INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INNOVACIÓN Y DESARROLLO TECNOLÓGICO EN CÓMPUTO

DESARROLLO DE UN SISTEMA DE NAVEGACIÓN PARA REALIDAD VIRTUAL, BASADO EN UN CAMINO SIN FIN CON CONEXIÓN USB

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRÍA EN TECNOLOGÍA DE CÓMPUTO

P R E S E N T A :

ING. GABRIEL LEONARDO SÁNCHEZ ESPINOSA

DIRECTORES DE TESIS:

DR. MAURICIO OLGUÍN CARBAJAL

DR. JUAN CARLOS HERRERA LOZADA

México D.F. a 10 de Noviembre de 2011

SIP-14-BIS

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 10:00 horas del día 10 del mes de Noviembre del 2011 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del CIDETEC para examinar la tesis titulada:

"DESARROLLO DE UN SISTEMA DE NAVEGACIÓN PARA REALIDAD VIRTUAL, BASADO EN UN CAMINO SIN FIN CON CONEXIÓN USB"

Presentada por el alumno:

SÁNCHEZ	ESPINOSA	GABRIEL LEONARDO
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	0	9	1	3	8	4

aspirante de:

Maestría en Tecnología de Cómputo

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA
Directores de tesis

DR. MAURICIO OLGUÍN CARBAJAL
Primer Vocal

DR. JUAN CARLOS HERRERA LOZADA
Segundo Vocal

M. EN C. ISRAEL RIVERA ZÁRATE
Presidente

M. EN C. JESÚS ANTONIO ÁLVAREZ CEDILLO
Secretario

M. EN C. MIGUEL HERNÁNDEZ BOLAÑOS
Tercer Vocal

PRESIDENTE DEL COLEGIO DE PROFESORES

DR. VÍCTOR MANUEL SILVA GARCÍA

S.E.P.
INSTITUTO POLITECNICO NACIONAL
CENTRO DE INNOVACION Y DESARROLLO TECNOLOGICO EN COMPUTO

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 24 del mes de noviembre del año 2011, el que suscribe Gabriel Leonardo Sánchez Espinosa, alumno de la Maestría en Tecnología de Cómputo con número de registro B091384, adscrito al Centro de Innovación y Desarrollo Tecnológico en Cómputo, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. Mauricio Olguín Carbajal y el Dr. Juan Carlos Herrera Lozada, y cede los derechos del trabajo intitulado "Desarrollo de un sistema de navegación para realidad virtual, basado en un camino sin fin con conexión USB", al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección: gabriell.sancheze@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Gabriel Leonardo Sánchez Espinosa

Resumen

Los sistemas de realidad virtual están cambiando a diario, encaminándose a convertirse en herramientas cotidianas, con las cuales hemos de convivir cada vez más a menudo. Puede considerarse un claro ejemplo a las nuevas televisiones 3D, los controles de videojuegos que detectan el movimiento y hasta a las aplicaciones de dispositivos móviles que utilizan sensores de diferentes tipos.

El presente trabajo está orientado a ampliar las capacidades de una cabina de inmersión para realidad virtual (CAVE, Cave Automatic Virtual Enviroment) al desarrollar un dispositivo de control del usuario en ambientes simulados, tomando como herramienta central la comunicación vía USB y la adaptación del control a una caminadora empotrada en el interior de la instalación. Esto permite actualizar el sistema, mejorar el desempeño y da paso a la liberación de puertos (se deja de usar el puerto PS/2), además de aumentar la sensación de inmersión del usuario.

El desarrollo está dirigido a movimientos comunes de una persona por medio de sensores adaptados en un espacio tridimensional, como son el desplazamiento y el cambio de dirección (rotación). Como instrumento de interacción con elementos virtuales, se encuentra un elemento de selección que es posible utilizar para los fines requeridos por el diseñador o programador del mundo artificial. Por medio de los experimentos pertinentes (evaluando desempeño, mediciones adecuadas y aceptación del usuario), el sistema ha mostrado su funcionamiento y mejora en relación al sistema anterior debido a las adaptaciones hechas y a las modificaciones de la interfaz.

Como resultado, se consigue un control intuitivo de desplazamiento por ambientes simulados que utiliza el puerto USB para mejorar su rendimiento, portabilidad y actualización; se liberan otros puertos que se usaban en el sistema original, lo que permite aumentar la sensación de inmersión del usuario; además, de ha mostrado bajo experimentación una buena aceptación por parte del usuario.

El sistema de navegación es una herramienta que permite realizar recorridos para educación, construcción, terapia, experimentación y demás con alto grado de inmersión debido a los controles actualizados, haciendo más evidente que los sistemas de realidad virtual ya son nuestro presente y conforme las tecnologías en desarrollo salgan a la luz, éstos seguirán en continuo crecimiento tanto en alcances como en desempeño.

Abstract

Virtual reality systems are changing every day, transforming in daily tools which we could live very often. We could considerer for example, TVs with 3D technology, videogame's controls that can detect the user's movement and mobile device applications with different kinds of sensor. The virtual reality systems are here right now, and as the new technologies are developed, this kind of systems would grow both in scope and performance.

The purpose of this work is to extend the immersion cockpit capabilities (CAVE), through developing a user's control device for virtual environment, which incorporate USB communication and a treadmill like user's control into the cockpit.

This work takes the natural moves of a person, like orientation and position changes, helped by adapted sensors in a 3D space. For interact with virtual elements, the treadmill incorporates a selector device, which can be used for designer or programmer requirements into artificial world.

As a result, is achieved an intuitive control for position changes in virtual environments, using USB technology for improve its performance, portability and update. The development also frees the other ports and allows increasing the immersion feeling, besides to count with a great acceptance by the user.

Agradecimientos

Por tanto tiempo invertido en mi trabajo de investigación, en correcciones, explicaciones, enseñanzas, y sobre todo, en la construcción de un proyecto llamado *Tesis*, agradezco a mis directores, Dr. Mauricio Olguín Carbajal y Dr. Juan Carlos Herrera Lozada por la guía a lo largo de estos meses y esa actitud de mejora continua que ha hecho de éste, un trabajo del cuál enorgullecerse.

A mi familia le estoy muy agradecido por brindarme su apoyo constantemente en los proyectos en que me he involucrado. Les debo mucho a mis padres, Gabriel y Elizabeth, por siempre regalarme su cariño, preocupación y consejo, mismos que han hecho de mí una persona comprometida con los objetivos que llego a plantearme.

A Hedrian y Andonni les debo, además de esa divertida convivencia y grandes momentos de aprendizaje, el darme el privilegio de ser su hermano. Quiero que sepan que todo el tiempo he intentado ser una influencia positiva en sus vidas, tratando de recompensar lo que ustedes aportan a mis experiencias. Por esto mismo, les dedico este trabajo como una parte de mí, de los intereses y gustos que constituyen mi personalidad, con el fin de que me conozcan un poco más y esperando conocer su opinión.

Por esas incesantes muestras de interés y verdadera inquietud por este trabajo y su exitosa conclusión, te agradezco infinitamente Brigitte. Sin el soporte que simbolizas para mí, las metas que una vez soñé alcanzar fácilmente podrían haberse esfumado en el aire.

Y a mis amigos, esos compañeros de vida que me acompañan desde niño, aquellos que entraron en mi entorno como consejeros y hermanos, les debo demasiado. Les agradezco las pláticas, los ánimos, los regaños y la visualización de un futuro prometedor. A su lado, todo sucede como uno lo desea. Por esto y más, les dedico este trabajo que representa un capítulo más en mi vida, en donde ustedes también están presentes. Gracias Javier, Abraham, Luis, Yezbek, Isaac, Víctor Hugo, Chucho, Juan, Oscar.

A todos les prometo que nunca me daré por vencido en la vida.

Índice de contenido

Resumen ii
Abstract iii
Agradecimientos iv
Índice de figuras viii
Índice de tablas x
Capítulo I . Introducción 1
1.1 Motivación 4
1.2 Relevancia 5
1.3 Justificación 5
1.4 Objetivos 6
1.4.1 Objetivo general 6
1.4.2 Objetivos específicos 6
1.5 Alcances y contribuciones 6
1.6 Organización del trabajo 7
Capítulo II . Estado del arte 9
2.1 Introducción 9
2.2 Antecedentes 9
2.3 Trabajos relacionados 9
Capítulo III . Marco teórico 25
3.1 Introducción 25
3.2 Antecedentes 25
3.3 Teoría 26
3.3.1 Realidad virtual 26
3.3.2 Los elementos de la realidad virtual 27
3.3.3 Sistema de despliegue 28
3.3.4 Sistemas de navegación 29
3.3.5 Sistemas de rastreo 31
3.3.6 Elementos de programación 34
3.3.7 Elementos del dispositivo 38
3.4 Materiales y métodos 42
Capítulo IV . Desarrollo 44

4.1 Introducción 44
4.2 Propuestas 44
4.3 Aportaciones 45
4.4 Desarrollo 45
4.4.1 Ambiente virtual 45
4.4.2 Componentes físicos 50
4.4.3 Comunicación USB 72
Capítulo V . Experimentos y resultados 77
5.1 Introducción 77
5.2 Planteamiento de los experimentos 77
5.2.1 Comunicación en red 78
5.2.2 Comunicación instrucciones USB – Servidor 79
5.2.3 Sensores para los controles 79
5.2.4 Manejo del sistema 81
5.2.5 Navegación 82
5.3 Realización de experimentos 83
5.3.1 Comunicación en red 83
5.3.2 Comunicación instrucciones USB - Servidor 84
5.3.3 Sensores para los controles 87
5.3.4 Manejo del sistema 90
5.3.5 Navegación 91
5.4 Evaluación de resultados 93
5.4.1 Comunicación en red 93
5.4.2 Comunicación instrucciones USB - Servidor 94
5.4.3 Sensores para los controles 94
5.4.4 Manejo del sistema 94
5.4.5 Navegación 95
Capítulo VI . Conclusiones y trabajo futuro 96
6.1 Aportes 97
6.2 Productos 98
6.3 Trabajo futuro 98
Referencias 100
Anexos 101
Código principal ControlUSB 101

