las prácticas docentes y mejorar la motivación y atención del alumno, ofreciendo nuevas herramientas para atender la diversidad de los escolares y en especial de aquellos con dificultades severas o moderadas en el aprendizaje [18].

Otros.

El entretenimiento es la principal función de la consola *Wii* de *Nintendo*, así que sus aplicaciones surgen de forma natural en este contexto. En el proyecto *Wiimedia* [19], se proponen algunas aplicaciones de juego de azar, empleando el *Wiimote* como sistema de control, estas aplicaciones son entre otras, juegos de carrera, de espada o software de dibujo. El *Wiimote* debido a su diseño y fácil integración con la computadora puede ser usado para navegar en la web, en la *figura 14* se demuestra que puede ser gran ayuda para personas discapacitadas.

Figura 14. Navegación web con el Wiimote.

Johnny Chung Lee es un desarrollador que también ha demostrado el gran potencial, utilidad y aplicabilidad del *Wiimote*, su trabajo se centra principalmente en dispositivos construidos mediantes emisores y cámaras de infrarrojos del *Wiimote*, ampliando así las posibilidades de utilización. Dando lugar a diferentes aplicaciones en estos campos de la Realidad Virtual [21].

- Seguimientos de dedos y manos. *Figura 16 (b).*
- Paneles interactivos. *Figura 15 (b).*
- Seguimiento de la cabeza. *Figura 15 (a)*.
- Realidad espacial aumentada. *Figura 16 (a).*

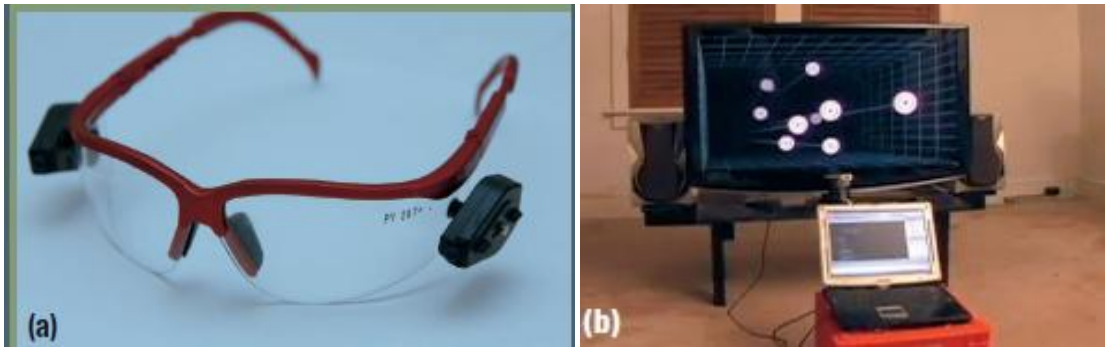

Figura 15. Seguimiento de la cabeza (a) y Panales Interactivos (b).

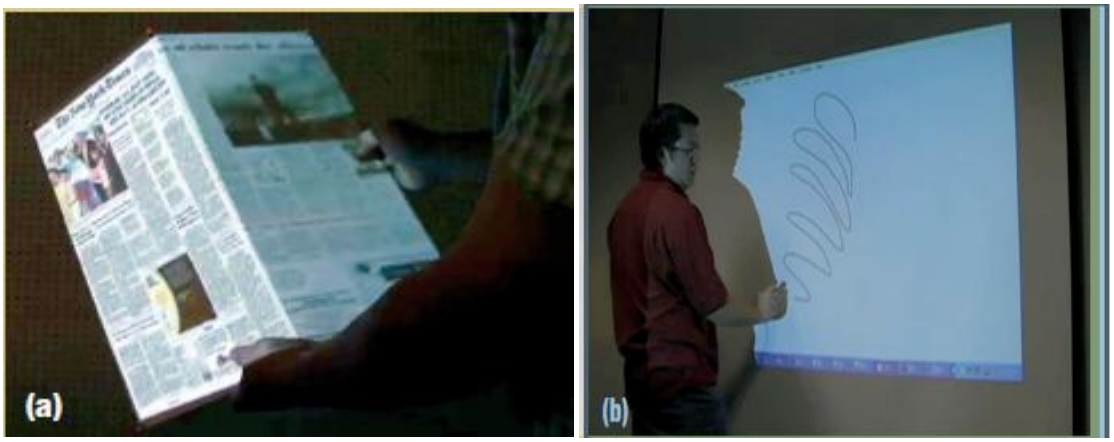

Figura 16. Realidad espacial aumentada (a) y Seguimientos de dedos y manos (b).

1.5. Biblioteca.

En las ciencias de la computación una biblioteca (del inglés *library*) es un conjunto de subprogramas utilizados para desarrollar un software. Las mismas contienen código y datos que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. De esta manera se utiliza lo que esta implementado en la biblioteca ahorrándose así tiempo y recursos, puesto que no se crea lo que ya existe, además esto permite que el código y los datos se compartan y puedan modificarse de forma modular [31].

1.5.1. Tipos de Bibliotecas.

Las bibliotecas pueden ser estáticas o dinámicas.

Biblioteca estática.

Una biblioteca estática, también conocida como archivo, consiste en un conjunto de rutinas que son copiadas en una aplicación por el compilador o el enlazador, produciendo archivos con código objeto y un fichero ejecutable independiente [31].

Biblioteca dinámica.

Un enlace dinámico significa que las subrutinas de una biblioteca son cargadas en un programa en tiempo de ejecución, en lugar de ser enlazadas en tiempo de compilación y se mantienen como archivos independientes separados del fichero ejecutable del programa principal [31].

1.5.2. Bibliotecas utilizadas para integrar el Wiimote.

A continuación se mostrará en una tabla las principales bibliotecas usadas para la integración del *Wiimote* a nivel mundial en diferentes sistemas operativos.

Tabla 1. Bibliotecas de integración del Wiimote.

Biblioteca	Windows	GNU/Linux
WiimoteLib	X	
WiiYourself!	X	
Motej	X	
CWiid		X
WiiuseJ	X	X
Wiiuse	X	X
WiiJuce	X	X

1.5.3. Características generales.

WiimoteLib.

Esta biblioteca fue escrita por Brian Peek [27], en el lenguaje C# por lo tanto está basada en la tecnología *.NET*, aunque solamente puede ser usada en *Windows* esta biblioteca permite leer los valores capturados por los acelerómetros, la cámara infrarroja del *Wiimote* y los botones del dispositivo. Recientemente han sido añadidas nuevas funcionalidades como la posibilidad de leer valores capturados por otros accesorios como la *Wii Balance Board* suministrada con el juego *Wii Fit*. También permite mandar ordenes al *Wiimote* como encender distintas luces *LED* de las que dispone, actualmente está en desarrollo la posibilidad de mandar sonidos para reproducirlos a través del altavoz del *Wiimote*.

Motej.

Es una biblioteca *Java* de comunicación con el *Wiimote*, bajo la licencia ASL 2.0. Está basada en dos paquetes: biblioteca y extras. La biblioteca ofrece acceso básico al *Wiimote*. Los extras por su parte extienden la funcionalidad básica ofrecida por la biblioteca añadiendo funciones más complejas para facilitar el trabajo del programador. La arquitectura de esta biblioteca se puede comprender con el siguiente esquema. Como se puede observar, la biblioteca depende de la implementación del estándar

de *Java JSR82*, por lo que requiere una *API* adicional para el acceso al dispositivo. El módulo de extras accede a las funciones básicas de la biblioteca para extender sus funcionalidades [23].

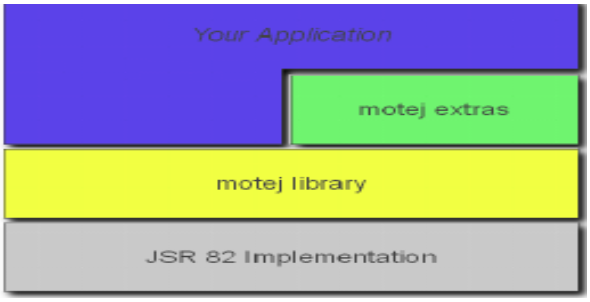

Figura 17. Arquitectura de la biblioteca Motej.

WiiYourself!