Código de interpretación de movimiento (Módulo de desplazamiento) 107
Código de interpretación de señales (Módulo de rotación y selección)..... 109
Código de recepción de órdenes y conversión (Módulo de generación de órdenes) 110
Código de transmisión de órdenes al equipo (Módulo de comunicación USB) 112
Código de despliegue del ambiente virtual e interacción de controles (Servidor)..... 113
Código de despliegue del ambiente virtual e interacción de controles (Cliente)..... 128

Índice de figuras

Figura 1.1 Caminadora deportiva 2
Figura 2.1 Uniport 10
Figura 2.2 Caminadora omnidireccional 11
Figura 2.3 Principio de funcionamiento de la caminadora omnidireccional 11
Figura 2.4 Trans-e: mi cuerpo, mi sangre, por Diana Domingues 13
Figura 2.5 Osmose, por Char Davies 13
Figura 2.6 Rotación y retroceso registrados por medio de una cámara 14
Figura 2.7 CirculaFloor 15
Figura 2.8 Distribución de los elementos del sistema CirculaFloor 15
Figura 2.9 Esquema de la cabina 17
Figura 2.10 Caminadora y sus componentes 17
Figura 2.11 El interior del Orb Mouse 18
Figura 2.12 Detección de dedos del Side Mouse 19
Figura 2.13 Base de balance, de Nintendo Wii 20
Figura 2.14 Funcionamiento de la base de balance 20
Figura 2.15 Wiimote, de Nintendo 21
Figura 2.16 Controles del sistema PlayStation Move (a) y Kinect (b) 22
Figura 2.17 Virtusphere 23
Figura 3.1 Tres I'es de la realidad virtual 28
Figura 3.2 Patentes más antiguas de dispositivos montados en la cabeza (1916 y 1960) 29
Figura 3.3 Esquema de 6 grados de libertad 30
Figura 3.4 Sistemas de navegación que incorporan inmersión de alto grado 31
Figura 3.5 Ejemplo de dispositivo de rastreo mecánico 32
Figura 3.6 Protocolo TCP y UDP 36
Figura 3.7 Recepción de segmentos con protocolo TCP a) sin errores y b) con errores 37
Figura 3.8 Principios de una caminadora 40
Figura 4.1 Modelos de árboles 47
Figura 4.2 Ambiente conformado por la unión de modelos y efectos 48
Figura 4.3 Diagrama de transmisión de datos 49
Figura 4.4 Comunicación entre equipos 49
Figura 4.5 Diagrama de bloques general 51
Figura 4.6 Diagrama de bloques del módulo de desplazamiento 51

Figura 4.7 Diagrama de flujo del submódulo de interpretación de movimiento 53
Figura 4.8 Diagrama de bloques del módulo de rotación y selección 54
Figura 4.9 Diagrama de flujo del submódulo de interpretación de señales 55
Figura 4.10 Diagrama de bloques del módulo de generación de órdenes 56
Figura 4.11 Diagrama de flujo del módulo de recepción de órdenes y conversión de la señal 57
Figura 4.12 Diagrama de bloques del módulo de comunicación USB 58
Figura 4.13 Circuito final con módulos integrados 59
Figura 4.14 Submódulo de conversión del primer sensor de desplazamiento 60
Figura 4.15 Submódulo de conversión del segundo sensor de desplazamiento 60
Figura 4.16 Etapa de interpretación de sub módulos 61
Figura 4.17 Placa de sensores de desplazamiento 61
Figura 4.18 Módulo de conversión directa del acelerómetro y de la selección 63
Figura 4.19 Placa de sensores de rotación y selección 64
Figura 4.20 Módulo de generación de órdenes 65
Figura 4.21 Placa de generación de órdenes 66
Figura 4.22 Módulo de comunicación USB 67
Figura 4.23 Placa de comunicación USB 68
Figura 4.24 Rueda del rodillo de la caminadora 69
Figura 4.25 Estructura para montar los sensores infrarrojos 69
Figura 4.26 Elementos montados en la caminadora (sensores de desplazamiento) 70
Figura 4.27 Acelerómetro encargado de la rotación 70
Figura 4.28 Botones situados a los extremos del control de giro 71
Figura 4.29 Módulos de control finalizados. Puede apreciarse el control de giro (izquierda) y el control de desplazamiento (derecha) 71
Figura 4.30 PIC18F4550 instalado para pruebas en protoboard 73
Figura 4.31 Pantalla de ControlUSB (intermediario) 74
Figura 4.32 Diagrama estructural del sistema 76

Índice de tablas

Tabla 1. Resultados de pruebas referentes a la comunicación en red 84
Tabla 2. Resultados de los experimentos referentes a la transmisión de órdenes 86
Tabla 3. Resultados de las pruebas a los sensores de los controles..... 89
Tabla 4. Resultados de la experimentación con distintos controles de giro..... 91
Tabla 5. Reunión de resultados de los experimentos referentes a los controles de giro..... 91
Tabla 6. Resultados de las pruebas entre sistemas..... 93

Capítulo I. Introducción

Para comenzar, hablemos de los conceptos básicos manejados en la tesis; uno de ellos es el de *realidad virtual*. Esta no es una idea reciente, ni tampoco una invención de últimas fechas. De hecho, la realidad virtual surge desde los primeros teléfonos, televisores, radios, etc. Puede hallarse en cualquier elemento que simule o que se encargue de representar un fenómeno de manera diferente a la usual, pero que cumpla con el funcionamiento al que se encamina. Es posible decirse que, si nuestros sentidos perciben algo que en esencia es real, es un elemento *virtualmente real*. Para profundizar en su relación con el trabajo actual, es posible ver al sistema de archivos y carpetas de una computadora como un almacenamiento de documentos virtual, ya que esencialmente se resguarda la información aunque no existan físicamente las carpetas o los archivos.

Ya que el objetivo que se plantea se relaciona con la interacción del mundo real y el virtual, debe existir algún elemento o conjunto de éstos que funcione de intermediario en el proceso. Los elementos físicos tienen la tarea de unificar la realidad virtual con el mundo en el que vivimos, proporcionando una sensación de inmersión en la realidad ficticia.

Continuando con las definiciones primarias, y entrando al apartado de los elementos tangibles del sistema, se puede considerar adecuado puntualizar el concepto de *camino sin fin*. Un camino sin fin, de manera simple, se puede decir que es un camino o vía infinito. Entrando a una definición más técnica producto de la observación, se trata de una vía sobre la cual un cuerpo puede desplazarse de un punto inicial a un punto final, pero tomando en consideración lo siguiente:

- Se cuenta con un grado de libertad, es decir, con movimiento en una dirección.
- Existe una fuerza que provoca el movimiento.
- El movimiento puede continuar infinitamente, si las condiciones son ideales.

En el caso del trabajo actual, se maneja un camino sin fin implementado en una caminadora deportiva, como se puede observar en la figura 1.1, donde se cubren los puntos antes mencionados, debido a que:

- Existen dos movimientos dentro de este sistema, donde pueden encontrarse un inicio y un fin respectivamente.
- Se tiene un sólo grado de libertad que es hacia delante o hacia atrás, aunque este último no es el sentido tradicional de su utilización.
- El cuerpo del usuario se mueve hacia adelante gracias a las fuerzas ya conocidas del andar diario. Al llegar al punto final, la fuerza de gravedad aplicada al cuerpo sobre la caminadora (usuario) provoca el desplazamiento hacia atrás debido a la ligera inclinación del aparato. En el caso de requerir un movimiento hacia adelante, la fuerza es provocada por el usuario por medio de fricción sobre el camino y como consecuencia de la ley de acción y reacción.
- Si una persona contara con suministros de energía perpetuos, ésta podría caminar por siempre.

Figura 1.1 Caminadora deportiva

Por sí misma, la caminadora no deja de ser sólo un artículo para ejercitarse. Para que realmente sirva como interfaz entre el usuario y el ambiente virtual, se requiere de elementos sensibles a ciertas condiciones del mundo real, conocidos de forma más común como *sensores*. Para el caso particular de la caminadora, los sensores son posicionados de manera estratégica en la estructura para registrar el movimiento de la banda, o camino sin fin, e interpretarlo como un desplazamiento en el ambiente simulado. Sin embargo, esto es sólo posible con un grado de libertad. En el caso del cambio de orientación, se necesita otro grupo de sensores o controles que ayuden a representar un giro dentro del mundo virtual. Actualmente, la caminadora instalada en la caverna consta de una adaptación mecánica de los componentes de un ratón para que por medio de una perilla sea posible girar hacia los lados durante el recorrido.

Aún queda un punto por definir: la visualización del entorno. Si se pretende navegar dentro de un mundo virtual, sería primordial contar con un elemento de despliegue del mismo, como pudiera ser una pantalla de computadora. En el caso de una cueva de inmersión, el ambiente es proyectado en *n* número de pantallas traslúcidas con la ayuda de varios cañones o proyectores.

Si ahora observamos la definición de una caverna de inmersión, vemos que tenemos completo el entorno, ya que se requiere:

- Sistema de rastreo (sensores).
- Elemento de despliegue del mundo virtual.
- Reducción de contacto del usuario con el mundo real.

Este último punto se consigue gracias al aislamiento que produce la cabina al "encerrar" al usuario entre las pantallas traslúcidas y fijar su atención en estas. Al conformar los elementos mencionados anteriormente, se obtiene un acercamiento a la instalación en la que se lleva a cabo el desarrollo. En el caso particular de la tesis, se parte de la instalación de la cabina de inmersión del CIDETEC, que cuenta con los sensores, el despliegue y el camino sin fin, que forman el modelo de partida en la observación del funcionamiento, la experimentación y la propuesta impresa en los objetivos.

La importancia de todo esto no es nada intrascendente, ya que la infinidad de aplicaciones de este sistema continuamente está creciendo, teniendo como ejemplo a la educación, ya que se fortalece la comprensión de conocimientos porque se aplican los métodos auditivo, visual y kinestésico [1]. En mayor medida puede reforzarse este último debido a su fuerte dependencia de movimientos, y se recomienda efectuar visitas fuera del salón de clases para mejorar la retención (es posible simular visitas a zonas arqueológicas, museos, ciudades, e incluso al macro y microcosmos). Si no existiera inmersión en el sistema, el alumno fácilmente se apartaría de la cátedra y perdería la atención.

1.1 Motivación

Los sistemas avanzan, se actualizan constantemente, y lo hacen con el fin de mantenerse vigentes y proporcionar a los usuarios mayores ventajas al utilizar dichos sistemas. Si se toma un dispositivo como algo imposible de mejorarse, se cae en una inevitable ruta al olvido y desplazamiento de un sistema mejor.

Desde que fue concebido el desarrollo de mundos simulados en la computadora, la interacción, o las necesidades de ésta, han variado con el paso de los años. Comúnmente, el desplazamiento en ambientes virtuales se lleva a cabo por medio de diferentes dispositivos:

- Ratón
- Teclado
- Palancas
- Simuladores

Para ciertos fines, estos elementos cubren las necesidades del mundo virtual. Pero si lo que se desea es simular un desplazamiento a pie, el control de estos dispositivos no se acerca a la experiencia real de caminar, por lo que la sensación de inmersión se ve reducida. Si se deja a un lado esta sensación de inmersión, también es posible ver que tener este tipo de controles no siempre es un lujo. En el caso de terapias para personas que han sufrido alguna clase de trauma en las articulaciones inferiores o para aquellos sistemas que permiten a deportistas incrementar su desempeño con entrenamientos rigurosamente planeados, el uso de palancas de control u otros elementos resulta inútil. También existen investigaciones que requieren de una actividad física similar al caminar diario tanto para estudiar reacciones de los usuarios como para mejorar entornos virtuales.

Además por más simple que sea el dispositivo, su operación no llega a ser tan intuitiva para las personas ajenas al uso de computadoras. Debido a esto, se requieren conocimientos previos en cuanto a su utilización, y más aún si son utilizados en entornos ajenos a su función primaria, como el teclado que fue creado para escribir en la computadora, y que en muchos casos los botones desencadenan eventos o se encargan de desplazar objetos.

1.2 Relevancia

Por medio de sensores adecuados y un posicionamiento estratégico de los mismos, la información que es arrojada puede tratarse para reflejar un cambio, tanto de orientación como de posición. Esto trae consigo una nueva forma de manejarse dentro de los mundos virtuales para dejar atrás a los otros dispositivos, ya que fueron hechos para otra función ajena a los ambientes creados en computadora.

Al conseguir un desarrollo apropiado a las características de la cabina de inmersión, o a cualquier otro sistema enfocado a la realidad virtual, se puede:

- Mejorar el control del mundo virtual.
- Liberar de la tarea de desplazamiento y la rotación a herramientas como el ratón y el teclado.

Así, el presente trabajo plantea la obtención de los elementos necesarios para llevar a cabo la inmersión de alto grado en los ambientes artificiales en el interior de la cabina, donde podrá utilizarse para múltiples fines, como pueden ser investigación, entretenimiento o salud. Ahora bien, si se determina la utilización del puerto USB para la conexión del dispositivo, la velocidad será elevada y prácticamente podría acoplarse a cualquier equipo de cómputo.

1.3 Justificación

La realidad virtual ha visto un crecimiento muy rápido en los últimos años. En algunas aplicaciones, el principal problema es la interacción con este tipo de herramientas, porque comúnmente se hace con el ratón de la computadora o con botones del teclado; esto no es una opción viable si se habla de una cabina de inmersión. En este tipo de instalación, el usuario se sitúa en el interior del cuarto y por medio de otros dispositivos se controlan ciertas variables del entorno, como el desplazamiento.

Ya que se cuenta con un camino sin fin para llevar a cabo esta tarea, se requieren sensores bien acoplados y que respondan rápido, debido a que el movimiento en el mundo virtual es en tiempo real.

1.4 Objetivos

1.4.1 Objetivo general

Diseñar un sistema de navegación de realidad virtual basado en un camino sin fin con conexión USB como herramienta para lograr un mayor grado de inmersión en los usuarios de realidad virtual.

Con esto se busca proporcionar al usuario un sistema que asemeje la sensación de caminar, con lo cual se ahorra tiempo en aprender su utilización. Cabe recalcar que se enfoca su utilización a escenarios en los que se requiera simular un recorrido a pie, como paisajes naturales, zonas arqueológicas, museos, proyectos arquitectónicos, complejos industriales, etc.

1.4.2 Objetivos específicos

1. Definición de la solución tecnológica para el hardware y software a usar.
2. Adecuación del sistema mecánico y sus sensores.
3. Diseño de la interfaz con la adecuación electrónica.
4. Pruebas y optimización del prototipo.

1.5 Alcances y contribuciones

- El sistema fácilmente será adaptable a cualquier cabina de inmersión que cumpla con los principios básicos de funcionamiento de la actual instalación.
- El desarrollo tendrá la oportunidad de utilizarse en otros ambientes no tan relacionados con la realidad virtual, y aprovechar los sensores y la programación a nivel de microcontrolador para fines distintos.
- Se cuenta con la publicación de tres artículos, dos en boletines nacionales y el tercero en una revista de divulgación internacional.

1.6 Organización del trabajo

La presente investigación se muestra en orden cronológico formado por los siguientes capítulos. Ésto permite una comprensión sencilla del texto partiendo de los antecedentes, del conocimiento que respalda a la tesis, de la construcción del trabajo, los experimentos y finalmente del aprendizaje y transformación de las metas planteadas como propuesta a favor de un futuro mejoramiento. En este capítulo uno, ya se ha hecho mención de motivación, objetivos y principios aplicados.

Dentro del capítulo dos se incluye un conjunto de trabajos previamente elaborados alrededor del mundo, que se relacionan en algún nivel con el presente desarrollo. El objetivo de contar con este punto, es el comparar resultados de investigadores, así como la teoría de sus textos, las pruebas realizadas, las herramientas de las que se han valido y las conclusiones que se han obtenido, con el fin de utilizar toda esta información a favor del presente proyecto.

En el capítulo tres se hace referencia a la teoría en la que se basa el trabajo de investigación. Abarcando desde principios de realidad virtual hasta la estructura y programación de un microcontrolador, pasando por la conexión por puerto USB y otros elementos electrónicos. Además, el funcionamiento de la cabina de inmersión se contempla en este apartado ya que también aporta teoría indispensable para la adaptación del trabajo a las condiciones deseables de desempeño.

Se presenta en el cuarto capítulo el progreso del trabajo punto a punto. Íntimamente relacionado con la sección de experimentación y registro de resultados, es una recopilación de las decisiones tomadas gracias al éxito de las pruebas efectuadas en el capítulo cinco. Cabe mencionar que se cuenta con datos relacionados al trabajo en la sección de apéndices para su revisión en caso necesario. El avance en el desarrollo del trabajo se alcanza gracias a la obtención de resultados de las pruebas de la investigación. Se imprime en el presente apartado el informe de cada experimento, contando con información estadística que revele la toma de decisiones con respecto al progreso por cada etapa. Se incluye texto e imágenes que ilustran las condiciones de cada prueba y también de los resultados obtenidos.

Se destaca en el último capítulo el potencial de la investigación, el trabajo que continúa y amplía los objetivos y se plantea para la mejora y actualización del desarrollo en un futuro. Además, se exponen las impresiones con respecto al avance y finalización del escrito, contemplando

conocimientos adquiridos, experiencias y observaciones llevadas a cabo tanto particular como generalmente.

Capítulo II. Estado del arte

2.1 Introducción

A lo largo de la historia, y en muchas partes del mundo, han surgido miles de investigaciones referentes al área de la computación, derivando esto en el surgimiento de nuevos conocimientos y de nuevas ramas de estudio.

En el actual apartado, se hace una evaluación de los trabajos de investigación con relación al tema de tesis y se discriminan aquellos factores que puedan o no contribuir al correcto ensamblaje y desarrollo del presente trabajo. Además, se profundiza en aspectos de realidad virtual que hacen hincapié en la inmersión del usuario, la explotación de las sensaciones y la facilidad de manejo del sistema.

2.2 Antecedentes

Como primer punto se tiene un trabajo bastante importante, tomando en cuenta el papel que desempeña y el sector que lo utilizó, se puede decir que fue antecedente de los sistemas de rastreo para ambientes virtuales; esto significa que partiendo de él es como ha crecido la tecnología hasta nuestros días y por esto ha sido incluido.

La secuencia seleccionada va determinada por el orden cronológico, pero también por la cercanía al desarrollo planteado. Esto permite que sean revisados de mejor manera y facilita en cierto grado la comprensión de lo que pretende alcanzar esta tesis. La transición de los trabajos de investigación mencionados en el texto no van ligados en su totalidad, ya que las herramientas que utilizan son las que merecen su estudio, más no del todo su funcionalidad.