Es una biblioteca basada en la ***WiimoteLib*** creada por Brian Peek, pero está escrita en C++ y extendida en funcionalidades. Además de las funcionalidades que ofrece la biblioteca original esta añade algunas novedades como:

- **Estimación de la orientación:** Mediante los métodos añadidos es posible detectar el ángulo de Wiimote de una manera más sencilla.
- **Mayor soporte de gestores de Bluetooth:** Es capaz de utilizar cualquier programa gestor de conexiones Bluetooth.
- **Soporta múltiples Wiimotes:** Es capaz de soportar más de un *Wiimote* aumentando así la cantidad de dispositivos con el que se puede trabajar.
- **Soporta Wii Motion Plus:** El *Wii Motion Plus* es un pequeño dispositivo rectangular de apenas cinco centímetros que mejora la precisión del *Wiimote* capturando así los movimientos más pequeños, además detecta velocidades de giro en los tres ejes. En la *figura 18* se muestra este novedoso dispositivo.

Una de las limitaciones que posee esta biblioteca que al igual que su predecesor ***WiimoteLib*** solamente puede ser usada en *Windows*, aunque es una de las bibliotecas más completas que existen hasta el momento [3].

Figura 18.Wii Motion Plus conectado al Wiimote.

CWiid.

Una joven biblioteca que ha surgido bajo la tutela del software libre, esta implementada en C y orientada solamente a sistemas *GNU/Linux* siendo esta característica una de sus principales limitaciones, actualmente está en fase de desarrollo y tiene una Interfaz de programación de aplicaciones (*API*) muy reducida además es más complicada de usar que otras. Soporta funciones básicas del *Wiimote*. Está compuesta por las siguientes partes:

- **Biblioteca Wiimote**: Incluye la *API* de la *CWiid*.
- **Wmgui**: Interfaz de usuario gráfica (*GUI*) para el *Wiimote*.
- **Wminput**: Controlador de eventos, *joysticks* y *mouse* para el *Wiimote*.
- **Wmdemo**: Pequeña demostración de la *API* de la biblioteca.

Aunque esta biblioteca no soporta *Wii Motion Plus* se le pueden hacer unos arreglos para que esta detecte dicho dispositivo [6].

Wiiuse.

Biblioteca implementada en C, está bajo la licencia *GNU GPLv3* y *GNU LGPLv3* (no comercial), aunque no tiene tantas funcionalidades como las bibliotecas ya mencionadas la ventaja de *Wiiuse* es que esta desarrollada para *Windows* y *GNU/Linux* haciendo fácil la portabilidad. Una de sus funcionalidades principales es:

- **Soporte de acelerómetros.**
- **Infrarrojos.**
- **Soporte de algunas extensiones (Nunchuck y Classic Controller).**

Una de las limitaciones de esta biblioteca es que no soporta la extensión *Wii Motion Plus* [24].

WiiuseJ.

Esta biblioteca no es más que una adaptación de la biblioteca *Wiiuse* [25] para ser usada en *Java* en lugar de C, puede ser usada tanto en *GNU/Linux* como en *Windows*, es fácil de usar y contiene los mismas funcionalidades que *Wiiuse* y algunas que se le han añadido como:

- **Capturar los eventos de los botones.**
- **Puede hacer vibrar al motor de vibración.**
- **Cambia el estado de las luces LEDs.**

WiiJuce.

Basada en el lenguaje de programación C++, biblioteca multiplataforma puede ser usada en *Windows*, *GNU/Linux* y hasta *Macintosh*. Una de sus principales características es que soporta más de un *Wiimote*, puede mandar órdenes a las luces Leds, al motor de vibración y recibe los estados de los botones. Fue liberada bajo la licencia ***GPL v3*** por lo tanto es libre de uso [28]. Esta biblioteca tiene una *API* muy reducida lo que conlleva a que se limite el desarrollo con ella.

1.5.4. Deficiencias generales.

Como se había resaltado anteriormente en la descripción de las bibliotecas de integración, cada una de estas tiene un talón de Aquiles, unas porque solamente pueden ser utilizadas en un sistema operativo especifico, otras porque contienen una cantidad muy limitada de funcionalidades y otras porque están escritas en un lenguaje de programación que no es compatible con el lenguaje utilizado en el desarrollo de las aplicaciones gráficas desarrolladas en el centro. Estas bibliotecas tienen un gran problema en común y es que para obtener las funcionalidades que brindan se requiere de un estudio intensivo de las mismas y puede resultar engorroso el entendimiento de sus arquitecturas, por lo que se necesita de tiempo para estudiarlas lo que conllevará a que se retrase el proceso de desarrollo de las aplicaciones.

1.6. Metodologías de desarrollo de sotfware.

Según Piattini la metodología de desarrollo de software es “un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevos softwares.” (PIATINNI 1996).

Si se quiere construir un software con alta calidad, desarrollado en el tiempo planificado y con los costes establecidos, pero que además satisfaga la necesidad de ser elaborado de una forma más acelerada y que exista una reducción del costo del producto, es necesario enfocarse en trabajar de forma organizada, donde se controle y documente todo lo relacionado con el proyecto en cuestión y puedan eliminarse los riesgos que podrían presentarse durante el desarrollo del mismo, lo cual no puede lograrse sin el empleo de una metodología eficaz que se adapte a las características propias del software que se está desarrollando.

Clasificar las metodologías es una tarea bien difícil por la gran cantidad de propuestas y diferencias en el grado de detalle, información disponible y alcance que poseen.

A grandes rasgos, considerando la filosofía de desarrollo se pueden agrupar en dos grupos: **metodologías ágiles** o **ligeras** y **metodologías pesadas**.

Metodologías ágiles: Son aquellas que están orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen por grupos pequeños, se hace hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso. Entre estas metodologías se tiene:

- Extreme Programming (XP).
- SCRUM.
- Crystal Clear.
- Feature Driven Development (FDD).
- Dynamic Systems Development Method (DSDM).
- Adaptive Software Development (ASD).

Metodologías pesadas: Están guiadas por una fuerte planificación durante todo el proceso de desarrollo, en el cual se establecen estrictamente las actividades involucradas, los roles definidos, los artefactos que se deben producir, las herramientas y notaciones que serán utilizadas así como el modelado y documentación detallada. Ejemplos de estas metodologías tenemos a: SW-CMM (*Software Capability Maturity Model*), RUP (*Rational Unified Process*) entre otras.

1.6.1. Metodología a utilizar.

Después de ver las diferentes metodologías de desarrollo de software, se ha llegado a la conclusión que la metodología más indicada para usar en el desarrollo de la aplicación es **XP.**

Dicha metodología está basada en la simplicidad, la comunicación y el reciclado continuo de código, para algunos es aplicar una pura lógica, está diseñada para grupos pequeños de programadores más de 10 ya sería muy complicado, y para que estén en el mismo centro de trabajo. XP es una metodología liviana y ágil, está orientada más a las personas que a los procesos. Sus principales objetivos son muy simples: la satisfacción del cliente, trata de dar al cliente el software que el necesita y cuando lo necesite, y como segundo objetivo es potenciar al máximo el trabajo en grupo, tanto jefes de proyectos, clientes y desarrolladores forman parte del equipo y están involucrados en el desarrollo del software [9].

1.7. Herramientas de desarrollo.

Para el desarrollo de la solución propuesta se usará ***Codeblocks* 10.05** como entorno de desarrollo integrado, y para crear la documentación de esta, el ***Visual Paradigm*,** además se utilizará las bibliotecas de integración ***WiiYourself*!** y ***CWiid***.

Codeblocks.

Es un *IDE* libre, multiplataforma construido para satisfacer las necesidades más exigentes de usuario. Está diseñado para ser muy extensible y totalmente configurable. Soporta múltiples compiladores, es un proyecto *Open Source* y tiene un sistema muy rápido de compilación. [31].

Visual Paradigm.

Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de un software: análisis y diseño, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad. Permite dibujar todo tipo de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, además es una herramienta libre y multiplataforma usada en muchos proyectos de nuestra universidad para gestionar la calidad del software.

WiiYourself!

Es una biblioteca libre y de código abierto, está escrita en el lenguaje de programación C++, es una de las bibliotecas más completas en funcionalidades, soporta múltiples *Wiimotes,* permite la lectura de los eventos emitidos por la batería, botones, los acelerómetros y rayos infrarrojos, permite cambiar el estado de los LEDs y el del motor de vibración. Soporta todos los gestores *Bluetooth* y facilita la detección de la orientación del *Wiimote.*

CWiid.