2.3 Trabajos relacionados

Los caminos sin fin han sido utilizados en proyectos relacionados con la realidad virtual. Sin embargo, la problemática con la que se han enfrentado los investigadores es el conseguir mediciones precisas de desplazamiento y giro, así como la velocidad de éstos. Se puede encontrar como ejemplo

al *UniPort* (figura 2.1), materializado en 1994, el cual fue el primer dispositivo construido para el movimiento y esfuerzo de la parte inferior del cuerpo [2]. Éste es un sistema que simula a un monociclo en cuanto a su operación básica, donde el usuario puede ir en cualquier dirección como si realmente caminara o corriera. Sin embargo, el dispositivo es fijo y los desplazamientos se llevan a cabo por el movimiento de los pedales. Los giros del *UniPort* se registran por medio del apoyo de la cintura del participante. Esto es un aspecto importante, ya que se empieza a ver un sistema omnidireccional por medio de sus sensores en partes estratégicas.

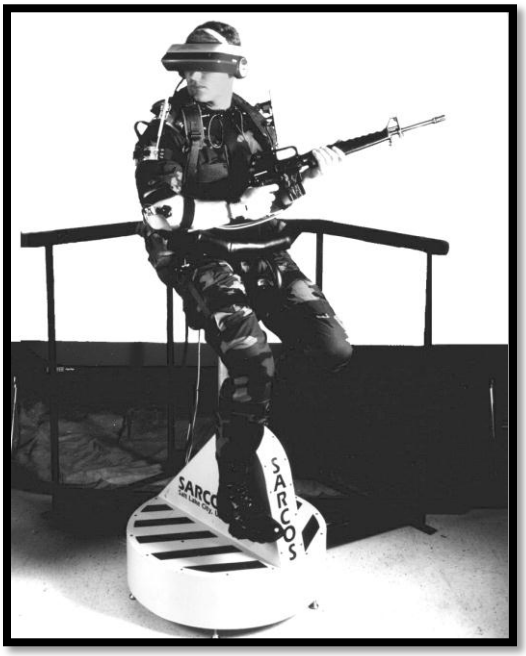

Figura 2.1 Uniport

La libertad de giros y movimientos del usuario es un punto a favor de este trabajo. Por otro lado, se ha encontrado que resulta muy difícil realizar pequeños movimientos, debido a que tiene problemas maniobrando con distancias cortas, además de resultar incómodo después de cierto tiempo de uso. No obstante, se toma como base para otras investigaciones posteriores, mismas en las que se deben crear condiciones ergonómicas y mejorar el desempeño para cumplir con las necesidades del usuario.

Otro de los trabajos representativos dentro del campo, es el realizado con respecto al camino sin fin básico (caminadora unidireccional) [2]. La banda sin fin omnidireccional (figura 2.2) es un dispositivo revolucionario. Consiste en dos cintas perpendiculares, una dentro de otra. La cinta superior, conformada por cadenas de cilindros rotatorios, descansa sobre la segunda banda

construida de manera similar (figura 2.3). Además, se conforma el sistema por servo-motores, los cuales se encargan de mover las bandas de acuerdo a la inclinación del usuario, misma que es registrada por sensores sujetos a la espalda baja del participante. Hay que destacar que la creación del camino sin fin omnidireccional ofrece un control de desplazamiento más intuitivo, por medio de movimientos opuestos de cierta aceleración y velocidad de las cintas, además de la fuerza ejercida por el arnés para mantener al usuario centrado.

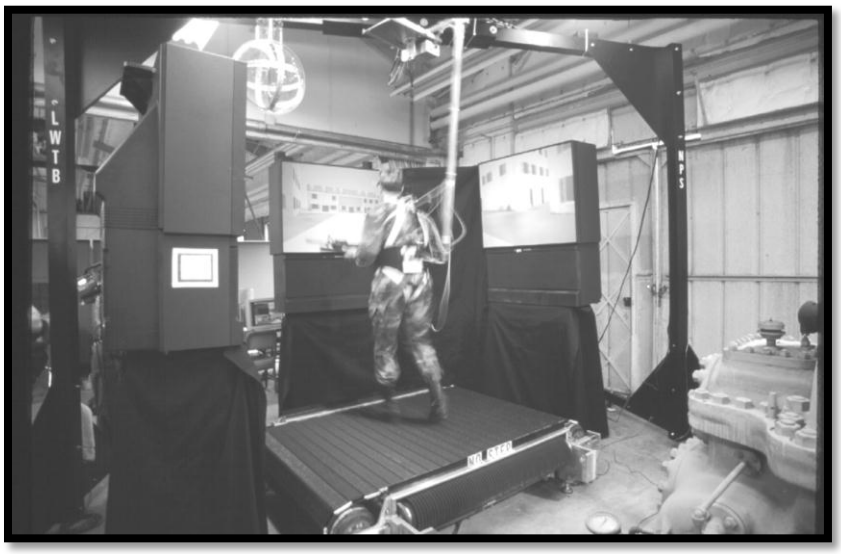

Figura 2.2 Caminadora omnidireccional

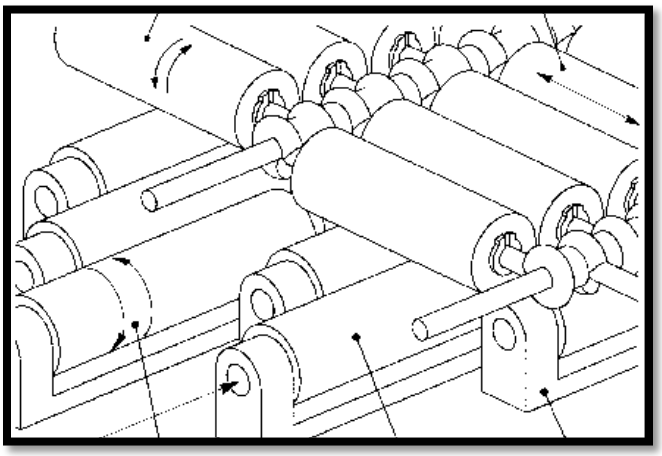

Figura 2.3 Principio de funcionamiento de la caminadora omnidireccional

Con base a la referencia [2] se expone una serie de caminadoras, entre las cuales encontramos al *UniPort*, y su evolución hasta llegar a considerar toda dirección de movimiento y una

mayor comodidad. Podemos ver que el nivel de libertad del usuario ha ido incrementándose al dejar de usar asientos o bandas en una sola dirección. Al llegar a este desarrollo, se puede ver a un usuario más dinámico y con una sensación de inmersión ampliada.

Es importante mencionar que una desventaja es que debe dedicarse tiempo a aprender a utilizar este sistema, debido a que no es tan sencillo su uso ya que se debe considerar la velocidad de desplazamiento, de desaceleración, la fuerza que debe aplicarse sobre la cinta y más conocimiento que el usuario necesita asimilar. Tal vez se necesite un control más intuitivo para reducir estos tiempos y que la sensación de inmersión crezca.

Por otra parte, en el trabajo [3] del 2003 se indica que se puede conocer un aspecto más profundo y humano de esta área. Se habla de la falsa idea de interactividad que la tecnología ofrece a las personas, de la pérdida del verdadero significado de la palabra *'realidad'*. También lo que se muestra en este análisis, es el trabajo de los artistas por recuperar la realidad utilizando a la realidad virtual inmersiva, donde se ha logrado una auténtica reacción por el mundo simulado ya que el usuario no sólo ocupa un espacio, sino que su propia condición de inmerso le hace interactuar con el medio, pero interactúa sensorialmente. Se consigue una realidad sensorial que nadie puede negar si es percibida por los propios sentidos del espectador.

Dentro del análisis hecho por [3] se expone [4] en 2001, donde se recrea una cueva en la cual el usuario experimenta *"alucinaciones virtuales"* en tiempo real (figura 2.4). Es un espacio sagrado donde los espectadores caminan sobre sensores que obedecen a sus desplazamientos, y en el que las imágenes y sonidos son provocados por el propio espectador a través de los sensores. Esta creación brasileña cuenta con claras desventajas, como la imprecisión en cuanto al registro del movimiento y a la limitante de que sólo este ambiente se ha desarrollado para esta caverna. Sin embargo, a diferencia de la caminadora omnidireccional, el usuario no sólo debe de poder moverse por todo el ambiente, sino que además el mismo ambiente crea reacciones a niveles más profundos del participante. Asimismo, puede verse el mismo fenómeno en los desarrollos de [5].

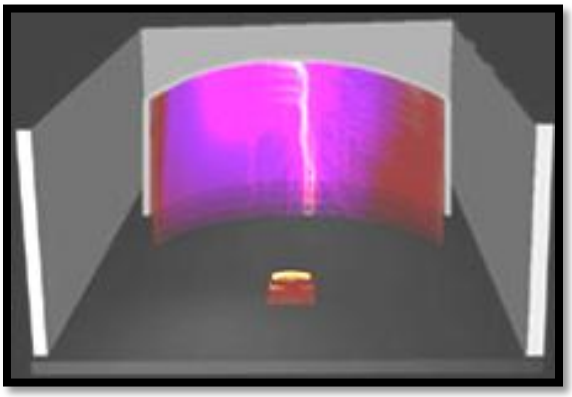

Figura 2.4 Trans-e: mi cuerpo, mi sangre, por Diana Domingues

En 1998, [5] plantea formas alternativas de interacción con los usuarios, ya que los modos manuales convencionales de la interacción tienden a reducir el cuerpo, al ojo y la mano. Debido a esto, se trabaja en una interfaz de incorporación a través de la respiración y el equilibrio acoplada a un chaleco, que sitúa la experiencia inmersiva en el propio cuerpo del participante. Estos trabajos utilizan dispositivos de montaje en la cabeza (HMD's) y cámaras de inmersión de cuerpo completo, además de un sistema de rastreo sónico en tres dimensiones (figura 2.5).