Biblioteca que permite interactuar con el Wiimote de forma simple, aunque está en desarrollo contiene las funcionalidades básicas para desarrollar la solución propuesta como lectura de los eventos emitidos por la batería, botones, los acelerómetros y rayos infrarrojos, permite cambiar el estado de los LEDs y el motor de vibración además está escrita en el lenguaje de programación C++, es libre y de código abierto. Soporta todos los gestores *Bluetooth* y mediante métodos añadidos se facilita la detección de la orientación del *Wiimote*.

1.8. Lenguajes.

Para el desarrollo del software se escogieron dos lenguajes fundamentales, C++ como lenguaje de programación y UML como lenguaje de modelado.

1.8.1. Lenguaje de programación.

Fue escogido el lenguaje C++ debido a que constituye uno de los lenguajes más robustos y más usados a nivel mundial, es un lenguaje de alto nivel y multiparadigma debido a que abarca tres paradigma de la programación: la programación estructurada, la programación orientada a objetos y la programación genérica, además es un lenguaje de programación estandarizado, ampliamente difundido, además que las bibliotecas utilizadas para desarrollar el software están implementadas en dicho lenguaje.

1.8.2. Lenguaje de modelado.

Se escogió UML como lenguaje para realizar el modelado del análisis y diseño de la aplicación debido a que:

- UML ofrece un modo estándar de visualizar, especificar, documentar y construir los artefactos del sistema basado en el software que debe usarse en el proceso completo de desarrollo del mismo.
- Permite modelar sistemas utilizando técnicas orientadas a objetos.
- Puede conectarse con lenguajes de programación.

Consideraciones Parciales.

En este capítulo se realizó una investigación acerca del estado del arte de las bibliotecas de integración del *Wiimote*, seleccionándose la *WiiYourself!* y la *Cwiid* para la realización del presente trabajo. Debido a sus facilidades, documentación y flexibilidad se decidió a utilizar la metodología *XP*, además para la implementación de la biblioteca propuesta se determinó usar el lenguaje de programación C++ y como entorno de desarrollo integrado el *Codeblocks* 10.05. Como herramienta *CASE* se seleccionó *Visual Paradigm* apoyado en *UML* para el modelado del sistema.

Características del Sistema

El presente capítulo tiene como objetivo hacer una valoración de las principales características de la biblioteca a desarrollar. Se hace mención de las fases de exploración y planificación, propias de la metodología de desarrollo a utilizar para la implementación de la biblioteca que se propone y se exponen los artefactos generados durante el transcurso de las mismas.

2.1 Propuesta del Sistema.

Para darle solución al problema científico se propone el desarrollo de una biblioteca que permitirá a los desarrolladores integrar el Wiimote con aplicaciones gráficas en diferentes sistemas operativos ya sea *Windows* y *GNU*/*Linux*, la solución propuesta está basada en la utilización de bibliotecas no multiplataformas de integración del *Wiimote* ya existentes, *WiiYourself*! y *CWiid,* estas bibliotecas contienen una gran gama de funcionalidades disponibles para el uso de este dispositivo en las plataformas para las cuales fueron concebidas, lo que implicará que la nueva biblioteca heredará las características de estas y añadirá nuevas funcionalidades para obtener una biblioteca multiplataforma.

La nueva biblioteca estará compuesta por clases relacionadas y reutilizables, diseñadas para proporcionar funcionalidades útiles de propósito general, estará caracterizada por su eficiencia, robustez, simplicidad y sencillez, permitiendo así que su uso sea de gran ayuda para realizar aplicaciones gráficas que utilicen la tecnología del *Wiimote* en sistemas operativos *Windows* y *GNU*/*Linux.* Además brindará fácil acceso a las funcionalidades del *Wiimote* de forma transparente para el desarrollador.

2.2 Modelo del Dominio.

Se propone el modelo del dominio correspondiente con la *figura 13*, el cual es una representación visual estática del entorno real objeto de la solución propuesta. Se representan las clases conceptuales fundamentales asociadas al trabajo con la biblioteca. Además se presentan los eventos que suceden en el entorno en el que se trabaja y se muestran los objetos más importantes en el contexto del sistema.

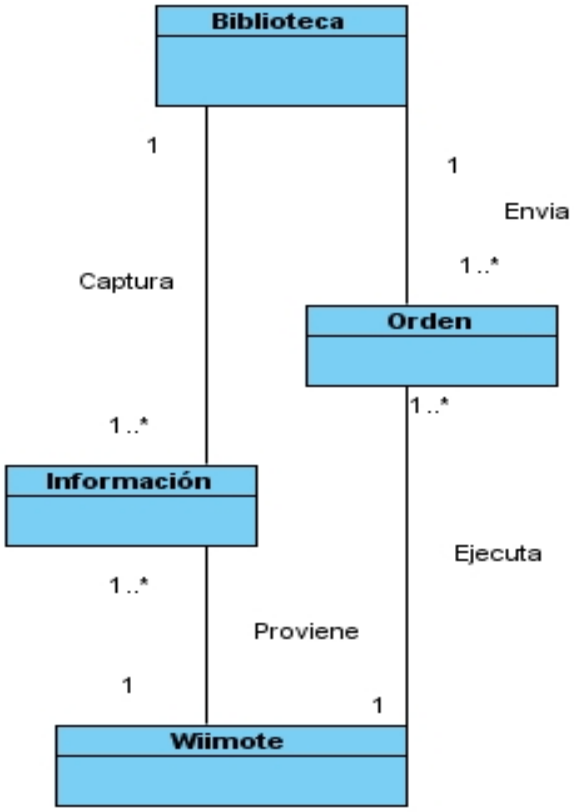

Figura 19. Diagrama del Modelo de Dominio.

A continuación se definen los conceptos utilizados en el modelo de dominio mediante un glosario de términos.

Biblioteca: Conjunto de clases y bibliotecas de software encargados de administrar las funcionalidades del sistema.

Información: Datos o parámetros provenientes del ***Wiimote***.

Orden: Comando o acción que la ***Biblioteca*** le ordena al ***Wiimote.***

Wiimote: Dispositivo inalámbrico.

La biblioteca es la encargada de capturar la información proveniente del ***Wiimote***, el cual ejecuta las órdenes o comandos enviados por la ***Biblioteca***.

2.3 Personal relacionado con el sistema.

Se define como personal relacionado con el sistema a todo aquel que obtiene un resultado del valor de uno o varios procesos que se ejecutan en el mismo. Además de aquellas personas que se encuentran involucradas en dichos procesos, y que participan en ellos pero no obtienen ningún resultado de valor.

Tabla 2. Personas relacionadas con el sistema.

Personas relacionadas con el sistema	Justificación
Usuario	Persona encargada de interactuar con el sistema para ejecutar las funcionalidades de la biblioteca

2.4 Requerimientos.

Los requerimientos establecen con detalle las funciones, servicios y restricciones operativas del sistema bajo las cuales se tienen las primeras ideas de una solución.

2.4.1. Requerimientos funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Estos requerimientos describen lo que el sistema debe hacer.

RF 1: Habilitar la conexión del dispositivo.

RF 2. Habilitar captura de los componentes del dispositivo.

RF 2.1. Habilitar captura de los botones del dispositivo.

RF 2.2. Habilitar captura del rayo infrarrojo del dispositivo.

RF 2.3. Habilitar captura del acelerómetro del dispositivo.

RF 3: Habilitar la desconexión del dispositivo.

2.4.2. Requerimientos no funcionales.

Hardware.

- *Wiimote:* Dispositivo para capturar los movimientos de los usuarios.
- *Bluetooth:* Dispositivo para la comunicación *PC-Wiimote.*

Rendimiento.

Debe tener un alto grado de velocidad de procesamiento o cálculo, tiempo de respuesta y de recuperación ante cualquier fallo.

Software.

Sistemas Operativos *Windows* y GNU/*Linux*, y las bibliotecas *Cwiid* y *WiiYourself*!

Portabilidad.

La biblioteca no debe requerir instalación, debe usarse en dos sistemas operativos *Windows* y *GNU*/*Linux*.

Diseño e Implementación.

Se codificará con el lenguaje de programación C++ utilizando el paradigma de la programación orientado a objeto (POO).

2.5 Diagrama de Clases del Diseño.

A continuación se muestra el diagrama de clases del diseño correspondiente con el sistema desarrollado.