Figura 2.5 Osmose, por Char Davies

En los ambientes mencionados, el usuario es libre para desplazarse a través de un cuarto. Su movilidad es limitada por las paredes de la habitación, ya que por el mismo objetivo de estos mundos virtuales no es necesario transportarse fuera de estas dimensiones. Un punto importante de estos proyectos es la interfaz que permite que exista interacción, así como el rastreo utilizado.

Entrando en un área distinta al arte, como lo es la robótica y la tele-operación, la realidad virtual juega el papel de facilitador hablando en específico del desplazamiento de un robot, invocando a [6] del 2004. Dicho modelo consiste en manejar un robot de manera similar a como se conduce un carrito de supermercado. El desarrollo consta de la fusión de la operación a distancia y el sencillo manejo del carrito de compras, basado en el empuje y en cambios de orientación con las manos mientras se camina, a través de la realidad virtual inmersiva. El movimiento hacia adelante se efectúa con un cálculo de la velocidad de rotación de una caminadora por medio de una cámara, tomando un punto de referencia en el disco para esta tarea. Una segunda cámara registra los gestos de las manos del usuario para llevar a cabo un giro o un desplazamiento hacia atrás. La rotación puede ser hacia la derecha o hacia la izquierda dependiendo de la mano que se ha movido. El retroceso se registra al mover ambas manos a la vez (figura 2.6).

Figura 2.6 Rotación y retroceso registrados por medio de una cámara

Sin embargo, como menciona [6], los sistemas de movimiento libre acostumbran utilizar las manos para otras tareas dentro del ambiente y no para el manejo a través de este. Por esto se debe implementar un control de giro que libere a las manos de esta labor. La ventaja de este sistema es que se acerca en mayor medida a un desarrollo más intuitivo en cuanto a control, sin mencionar que en este trabajo se ha liberado al participante de las cintas que lo aprisionaban e incomodaban. Se puede ver que en el desarrollo tratado, las cámaras juegan un papel importante. A diferencia de los trabajos recopilados de [3], el rastreo del participante se lleva a cabo por métodos visuales, solucionando así errores referentes a malas interpretaciones por el rudimentario sistema de cables en el suelo.

Por medio de otros mecanismos de rastreo que evitan incomodar al usuario, se ha desarrollado un sistema inusual que simula un camino sin fin: [7], del 2005 (figura 2.7). Conformado por un grupo de cuatro losas móviles, se puede mover el usuario en cualquier dirección ya que dichas losas se acomodarán automáticamente. La circulación de los elementos móviles permite al participante caminar por un ambiente virtual mientras que su posición se mantiene constante [7]. El antes mencionado sistema de rastreo consta de un receptor ultrasónico fijo y de emisores en cada losa. Con esto se puede conocer la posición y dirección de movimiento del usuario y así los elementos móviles se acomodarán gracias a un sistema de control que confiere las instrucciones necesarias de manera inalámbrica (figura 2.8).

Figura 2.7 CirculaFloor

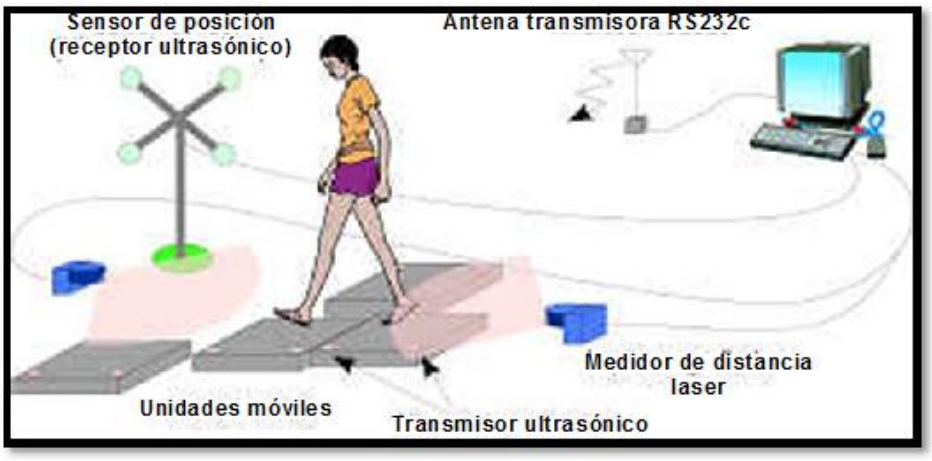

Figura 2.8 Distribución de los elementos del sistema CirculaFloor

Una mejoría que puede encontrase con respecto a [6], es que las manos ya no son utilizadas para dar giros. Como en la caminadora omnidireccional, el usuario se desplaza como si en realidad caminara por un mundo virtual.

Otra ventaja es reconocida casi de manera inmediata: todo es inalámbrico. Si el usuario no siente ningún dispositivo conectado a su cuerpo, la sensación de inmersión será mucho mayor, además de que se puede desplazar hacia cualquier dirección. Aunque es importante considerar el tiempo necesario para que las losas puedan colocarse. Esto limita al participante a realizar una caminata lenta o pausada.

Tal vez sea necesario implementar algún sistema de control en el que los dispositivos móviles puedan participar directamente, y no tener que mandar los datos a una computadora.

También, basado en el uso de un camino sin fin, en [8] del 2008 se ha desarrollado una cabina de inmersión, en la cual el usuario puede desplazarse por medio de una caminadora. Dicho elemento se localiza en el centro de la cabina y permite al espectador tener el rango de visión completo del mundo virtual mientras se desplaza (figura 2.9). El camino sin fin, en este caso representado por la caminadora (figura 2.10), permite olvidar la realidad de la cámara de inmersión para situarse dentro del medio simulado, ya que se tiene la hipótesis de que al permitirle al usuario “caminar” dentro del mundo virtual, la sensación de inmersión será aún mayor pero sin riesgo para él o el equipo. Además, se propone el uso de lentes para que puedan separarse las imágenes proyectadas. Con esto, se obtiene una imagen tridimensional y la persona aumenta aún más la sensación de inmersión.

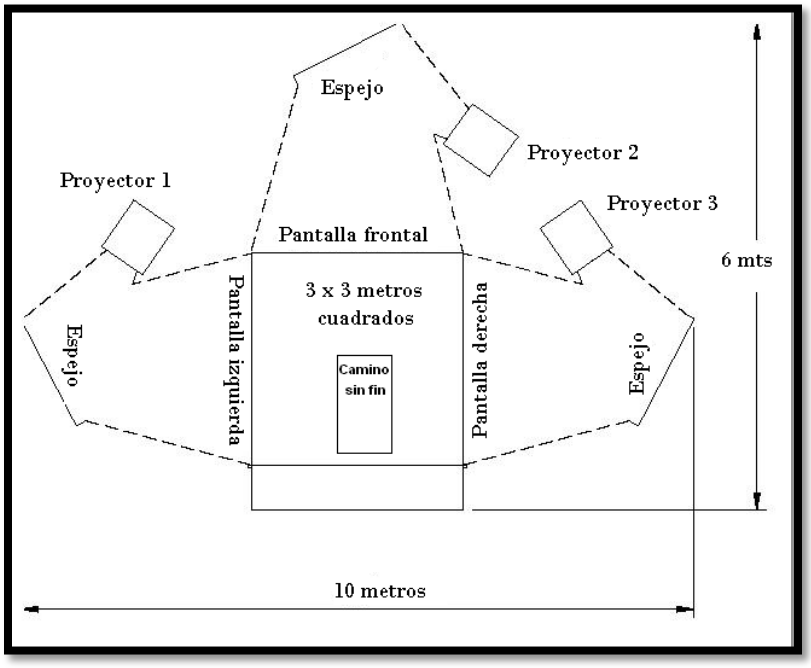

Figura 2.9 Esquema de la cabina

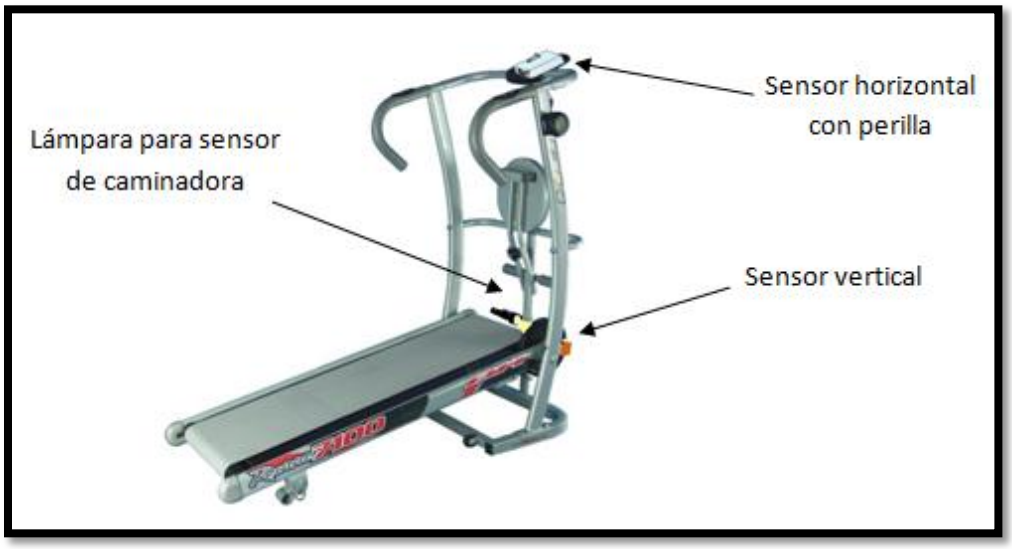

Figura 2.10 Caminadora y sus componentes

Una posible desventaja de esta cámara es que sólo se puede desplazar el usuario hacia adelante y hacia atrás de manera intuitiva. El usuario tampoco puede moverse hacia la derecha o hacia la izquierda, pero sin girar sobre su eje por medio de un dispositivo acoplado a la caminadora. Con esto puede restarse en cierto grado la inmersión.