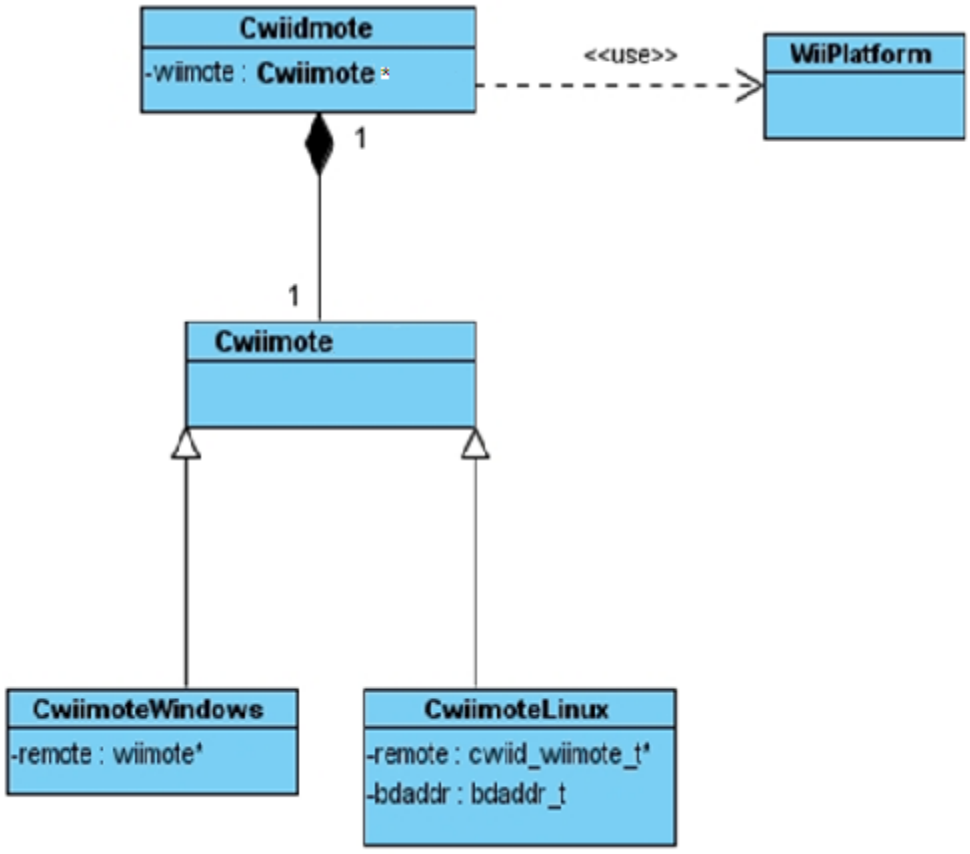

Figura 20. Diagrama de Clases del Diseño.

2.6 Fase de Exploración.

Durante esta etapa se define el alcance del proyecto, además los clientes plantean a grandes rasgos las historias de usuarios que son de gran interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, las tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.

2.6.1. Historias de Usuario.

Las historias de usuario (HU) se utilizan para representar una breve descripción del comportamiento del sistema, emplea terminología del cliente sin lenguaje técnico, se realiza una por cada funcionalidad del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación.

Como resultado del trabajo realizado durante las fases de exploración se identificaron las siguientes historias de usuario.

Tabla 3. HU Habilitar la conexión del dispositivo.

Historia de Usuario	
No: 1.	**Nombre:** Habilitar la conexión del dispositivo.
Usuario: Desarrollador.	
Prioridad en el Negocio: Alta.	**Nivel de Complejidad:** Alta.
Tiempo de Estimación: 2 semanas.	**Iteración Asignada:** 1.
Descripción: Establece una conexión bluetooth con el Wiimote, haciendo posible así que se comience el trabajo con el dispositivo.	
Información adicional (Observaciones): Antes de realizar cualquier acción con el Wiimote, primeramente se debe conectar.	

Tabla 4. HU Habilitar captura de los botones del dispositivo.

Historia de Usuario	
No: 2.	**Nombre:** Habilitar captura de los botones del dispositivo.
Usuario: Desarrollador.	
Prioridad en el Negocio: Alta.	**Nivel de Complejidad:** Alta.
Tiempo de Estimación: 1 semana.	**Iteración Asignada:** 1.
Descripción: Permite realizar la captura de la información de brinda los botones del dispositivo.	
Información adicional (Observaciones): Para realizar cualquier acción con los botones es necesario hacer la captura de los mismos.	

Tabla 5. HU Habilitar la captura del acelerómetro del dispositivo.

Historia de Usuario	
No: 3.	**Nombre:** Habilitar la captura del acelerómetro del dispositivo.
Usuario: Desarrollador.	
Prioridad en el Negocio: Alta.	**Nivel de Complejidad:** Alta.
Tiempo de Estimación: 1 semana.	**Iteración Asignada:** 1.
Descripción: Permite realizar la captura de la información que brinda el acelerómetro del dispositivo.	
Información adicional (Observaciones): En caso de que se quiera utilizar las aceleraciones en los diferentes ejes de coordenada es necesario habilitar la captura del acelerómetro.	

Tabla 6. HU Habilitar la captura del rayo infrarrojo.

Historia de Usuario	
No: 4.	**Nombre:** Habilitar la captura del rayo infrarrojo.
Usuario: Desarrollador.	
Prioridad en el Negocio: Alta.	**Nivel de Complejidad:** Alta.
Tiempo de Estimación: 1 semana.	**Iteración Asignada:** 2.
Descripción: Permite realizar la captura de la información que brinda el rayo infrarrojo del dispositivo.	
Información adicional (Observaciones: En caso de que se quiera realizar cualquier operación con el rayo infrarrojo del dispositivo es necesario habilitar la captura del rayo infrarrojo.	

Tabla 7. HU Habilitar la desconexión del dispositivo.

Historia de Usuario	
No: 5.	**Nombre:** Habilitar la desconexión del dispositivo.
Usuario: Desarrollador.	
Prioridad en el Negocio: Alta.	**Nivel de Complejidad:** Alta.
Tiempo de Estimación: 1 semana.	**Iteración Asignada:** 3.
Descripción: Elimina la conexión bluetooth con el Wiimote y para de leer la información del dispositivo.	
Información adicional (Observaciones): Antes de terminar de usar el dispositivo es necesario desconéctalo primero.	

2.7 Fase de Planeamiento.

En esta fase el cliente establece la prioridad de las historias de usuario y se acuerda el alcance del release. Los programadores estiman cuanto esfuerzo requiere cada historia de usuario y a partir de allí se define el cronograma.

2.7.1. Estimación de esfuerzo por Historia de Usuario.

Para un buen desarrollo del sistema propuesto, se realizó una estimación por cada una de las historias de usuario identificadas, llegando a los resultados que se muestran a continuación.

Tabla 8. Estimación de esfuerzos por Historias de Usuario.

Historias de usuarios.	**Puntos de estimación.**
Habilitar la conexión del dispositivo.	2 semanas.
Habilitar la captura de los botones del dispositivo.	1 semana.
Habilitar la captura del acelerómetro del dispositivo.	1 semana.
Habilitar la captura del rayo infrarrojo del dispositivo.	1 semana.
Habilitar la desconexión del dispositivo.	1 semana.
Total.	6 semanas.

2.7.2. Plan de iteraciones.

Las HU seleccionadas para cada entrega son desarrolladas y probadas en el ciclo de iteración, de acuerdo al orden establecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada HU traduce en tareas específicas de programación. Así mismo, para cada HU se establecen las pruebas de aceptación.

Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores.

Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como prever que no vuelva a ocurrir. Además de las entregas habituales al final de cada iteración, todas las semanas se hace un resumen con la presencia del cliente y se muestran los adelantos del sistema, y con el mismo objetivo de realizar las reuniones diarias entre los desarrolladores al final de cada sesión de trabajo.

Una vez definidas las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se decide realizar el sistema en tres iteraciones, las cuales se describen detalladamente a continuación:

2.7.2.1. Iteración 1.

Esta iteración tiene como objetivo darle cumplimiento a las HU que se son consideradas las de mayor importancia para el desarrollo de la biblioteca. Al concluir dicha iteración se contará con la primera versión de prueba y dando al sistema las primeras funcionalidades de las cuales fueron descritas en las HU 1, 2 y 3.

2.7.2.2. Iteración 2.

En esta iteración se realizará la implementación de las HU con prioridad de negocio alta. Además, se corregirá los errores o disconformidades del cliente con las HU implementadas en la iteración anterior. Esta segunda versión será mostrada a los clientes con el único objetivo de realizar cambios a la aceptación del mismo.

2.7.2.3. Iteración 3.

En la tercera iteración, ya implementadas las funcionalidades especificadas, se realiza el desarrollo de las ultimas HU con menor prioridad. De esta manera se obtiene la versión 1.0 del producto final.

2.7.3. Plan de duración de las iteraciones.

Como parte del ciclo de vida de un proyecto guiado por la metodología de desarrollo de software XP, se crea un plan de duración de cada una de las iteraciones que se llevarán a cabo durante el desarrollo del proyecto. Este plan tiene como finalidad mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las iteraciones.

Tabla 9. Plan de duración de iteraciones.

Iteración.	**Historias de usuarios.**	**Duración total de iteraciones**
Iteración 1	Habilitar la conexión del dispositivo.	4 semanas.
	Habilitar captura de los botones del dispositivo.	
	Habilitar captura del acelerómetro del dispositivo.	
Iteración 2.	Habilitar captura del rayo infrarrojo del dispositivo.	1 semana.
Iteración 3.	Habilitar la desconexión del dispositivo.	1 semana.

Consideraciones Parciales

En este capítulo se inicia el desarrollo de la solución propuesta que se desea implementar, se analizó la propuesta del sistema y los requerimientos funcionales y no funcionales del mismo. Además se trató todo lo referente a las dos primeras fases de la metodología de desarrollo de software a utilizar, fase de exploración y planificación del sistema, donde se documentó todos los artefactos generados en el transcurso de las mismas y quedó plasmado que el desarrollo de la biblioteca se hará en tres iteraciones.

Construcción de la Solución Propuesta.

En el presente capítulo se presentará el diagrama de clases de la arquitectura de la biblioteca, así como el estándar de codificación utilizado para la implementación de las clases. Se detallará las tres iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiendo las tareas generadas por cada historia de usuario y las pruebas de aceptación efectuadas.

3.1. Diseño de la solución propuesta.

La metodología de desarrollo XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

Para el diseño de las aplicaciones, la metodología XP no requiere la presentación del sistema mediante diagrama de clases utilizando notificación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad, Colaboración). No obstante el uso de estos diagramas puede aplicarse siempre y cuando influyan en el mejoramiento de la comunicación entre el equipo de desarrollo, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante

3.1.1. Tarjetas CRC.

Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, de las responsabilidades y sus colaboradores. En la siguiente tabla se muestra como se distribuye la información.

Tabla 10. Tarjeta CRC CWiidmote

Tarjeta CRC	
Clase: CWiidmote.	
Responsabilidad: Conectar el *Wiimote*. Desconectar el *Wiimote*. Obtener información del acelerómetro. Obtener estado del *Wiimote*. Calibrar el acelerómetro. Activar la captura del estado de los botones, el rayo infrarrojo y el acelerómetro. Cambiar el estado del *Rumble* (Motor del Wiimote). Cambiar el estado de los *Leds* (Luces). Obtener estado de la batería. Saber si está conectado el *Wiimote*. Obtener estado de los botones. Obtener estado de los *Leds*. Obtener posición del acelerómetro en los distintos ejes de movimiento. Obtener el valor del pitch del *Wiimote*. Obtener el valor del roll del *Wiimote*.	**Colaboraciones:** CWiiMote.

Tabla 11. Tarjeta CRC CWiiMote.

Tarjeta CRC

Clase: CWiiMote.	
Subclase: CWiiMoteWin. CWiiMoteLinux	
Responsabilidad:-.	**Colaboraciones:-.**

Tabla 12. Tarjeta CRC CWiiMoteWin.

Tarjeta CRC	
Clase: CWiiMoteWin.	
Super Clase: CWiiMote	
Responsabilidad: Conectar el *Wiimote*. Desconectar el *Wiimote*. Obtener información del acelerómetro Activar la captura del estado de los botones, el rayo infrarrojo y los acelerómetros. Cambiar el estado del *Rumble* (Motor del *Wiimote*). Cambiar el estado de los *Leds* (Luces). Obtener estado de la batería. Saber si está conectado el *Wiimote*. Obtener estado de los botones. Obtener estado de los *Leds*. Obtener posición del acelerómetro en los distintos ejes de movimiento. Obtener el valor del pitch del *Wiimote*. Obtener el valor del roll del *Wiimote*.	**Colaboraciones:** wiimote.

Tabla 13. Tarjeta CRC CWiiMoteLinux.

Tarjeta CRC	
Clase: CWiiMoteLinux.	
Súper Clase: CWiiMote	
Responsabilidad: Conectar el *Wiimote*. Desconectar el *Wiimote*. Obtener información del acelerómetro. Obtener estado del *Wiimote*. Calibrar el acelerómetro. Activar la captura del estado de los botones, el rayo infrarrojo y el acelerómetro. Cambiar el estado del *Rumble* (Motor del Wiimote). Cambiar el estado de los *Leds* (Luces). Obtener estado de la batería. Saber si está conectado el *Wiimote*. Obtener estado de los botones. Obtener estado de los *Leds*. Obtener posición del acelerómetro en los distintos ejes de movimiento. Obtener el valor del pitch del *Wiimote*. Obtener el valor del roll del *Wiimote*.	**Colaboraciones:** cwiid. Math.

3.1.2. Diagrama de componentes.

Un diagrama de componentes describen los elementos físicos del sistema y sus relaciones, muestran opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas, pueden ser simples archivos, paquetes o bibliotecas [17].

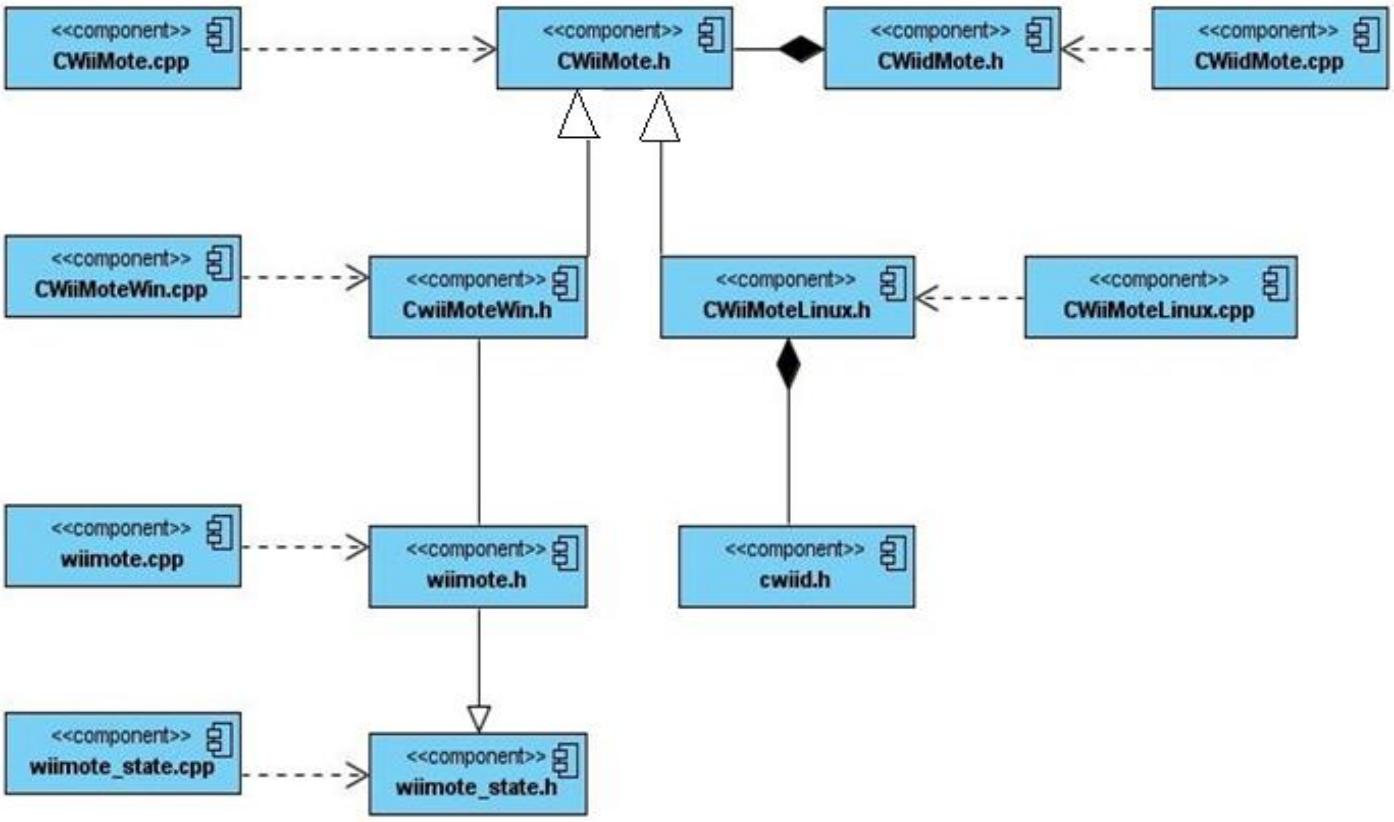

Figura 21. Diagrama de Componentes.