A pesar de esto, una clara ventaja frente a otros trabajos, como la cinta omnidireccional, *UniPort* y *CirculaFloor*, es el contar con lentes que permiten una visión tridimensional en lugar de un dispositivo montado sobre la cabeza. También puede verse que a diferencia del *CirculaFloor*, el procesamiento es más rápido debido a que es menor la información ingresada al sistema.

Por parte de *Microsoft* en 2009, la tecnología *multi-touch* ha visto la luz en el desarrollo de sus nuevos ratones; con estos conocimientos emergentes, actualmente se tiene la oportunidad de manipular el contenido digital con una mayor destreza [9]. Aunque esto no está relacionado con la realidad virtual de manera directa, las técnicas utilizadas podrían adaptarse para sistemas de rastreo de una cámara de inmersión, por mencionar un ejemplo.

Si se ve el interior del *Orb Mouse* (figura 2.11), puede encontrarse una cámara de video en conjunto con sensores ópticos y ***LED's*** infrarrojos, los cuales valiéndose de una cúpula difusa y un espejo, capturan la imagen, misma que es filtrada hasta identificar la posición de los dedos. Tomando como ejemplo al *Side Mouse* (figura 2.12), la cámara registra la luz infrarroja que rebota en los dedos, interpretando así sus movimientos y posición. Gracias al texto de [9] y el personal asociado, se pueden apreciar las capacidades de estas novedades y su aplicación específica a esta área. Sin embargo, esto no debe ser una limitante.

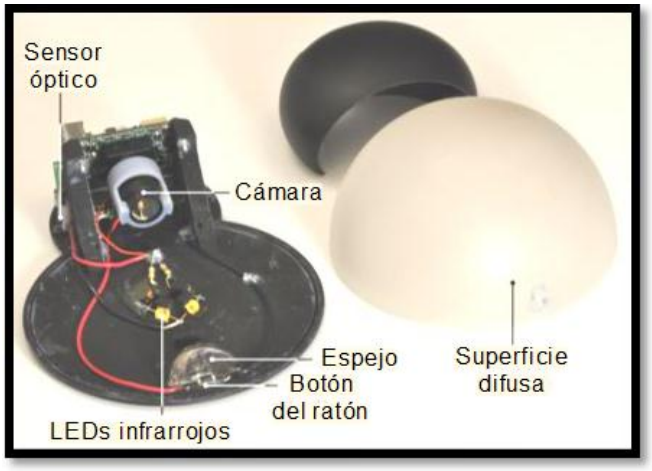

Figura 2.11 El interior del Orb Mouse

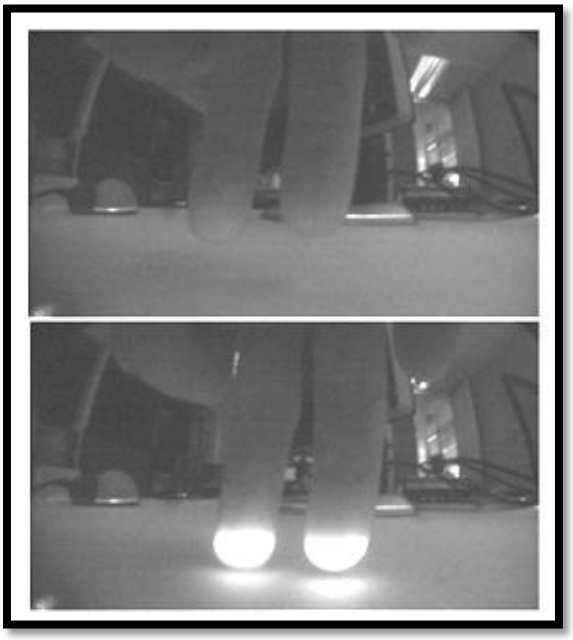

Figura 2.12 Detección de dedos del Side Mouse

Como puede verse, *Microsoft* ha estado ocupado en el desarrollo de estos dispositivos que basan su funcionamiento en la tecnología *multi-touch*. Dicha tecnología podría ser aplicada para resolver la problemática del desplazamiento por mundos virtuales, elaborando una cinta sensible a la presión del usuario o que sea capaz de registrar la fuerza que este mismo ejerce sobre la superficie. Otra opción viable es el utilizar la tecnología del *Side Mouse*. Si se colocase una cámara al nivel de la cinta o superficie a utilizar, se podría reconocer la luz que rebota en los pies e interpretarla como un paso, giro, etc.

La tecnología sensible al contacto, y aquella basada en cámaras de video y filtros de imágenes, pueden acabar con las problemáticas presentes en el trabajo de la cabina de inmersión de [8], como el registro de cambios de dirección. Además, el *CirculaFloor* podría probar este sistema de rastreo para sus losas y comparar los retardos, con lo cual podría mejorarse en caso de obtener mejores resultados.

Ahora bien, el movimiento a través de mundos virtuales debe ser intuitiva y no interferir con otras interacciones, pero muchas herramientas de navegación requieren de por lo menos del uso de una mano [10]. Se presenta la propuesta en 2009 de utilizar la base de balance de la consola de videojuegos *Wii* (figura 2.13), propiedad de *Nintendo*, para lograr el objetivo de liberar las extremidades de tareas no naturales. Esta tabla se encarga de registrar el peso del usuario. La

investigación va enfocada a darle una interpretación a las señales de la tabla y así conseguir un control de navegación (figura 2.14).

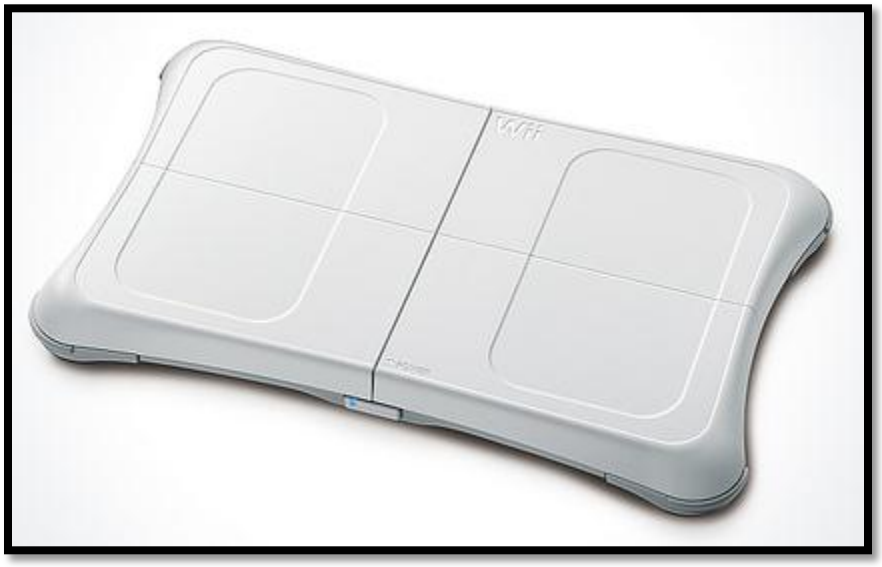

Figura 2.13 Base de balance, de Nintendo Wii

Figura 2.14 Funcionamiento de la base de balance

Hay que mencionar que esta plataforma viene a complementar al popular sistema de juego del *Wii* lanzado en 2006, y que basado en detección de movimiento por medio del *Wiimote* (figura 2.15). Este dispositivo cuenta con un sensor *MOT* (Multi-Object Tracking) de *PixArt Imaging*, como se explica en [11], y para funcionar de forma tan sensible, su resolución es de 1 Mega Pixel, pudiendo rastrear ordenadas desde 0 hasta 1023 grados en el eje horizontal y desde 0 hasta los 767 grados

en el vertical y se conecta a la consola por medio de Bluetooth, lo cual significa una gran ventaja al volverse inalámbrico. Gracias a un grupo de luces infrarrojas (cinco por lado) localizadas sobre o por debajo de la pantalla de televisión, se puede registrar la posición del usuario. Integrado esto último con los acelerómetros y giroscopios internos del *Wiimote*, además de los conocidos botones de un control de videojuegos, proporciona una herramienta de gran valor para la interacción con los títulos de *Nintendo* y una soltura que antes no se tenía en el área comercial.

Figura 2.15 Wiimote, de Nintendo

Entre los desarrollos más recientes, de los cuales el *WiiMote* fue precursor, se tienen dos sistemas de rastreo: los controles *PlayStation Move* de *Sony* del 2010 y *Kinect* de *Microsoft* del mismo año (figura 2.16). El primero de estos utiliza una cámara, también propiedad de la empresa, para registrar la luz de una especie de bombillo que cambia su color dependiendo de la aplicación que se esté tratando. Al igual que el mando del *Wii*, cuenta con una serie de sensores que permiten al sistema interpretar las acciones del usuario. Sin embargo, el mando de *PlayStation Move* está equipado de un acelerómetro que no sólo interpreta el movimiento y posición del periférico (que se coordina con la situación en el espacio que "lee" la cámara *PlayStation Eye*), sino que también calcula la distancia recorrida en el aire y el tiempo que emplea en hacerlo para interpretar la fuerza y velocidad del movimiento y trasladarlo al juego [12].