3.1.3. Patrones de diseño.

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. Existen problemas durante el desarrollo de software que se repiten o que son análogos, que responden a cierto patrón. Por lo que es deseable tener una colección de dichos patrones con soluciones óptimas para cada caso a modo de buenas prácticas. Durante el desarrollo de la biblioteca se utilizó el patrón que se destaca a continuación:

- **Encapsulamiento (ocultación de datos):** Propone el ocultamiento de los datos miembro de un objeto, de manera que solo se puede cambiar mediante las operaciones definidas para ese objeto.
- **Subclase:** Este patrón propone heredar miembros por defecto de una superclase, seleccionando la implementación correcta a través de resoluciones sobre que implementación debe ser ejecutada.

3.2. Desarrollo de las iteraciones.

En la fase de Planificación se detallaron los HU correspondientes con cada una de las iteraciones a desarrollar, teniendo en cuenta las necesidades requeridas por el cliente. Durante el transcurso de las iteraciones se lleva a cabo una revisión del plan de iteraciones y se modifica en caso de que sea necesario. Como parte de este plan, se descomponen las HU en tareas de programación o de ingeniería, asignando a un equipo de desarrollo (o una persona), responsable de su implementación, aplicando la práctica de la programación en parejas. Estas tareas no tienen que necesariamente ser entendidas por el cliente, pueden ser escritas en lenguaje técnico y son para el uso estricto de los programadores.

Teniendo en cuenta la planificación realizada con anterioridad, se llevó a cabo el desarrollo del sistema en tres iteraciones, obteniéndose como finalidad un producto con todas las restricciones y características deseadas por el cliente. A continuación se detallan cada una de las iteraciones.

3.2.1. Iteración 1.

En esta iteración se da cumplimiento a la implementación de las HU que corresponden con los números 1, 2 y 3 consideradas de mayor importancia para el desarrollo de la aplicación, con el fin de obtener una

versión del producto con algunas funcionalidades críticas como: Conectar el dispositivo, habilitar la captura de los componentes del mismo.

Tabla 14. HU abordadas en la primera iteración.

Historias de Usuario	**Tiempo de implementación (semanas).**	
	Estimación	**Real**
Habilitar la conexión del dispositivo.	2	2
Habilitar captura de los botones del dispositivo.	1	1
Habilitar captura del acelerómetro del dispositivo.	1	1

Tabla 15. Tarea de Ingeniería de la HU 1.

Tarea de Ingeniería.	
No. De la Tarea: 1.	No. De la HU: 1.
Nombre de la tarea: Habilitar la conexión del dispositivo.	
Tipo de tarea: Desarrollo.	Puntos estimados: 2.
Fecha inicio:16/03/2011	Fecha fin: 30/03/2011.
Programador responsable: Antonio David Rubán Espinal.	
Descripción: Esta en la funcionalidad es la más importante debido a que es la primera acción que debe realizarse si se quiere utilizar el dispositivo. Se implementa con el objetivo de conectar el dispositivo a la aplicación que se esté realizando en ese momento.	

Tabla 16. Tarea de Ingeniería de la HU 2.

Tarea de Ingeniería.	
No. De la Tarea: 2.	No. De la HU: 2.
Nombre de la tarea: Habilitar la captura de los botones del dispositivo.	
Tipo de tarea: Desarrollo.	Puntos estimados: 1.
Fecha inicio:31/03/2011	Fecha fin: 06/04/2011.
Programador responsable: Antonio David Rubán Espinal.	

Descripción: Funcionalidad que se encarga de capturar y control de los eventos emitidos por los botones del dispositivo al ser presionados.

Tabla 17. Tarea de Ingeniería de la HU 3.

Tarea de Ingeniería.	
No. De la Tarea: 3.	No. De la HU: 3.
Nombre de la tarea: Habilitar la captura del acelerómetro del dispositivo.	
Tipo de tarea: Desarrollo.	Puntos estimados: 1.
Fecha inicio:07/04/2011	Fecha fin: 14/04/2011.
Programador responsable: Antonio David Rubán Espinal.	
Descripción: Esta funcionalidad se encarga de capturar y controlar las aceleraciones en los tres ejes de coordenadas emitidas por el acelerómetro del dispositivo.	

3.2.2. Iteración 2.

En esta iteración se implementaron las HU que corresponden con el número 4. Esta HU brinda la funcionalidad de capturar el comportamiento del rayo infrarrojo del dispositivo.

Tabla 18. HU abordadas en la segunda iteración.

Historias de Usuario	**Tiempo de implementación (semanas).**	
	Estimación	**Real**
Habilitar captura del rayo infrarrojo del dispositivo.	1	1

Tabla 19. Tarea de Ingeniería de la HU 4.

Tarea de Ingeniería.	
No. De la Tarea: 4.	No. De la HU: 4.
Nombre de la tarea: Habilitar la captura del rayo infrarrojo del dispositivo.	

Tipo de tarea: Desarrollo.	Puntos estimados: 1.
Fecha inicio:15/04/2011	Fecha fin: 22/04/2011.
Programador responsable: Antonio David Rubán Espinal.	
Descripción: Esta funcionalidad se implementa con el objetivo de obtener los parámetros emitidos por el rayo infrarrojo del dispositivo.	

3.2.3. Iteración 3.

En esta última iteración se le dio cumplimiento a la HU 5. Dicha HU desempeña la funcionalidad de desconectar el dispositivo de la aplicación.

Tabla 20. HU abordadas en la tercera iteración.

Historias de Usuario	**Tiempo de implementación (semanas).**	
	Estimación	**Real**
Habilitar la desconexión del dispositivo.	1	1

Tabla 21. Tarea de Ingeniería de la HU 5.

Tarea de Ingeniería.	
No. De la Tarea: 5.	No. De la HU: 5.
Nombre de la tarea: Desconectar el dispositivo.	
Tipo de tarea: Desarrollo.	Puntos estimados: 1.
Fecha inicio: 23/04/2011.	Fecha fin: 30/04/2011.
Programador responsable: Antonio David Rubán Espinal.	
Descripción: Esta funcionalidad se utiliza para cerrar las conexiones bluetooth usadas por el Wiimote.	

3.3. Pruebas.

Unos de los pilares de la metodología XP es el uso de las pruebas para componer el funcionamiento de los códigos que se vayan implementando. La metodología XP divide las pruebas en dos grupos: pruebas

unitarias y pruebas de aceptación. Las pruebas unitarias son desarrolladas por los programadores y se encargan de verificar el código automáticamente y las pruebas de aceptación están destinadas a verificar que al final de cada iteración las Historias de Usuario cumplen con la funcionalidad asignada y satisfagan las necesidades del cliente.

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente, por esto el cliente es el indicado para diseñar las pruebas de aceptación [16].

3.3.1. Desarrollo dirigido por pruebas.

El desarrollo dirigido por pruebas (TDD), es una práctica de programación que involucra otras prácticas como: Escribir las pruebas primero (*Test First Development*) y Refactorización (*Refactoring*). Para escribir las pruebas generalmente se utiliza la Prueba Unitaria (*Unit Test*).

3.3.2. Pruebas de Aceptación.

Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

Las pruebas de aceptación son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendado publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de la información.

Para la realización de cada una de las pruebas de aceptación se siguieron una serie de pasos que se muestran a continuación:

- Identificar todas las acciones en la historia de usuario.
- Para cada acción escribir al menos una prueba.
- Para algunos datos, reemplazar las entradas que hacen que la acción ocurra y llenar en la casilla resultados esperados y los resultados obtenidos.
- Para otros datos, reemplazar las entradas que hacen que la acción falle y registrar los resultados.

Tabla 22. Prueba de Aceptación para HU "Habilitar la conexión del dispositivo".

Caso de Prueba de Aceptación.	
Código: HU1_P1	Historia de Usuario: 1.
Nombre: Comprobar la conexión del dispositivo.	
Descripción: Evaluar la conexión del dispositivo.	
Condiciones de Ejecución: El cliente debe comprobar que se realice la conexión del dispositivo correctamente.	
Entrada/ Pasos de Ejecución: Probar que se conecte el dispositivo en la aplicación gráfica.	
Resultado Esperado: Se conecta correctamente el dispositivo.	
Evaluación de la Prueba:	

Tabla 23. Prueba de Aceptación para HU "Habilitar la captura de los botones del dispositivo".

Caso de Prueba de Aceptación.	
Código: HU2_P2	Historia de Usuario: 2.
Nombre: Comprobar la captura de los botones del dispositivo.	
Descripción: Evaluar la captura de los eventos emitidos por los botones que conforman el dispositivo.	
Condiciones de Ejecución: El cliente debe comprobar que la captura de los eventos emitidos se realice correctamente.	
Entrada/ Pasos de Ejecución: Probar que se capture los eventos emitidos por los botones del dispositivo en la aplicación gráfica.	
Resultado Esperado: Se captura exitosamente los eventos emitidos por los botones del dispositivo.	
Evaluación de la Prueba:	

Tabla 24. Prueba de Aceptación para HU "Habilitar la captura del acelerómetro del dispositivo".

Caso de Prueba de Aceptación.	
Código: HU3_P3.	Historia de Usuario: 3.
Nombre: Comprobar la captura del acelerómetro del dispositivo.	

Descripción: Evaluar la captura de las aceleraciones emitidas por el acelerómetro.
Condiciones de Ejecución: El cliente debe comprobar que se realice la captura de las aceleraciones.
Entrada/ Pasos de Ejecución: Probar la captura del acelerómetro.
Resultado Esperado: Se captura exitosamente las aceleraciones emitidas por el acelerómetro.
Evaluación de la Prueba:

Tabla 25. Prueba de Aceptación para HU "Habilitar la captura del rayo infrarrojo".

Caso de Prueba de Aceptación.	
Código: HU4_P4.	Historia de Usuario: 4.
Nombre: Comprobar la captura del rayo infrarrojo del dispositivo.	
Descripción: Evaluar la captura de los parámetros emitidos por el rayo infrarrojo del dispositivo.	
Condiciones de Ejecución: El cliente debe probar que se realice la captura de los parámetros emitidos por el rayo infrarrojo en la aplicación gráfica deseada.	
Entrada/ Pasos de Ejecución: Probar la captura del rayo infrarrojo en cualquier aplicación gráfica.	
Resultado Esperado: Se comprueba correctamente la captura del rayo infrarrojo.	
Evaluación de la Prueba:	

Tabla 26. Prueba de Aceptación para HU "Habilitar la desconexión del dispositivo".

Caso de Prueba de Aceptación.	
Código: HU5_P5.	Historia de Usuario: 5.
Nombre: Comprobar la desconexión del dispositivo.	
Descripción: Evaluar la desconexión del dispositivo.	
Condiciones de Ejecución: El cliente debe comprobar la desconexión del dispositivo en la aplicación gráfica.	
Entrada/ Pasos de Ejecución: Probar la desconexión del dispositivo.	
Resultado Esperado: Se comprueba correctamente la desconexión del dispositivo.	
Evaluación de la Prueba:	

Consideraciones Parciales

En este capítulo se brindó información detallada de las clases que componen la aplicación, así como sus atributos y relaciones entre ellas. Se elaboraron las tarjetas CRC y se definen los estándares de codificación durante el desarrollo. De igual manera de desarrollar las tareas que dan solución a las historias de usuario. Se realizan pruebas unitarias y se definen las pruebas de aceptación. Con la culminación de este capítulo se da por terminada la propuesta de solución de la aplicación a implementar.

Conclusiones

Como resultado de este trabajo se logró el diseño e implementación de una biblioteca que permite la integración del *Wiimote* con las aplicaciones gráficas que se desarrollan en el centro CEDIN, en los sistemas operativos *Windows* y *GNU/Linux.*

Se obtuvo una biblioteca que ahorrará tiempo de desarrollo al equipo que requiera utilizar el *Wiimote* en las aplicaciones gráficas que implementen.

Mediante el uso de la biblioteca desarrollada se logró la integración del *Wiimote* con las siguientes aplicaciones gráficas en desarrollo "*Beat it*", "*Crashing*" y "*Lanzado*"

Recomendaciones

Debido a los resultados de la investigación efectuada y la experiencia adquirida durante la realización de este trabajo, y con el propósito de asegurar una futura modificación y mejora de la biblioteca propuesta, se exponen a continuación algunas recomendaciones:

- Mantener actualizada la biblioteca propuesta.
- Continuar con el proceso de desarrollo de la biblioteca con el objetivo de añadirle nuevas funcionalidades, específicamente en el área del sonido.

Bibliografía

1. **Alcor Martín, Enrique.** *Estudio y desarrollo de interfaces hombre-máquina basados en sensores inalámbricos.* Madrid : s.n., 2008.

2. **Gutiérrez, Iván Hernández.** *Wii-imersion: Aprovechando las capacidades del Wiimote para la mejora de la sensación de inmersión.* 2010.

3. WiiYourself! [En línea] 2010. http://wiiyourself.gl.tter.org/.

4. Cwiid. [En línea] 2010. http://www.cwiid.org/.

5. **Navarro, Miguel Angel Granado.** *Estudio de las posibilidades de un Wiimote como interfaz de control.* 2009.

6. *CWiid.* [En línea] 2010. http://abstrakraft.org/cwiid/.

7. *MEDIALAB VISUALIZATION WORKSHOP.* [En línea] http://www.cmpe.boun.edu.tr/medialab/VW/Wiimote_Linux_integration.pdf.

8. **Lee, Johnny Chung.** *Johnny Chung Lee Projects.* [En línea] 2010. http://johnnylee.net/projects/.

9. **Calderón, Amaron y Valverde Rebaza, Sarah Damaris.** *Metodologías ágiles.* Trujillo, Perú : s.n., 2007.

10. **Calero Solis, Manuel.** *Una explicación de la programación extrema (XP).* Madrid : V Encuentro usuarios xBase, 2003.

11. *Diseño metodológico de la investigación científica. Teoría de Muestreo: población y muestra.Diseño experimental y métodos.* Ciudad Habana : Revisado por Msc Pedro Carlos Pérez Martinto.

12. **Markiewicz, Marcus Eduardo y de Lucena, Carlos J.P.** *Desarrollo del Framework Orientado a Objeto.*

13. **Sommerville, Ian.** *Ingeniería de Software. Séptima Generación.* Madrid : PEARSON EDUCACIóN.S.A, 2005.

14. **Prieto, Félix, y otros.** *Construcción de frameworks basada en análisis de concepto formales y soportada por Mecanos.*

15. **Ideas, Dos.** [En línea] http://www.dosideas.com/wiki/Test_Driven_Development.

16. **Gutiérrez, J. J., y otros.** *PRUEBAS DEL SISTEMA EN PROGRAMACIóN.* University of Sevilla : s.n.

17. **Ramírez., Lic. Elisa Arizaca.** [En línea] http://virtual.usalesiana.edu.bo/web/practica/archiv/compon.doc..

18. CEDETEL. [En línea] http://www.cedetel.es/.

19. **Muros Ponce, Francisco Javier, Vicaria Flores, Juan Antonio y Maestre, José María.** Aplicaciones del controlador wiimote para personas con discapacidad. [En línea] http://www.google.com/url?sa=t&source=web&cd=4&ved=0CCwQFjAD&url=http%3A%2F%2Fautomatica2009.uva.es%2Ffiles%2Fp64.pdf&rct=j&q=navegacion%20web%20wiimote&ei=UKjMTcjsOZCitgegvKyVBQ&usg=AFQjCNHAktJBdZk3UZgE4Bdcnp7bOyXOQg&cad=rja.

20. **Lee, Johnny Chung.** *Hacking the Nintendo Wii Remote.* s.l. : IEEE Pervasive Computing, 2008.

21. Wii Device Library. [En línea] "http://www.wiili.org/index.php/Wii_Device_Library.

22. **SourceForge.** motej- A Wiimote library for Java. [En línea] http://motej.sourceforge.net/index.html.

23. wiiuse. [En línea] http://www.wiiuse.net.