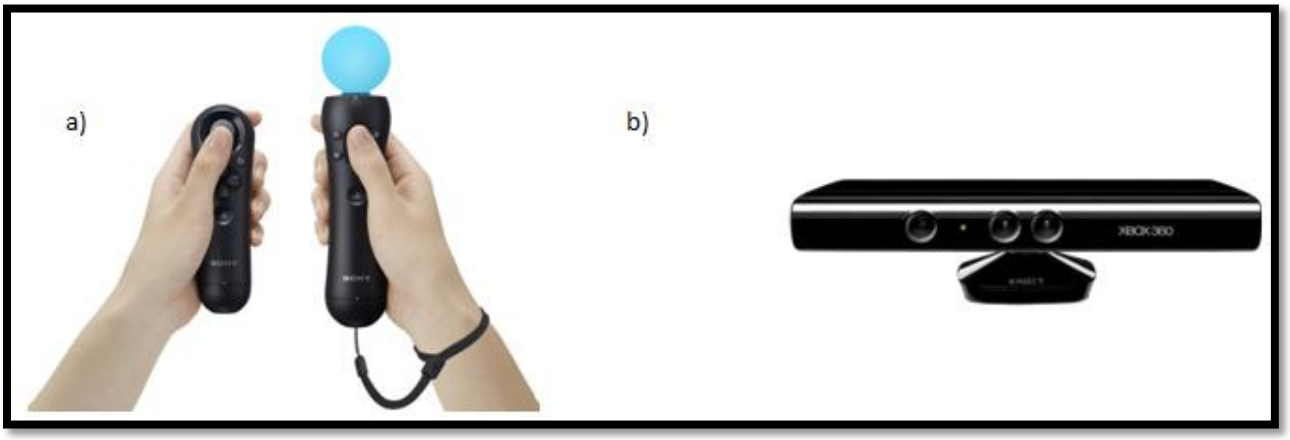

Figura 2.16 Controles del sistema PlayStation Move (a) y Kinect (b)

Pasando a un cambio más drástico en cuanto a la arquitectura del sistema, tenemos al *Kinect* para *Xbox 360* de *Microsoft*. Este revolucionario desarrollo incorpora cámaras, micrófonos y sensores para considerar los movimientos corporales como un mando para los videojuegos. Se proyecta una luz infrarroja sobre el área que la cámara visualiza, en forma de pulsos, con lo que puede obtenerse información como profundidad, desplazamiento o simplemente puede conocerse la posición del usuario. Muchos investigadores concluyen que la relevancia de este dispositivo radica en el software, mas no en el hardware, ya que no contienen nada fuera de lo común.

A partir de estos trabajos, se puede ver que el desplazamiento en ambientes virtuales también puede llevarse a cabo por movimientos de la vida diaria, sin estar hablando estrictamente de mover las piernas y caminar. Pero debe de mantenerse el objetivo de alcanzar un desarrollo que funcione a través de movimientos naturales y así alcanzar la máxima inmersión posible.

Considerando los trabajos referentes a la tecnología *multi-touch*, la base de balance puede retomar estos desarrollos y agregar aún más funciones a la base. Esto permitirá registrar una mayor cantidad de datos de los proporcionados hoy en día por sus cuatro sensores de presión.

Retomando el análisis de investigaciones referentes a los caminos sin fin, se puede mencionar a un dispositivo poco usual. La empresa *Virtusphere* se dedica a la fabricación de esferas huecas enfocadas a cumplir la función de camino infinito para la navegación en mundos virtuales. El sistema, patentado en 1998, consta de una esfera de tres metros de diámetro colocada sobre una base especialmente creada para que el dispositivo pueda girar en un mismo lugar, permitiendo una rotación libre en cualquier dirección de acuerdo a los pasos que dé el usuario [13]. La esfera registra

por medio de sensores colocados en la base estos giros y por medio de un *HMD* inalámbrico se proyecta al usuario el mundo virtual (figura 2.17).

Figura 2.17 Virtusphere

Este dispositivo tiene una enorme ventaja: permite un movimiento libre del usuario en prácticamente todo grado de libertad. La sensación de inmersión es muy buena y además el contar con un casco sin cables permite al usuario perderse en el ambiente y deshacerse de cualquier barrera que obstaculice el desplazamiento. Un punto que podría ser negativo es el uso de un dispositivo montado en la cabeza ya que a veces puede resultar molesto con el tiempo de uso, como en proyectos mencionados anteriormente.

En este desarrollo, el camino sin fin es más sencillo pero no por eso se vuelve menos preciso. A diferencia del *CirculaFloor*, se registra de manera más exacta el movimiento y más rápido, además de que el usuario ya no necesita cierta velocidad en sus desplazamientos. Si se considera al proyecto basado en la consola *Wii*, es fácil apreciar las ventajas referentes: movimiento intuitivo por el ambiente en lugar de permanecer en un punto.

Dentro de todos los trabajos anteriores, se pudo observar que existen herramientas de rastreo adecuadas para su caso correspondiente. Pasando desde medios mecánicos, algo obsoletos en la actualidad, hasta ópticos y con tratamiento de imágenes y de información compleja, las ventajas y desventajas que cada uno significa es una cuestión a estudiarse. Sin embargo, independientemente de la aplicación, existen puntos valiosos y también débiles en cada uno y que demandan interés.

Tomando como modelo de estudio al Kinect, vemos que realiza un procesamiento y registro muy bueno del usuario para detectar movimiento. Sin embargo, esto no sería posible sin una adecuada iluminación, o contraste, para que las cámaras capturen a la persona que juega. Por esto mismo, no podríamos utilizarlo dentro de la cabina de inmersión, o al menos no tal cual como se plantea su utilización con el Xbox 360.

Si ahora vemos a la Virtusphere podemos observar que el registro del movimiento del usuario está directamente ligado al movimiento de la esfera. Esto nos dice que siguiendo los cambios de la esfera se seguirán los de la persona en su interior. Ahora bien, existen varias formas de registrar este movimiento. Alguna podría ser mecánica, aunque entra en consideración la opción óptica.

Todas las consideraciones de esta especie deben de analizarse a fondo, ya que no se pretende seguir a un trabajo o basarse del todo en el desarrollo de alguna investigación. Más bien, se trata de encontrar herramientas adecuadas a las condiciones de las instalaciones y a los objetivos planteados.

Capítulo III. Marco teórico

3.1 Introducción

Es posible apreciar las limitaciones del trabajo de interacción con la realidad virtual, tomando algunos elementos de referencia, como son principios de funcionamiento de componentes electrónicos o propiedades de un programa. Además, definiendo la tecnología que se toma en cuenta para la comunicación, el despliegue y la traducción de señales, se facilita la comprensión y la posible mejora a futuro de la tesis.

A continuación se tratan aspectos generales de los factores que conforman al trabajo de investigación, dividiendo el conocimiento en tres grupos: software, elementos tangibles e interconexión.

3.2 Antecedentes

En el comienzo de este trabajo se hace clara la relación con la *realidad virtual*, ya que el sistema se ampara en muchas investigaciones basadas en sus principios. Por esto mismo, extensamente se describen sus raíces y las reglas que la rigen, además de su evolución y acercamiento hacia las condiciones del presente desarrollo. Cabe recalcar que no se pretende mencionar el inicio mismo de todo trabajo, pues se tendría que definir desde el ábaco y las matemáticas, pasando por la electrónica y la concepción de la computadora moderna.

Más adelante se centra el texto a los *microcontroladores*, tomando como punto fuerte en este apartado su relación con los puertos de una computadora y teoría de transmisión de información. Sin tomar partido dentro de la circuitería básica, incluyendo resistencias, capacitores u otros componentes, se expone la conectividad de este dispositivo a los diferentes elementos necesarios para su correcto funcionamiento.

Entrando a otra sección crítica del trabajo, se hace mención de los puertos de un equipo de cómputo, tanto serie como paralelo, aunque concentrando mayor atención a la conectividad por medio del USB. Como bien puede apreciarse, existen varias ventajas al utilizar el puerto USB tanto en velocidad como en facilidad de conexión.

3.3 Teoría

3.3.1 Realidad virtual

Partiendo de la división de este término se puede estudiar mejor su significado. Algunos autores definen la palabra *virtual* como “lo que es en esencia o efecto, más no en realidad” [14], donde se aprecia que la virtualidad no es un término exclusivo de las ciencias computacionales. Para ejemplificar lo anterior, se presenta lo siguiente: Cuando una computadora requiere de mayor memoria RAM, esta puede expandirse de manera virtual utilizando parte del disco duro. Con esto, la capacidad de la memoria RAM ha crecido sin que físicamente (o realmente) se tenga una memoria de dicho volumen.

Ahora bien, el concepto de realidad es más complicado de exponer. Existen una infinidad de definiciones a este término. Se tiene como *realidad* a la “calidad o estado de ser real. Algo que existe independientemente de las ideas que puedan afectarle. Algo que constituye a una cosa real o concreta y que puede distinguirse de algo meramente aparente” [14]. Puede sugerirse a la *realidad* como un universo o espacio que existe y que se puede experimentar.

Teniendo las definiciones primordiales, queda definir el término en su constitución. Cabe mencionar que la extensión del concepto realidad virtual va más allá de las computadoras, ya que puede considerarse a una novela o película ejemplos de realidades “no reales” que exponen a los lectores o espectadores mundos ficticios.

Una de las interpretaciones más utilizadas, hablando propiamente del área de sistemas, es el considerar a la *realidad virtual* como “un conjunto de elementos artificiales, generados con recursos computacionales, que simulan al mundo real o parte de este” [15], ya que su tarea es engañar al o los espectadores. Desgraciadamente, aún no se puede asegurar qué es real si se toma en cuenta que los sentidos son los receptores de todo lo que nos rodea, lo que existe, y que además pueden ser burlados.

Una interpretación menos ambigua podría ser la siguiente: la realidad virtual es aquella “realidad” creada por medios computacionales y que se encarga de proporcionar al usuario un medio de despliegue de dicho mundo y herramientas de interacción.