24. Google Code.WiiuseJ. [En línea] http://code.google.com/p/wiiusej.

25. WiiLi.WiiremoteJ. [En línea] http://www.wiili.org/WiiremoteJ.

26. Coding4Fun. Managed Library for Nintendo's Wiimote. [En línea] http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx.

27. WiiJuce. [En línea] http://www.diplabs.com/svn/wiijuce/www/index.html.

28. OGRE. [En línea] http://www.ogre3d.org.

29. Codeblocks. [En línea] http://www.codeblocks.org/.

30. Diferenciación de conceptos en el área del desarrollo de aplicaciones. [En línea] http://sodvi.info/blog/es-MX/entry/1711516073/Diferenciacion-de-conceptos-en-el-area-del-desarrollo-de-aplicaciones.

31. *Developer Network.* [En línea] http://developer.qt.nokia.com/wiki/Qt_Coding_Style_Spanish.

32. *Framework de colecciones.* [En línea] http://www.dccia.ua.es/dccia/inf/asignaturas/RG/2008-2009/pdf/colecciones-java.pdf.

33. Wiili.org. [En línea] http://homepage.mac.com/ianrickard/wiimote/wiili_wimote.html.

34. Classic Systems Nintendo Entertaimenmt Syste. [En línea] [Citado el: 05 de 06 de 2011.] http://www.vivienlos80s.com/2010/02/nintendo/.

35. Homenaje a Nintendo la mejor empresa y videojuegos. [En línea] [Citado el: 05 de 06 de 2011.] http://listas.20minutos.es/lista/homenaje-a-nintendo-la-mejor-empresa-de-videojuegos-y-consolas-221690/.

36. Poco a poco fue evolucionando. [En línea] 05 de 06 de 2011. http://juegoswiiypc.blogspot.es/.

37. Project worldwide sales of next-gen consoles. [En línea] http://www.google.com/imgres?imgurl=http://images3.wikia.nocookie.net/__cb20080521193631/vgsales/images/9/93/ProjectedConsoleChart_v1_460x323.gif&imgrefurl=http://gamrconnect.vgchartz.com/thread.php%3Fid%3D126952%26page%3D4&usg=__1V2hELaZzHT_ebisZiUvg-k3e.

Anexos

Anexo A: A continuación se muestran una serie de imágenes obtenidas mediante el uso de la biblioteca desarrollada en algunas aplicaciones gráficas.

Figura A.1: Aplicación gráfica "Crashing".

Figura A.2: Aplicación gráfica "Beat it".

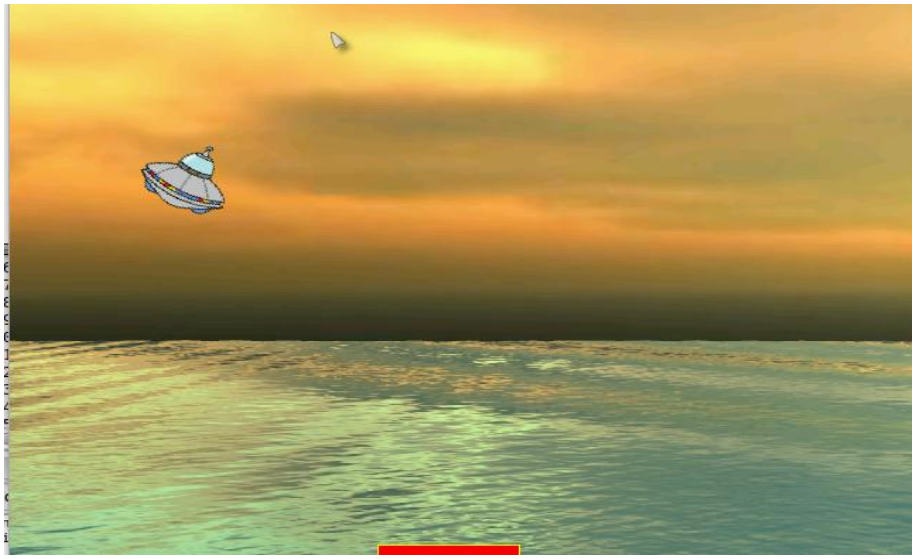

Figura A.3: Aplicación gráfica "Lanzado".

Anexo B: Manual de Usuario para la utilización de la biblioteca.

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

Para poder utilizar el *Wiimote* en nuestras aplicaciones gráficas en *Windows* y GNU/*Linux* debemos seguir los siguientes pasos:

1. **Establecer la conexión entre el Wiimote y nuestra aplicación**.
2. **Elegir la información que deseamos obtener del Wiimote.**
3. **Leer las informaciones o enviar órdenes al Wiimote**.

1. **Establecer la conexión entre el Wiimote y nuestra aplicación gráfica**.

Este paso consiste en buscar el *Wiimote* conectado a la PC y establecer la conexión con el mismo. Con la utilización de la biblioteca desarrollada basta con las siguientes líneas de código para empezar a comunicarnos con el *Wiimote*.

```
Cwiidmote wiimote;
wiimote.cwiiConnect();
```

2. **Elegir el tipo de información que deseamos obtener del Wiimote.**

Tras realizar la conexión del *Wiimote* con nuestra aplicación, debemos habilitar la captura de los componentes del dispositivo.

```
wii.enableAccIrBtn();
```

3. **Leer las informaciones o enviar órdenes al Wiimote**.

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

3.1. **Lectura de la posición del acelerómetro**.

```
double cwiiAccelerationX();
double cwiiAccelerationY();
double cwiiAccelerationZ();
```

3.2. **Lectura de la pulsación de botones, estado de los Leds y batería.**

Pulsación de botones: Se podrá averiguar si alguno de los botones de los que dispone el *Wiimote* ha sido presionado (estos comandos devolverán un valor booleano).

```
bool cwiiButtonA();
bool cwiiButtonB();

bool cwiiButton1();
bool cwiiButton2();

bool cwiiButtonUp();
bool cwiiButtonDown();
bool cwiiButtonLeft();
bool cwiiButtonRight();
```

Estado de los Leds: devolverá un valor booleano.

```
bool cwiiLed1On();
bool cwiiLed2On();
bool cwiiLed3On();
bool cwiiLed4On();
```

Nivel de carga de la batería: devolverá un valor tipo entero.

```
int cwiiBattery();
```

3.3. Envío de órdenes.

La biblioteca permite el envío de algunas órdenes básicas al *Wiimote* como el encendido y apagado tanto de los *Leds* como el del motor de vibración.

Para utilizar el comando que permite encender o apagar los *Leds*, se debe especificar uno por uno el estado que se desea que quede el *led*.

Cambio del estado de los Leds.

```
void cwiiSetLedState(unsigned int led_state);

wii.cwiiSetLedState(0X01); //cambiar led 1
wii.cwiiSetLedState(0X02); //cambiar led 2
wii.cwiiSetLedState(0X04); //cambiar led 3
wii.cwiiSetLedState(0X08); //cambiar led 4
```

Activar y desactivar el motor de vibración.

```
void cwiiSetRumbleState(bool rumble);

wii.cwiiSetRumbleState(true); //activar rumble
wii.cwiiSetRumbleState(true); //desactivar rumble
```

Glosario

C

C++: Lenguaje de programación orientado a objeto.

Código abierto: (del inglés *open source*) es el término con el que se conoce al *software* distribuido y desarrollado libremente.

E

Estándar: adj. Se dice de lo que sirve como tipo, modelo, norma, patrón o referencia por ser corriente.

I

Interfaz de Programación de Aplicaciones (API): Conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

L

Licencia Pública General GNU: más conocida por su nombre en inglés *GNU General Public License* o simplemente su acrónimo de inglés GNU GPL, es una licencia creada por la *Free Software Foundation*. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades de usuarios.

Lenguaje de Programación: Conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

M

Metodología: Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal.

Metodología de Desarrollo: Marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Multiplataforma: Que la aplicación corra en varios sistemas operativos.

R

Realidad Virtual: Simulación generada por computadoras de imágenes o ambientes tridimensionales interactivos con cierto grado de realismo físico o visual.

Render: Es el proceso de convertir modelos geométricos *3D* en escenas *2D* para presentarlas al usuario.

S

Software: Equipamiento lógico o soporte lógico de un ordenador. Comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (*hardware*). Suele ser utilizado para referirse a los programas que se integran a un sistema de información interconectado y complementario.

T

Tridimensional (3D): adj. Se dice de lo que se desarrolla en las tres dimensiones espaciales, altura, anchura y profundidad.