3.3.2 Los elementos de la realidad virtual

Para hacer mención del concepto “realidad virtual” se deben tener en cuenta algunos puntos esenciales. Primeramente debe contarse con un contenido, es decir, la información que se proporciona a través de algún medio (sea cual sea). Desde los textos literarios es sencillo percibir ese universo creado por el autor, producto de la imaginación y la inventiva. Sin este punto clave, no existiría aquel mundo ficticio encargado de proporcionar una *realidad* a cualquier espectador. Se podría decir que gracias al mundo virtual se tiene una descripción de una colección de objetos en un espacio, además de las reglas y relaciones que gobiernan a estos objetos.

Ahora bien, otro punto crucial es el de la *inmersión*. El usuario debe de sentirse en algún aspecto dentro de este ambiente simulado, ya que de lo contrario ¿cómo podría llamarse realidad? Existen dos clases de inmersión: física y mental. Mientras que el primer tipo de inmersión sucede cuando son estimulados los sentidos por algún medio tecnológico, el estado de inmersión mental se refiere a involucrar emociones o a contar con la sensación de presencia dentro del ambiente simulado, lo cual puede asemejarse al efecto que causa una novela al envolver al lector en su mundo. Esto se logra sin utilizar ningún dispositivo extra.

Como tercer y último apartado se tiene a la *interacción*. Es la parte encargada de relacionar al usuario con los elementos sintéticos. Mientras se esté utilizando un mundo virtual, o navegando en él, debe existir alguna forma de comunicarse con este entorno. Generalmente, el ratón o el teclado (comúnmente, también las palancas de juego y controles) cumplen con esta tarea. Por medio de estos dispositivos es posible desplazarse por los escenarios, sujetar objetos, desencadenar ciertos eventos programados, etc. De la misma manera, se puede encontrar una retroalimentación hacia el usuario por otros medios. Habitualmente las pantallas de computadora, u otro dispositivo de salida similar, proporcionan la información a través del sentido de la vista, pero existen también dispositivos hápticos que se encargan de facilitar al usuario elementos tales como fuerza o presión, consecuencia de sucesos en el mundo virtual.

En forma de esquema, se pueden unir estos tres componentes y conformar las llamadas tres *I’es* de la realidad virtual (figura 3.1).

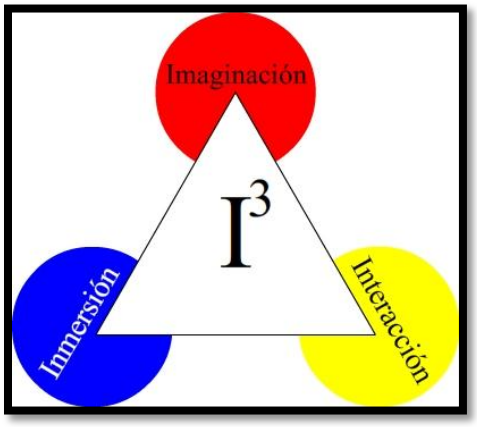

Figura 3.1 Tres I'es de la realidad virtual

3.3.3 Sistema de despliegue

Puede no ser considerado al sistema de despliegue como un componente indispensable, pero si uno bastante requerido. La diferencia entre "indispensable" y "requerido" es fácil de señalar si se tienen en cuenta los siguientes puntos:

- El ambiente simulado existirá, a pesar de no haber observadores de éste.
- La navegación puede valerse de la imaginación (por mencionar un ejemplo).
- La sensación de inmersión baja en gran medida, pero no en todos los casos se vuelve nula.

En especial, el último elemento enlistado juega un papel definitivo en cuanto a considerar un sistema de realidad virtual propiamente digno de su definición. Sin embargo, no se entrará en discusión acerca de este punto, ya que el trabajo no está encaminado a ello. Principalmente, se hace mención de esta cuestión para recalcar lo importante que es contar con un sistema de muestreo del ambiente simulado. El fin de un sistema de despliegue es mostrar al usuario por medio de proyecciones, pantallas de equipos de cómputo (ventanas[1]), móviles, etc., para que sea posible conocer el entorno, sus interacciones posibles, texturas, colores y todo lo que percibimos en nuestro mundo real con la vista. Los dispositivos de montaje sobre la cabeza son un buen ejemplo de sistema de despliegue, aunque está claro que suele utilizarse también como sistema de rastreo y también en algunos casos se acoplan elementos para la navegación. Aquí se muestran algunas de las primeras patentes de estos dispositivos (figura 3.2):

[1] Cuando el mundo virtual se muestra en la pantalla de una computadora, comúnmente se le llama "ventana" por la similitud con una ventana real, donde es posible apreciar desde el otro lado algún escenario diferente.

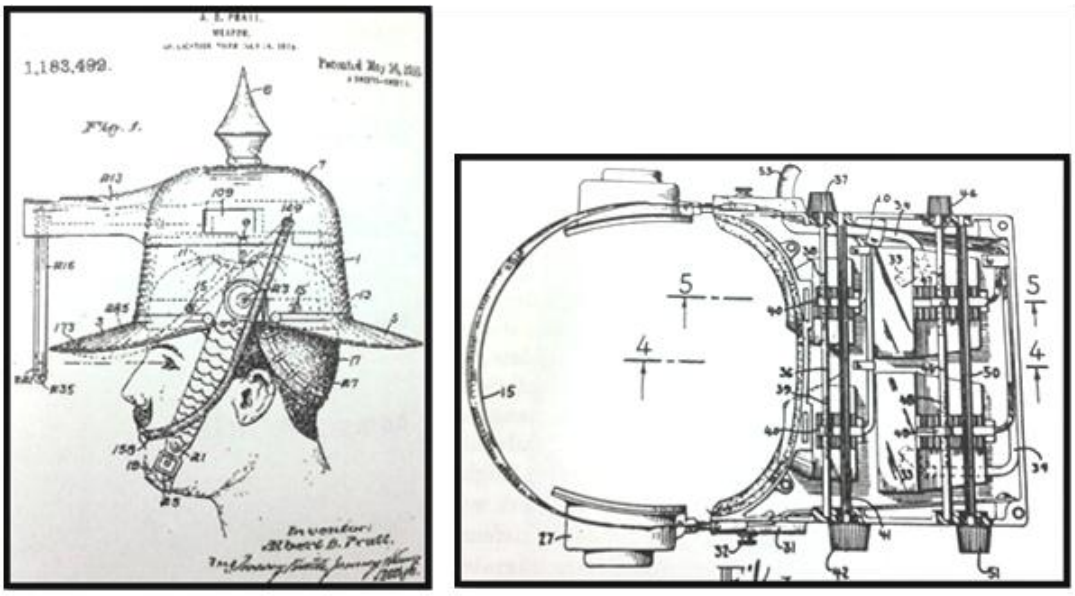

Figura 3.2 Patentes más antiguas de dispositivos montados en la cabeza (1916 y 1960)

En una caverna de inmersión, el despliegue se lleva a cabo gracias a la proyección, tanto frontal como lateral, de las paredes de la instalación. Si no se cuenta con un buen despliegue en relación nuestra posición en el mundo y nuestras acciones e interacciones, la inmersión va perdiéndose.

3.3.4 Sistemas de navegación

Para los ambientes virtuales, existen muchas formas de interacción con los usuarios en cuanto al cambio de posición. Desde los dispositivos más comunes de las computadoras como los teclados, ratones y palancas de juego, hasta elementos complejos y enfocados a tareas específicas, como son cámaras de simulación de aviones o controles de manejo de automóviles (volantes, pedales).

Uno de los conceptos que se necesita definir primeramente, es el de *grado de libertad*. Un grado de libertad es el número mínimo de elementos cambiantes que ayudan a definir el sistema mecánico de un cuerpo. Son utilizados para el cálculo de movimiento, velocidad, aceleración, etc. Dicho en otras palabras, un grado es equivalente a la posibilidad de cambio con respecto a un eje en el espacio. Por ejemplo, una ruleta solo puede girar sobre un eje. Se puede decir entonces que la ruleta tiene un grado de libertad. Si se hablara de un elevador, también sería posible apreciar un grado sobre el eje vertical. Ahora bien, tomando para su análisis a un automóvil, se pueden identificar dos grados: adelante/atrás y el giro sobre el eje vertical. Al caminar podría existir una variada

combinación de cambios de posición y/u orientación. Normalmente podemos andar hacia adelante y hacia atrás, de un lado al otro y girar sobre nuestro propio eje. Lo anterior puede verse como movimiento en tres grados de libertad.

En general, los dispositivos para interactuar deben de contar con un elemento básico: permitir el desplazamiento en por lo menos un grado de libertad. Los sistemas de navegación nunca se han conformado con esto y siempre se enfocan en proporcionar, además de la posibilidad de moverse por ambientes simulados en tres o seis grados (figura 3.3), una sensación intensa de inmersión. Es posible encontrar ejemplos, como la *VirtuSphere* o las cavernas (CAVE), que crean atmósferas que envuelven a los usuarios y proporcionan al mismo tiempo mecanismos de rastreo y de navegación (figura 3.4). En la actualidad, las consolas de videojuegos también han aportado además de herramientas de navegación, como son los botones y palancas, ambientes de inmersión.

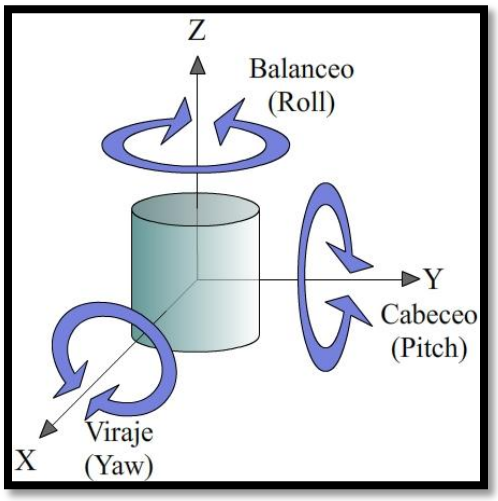

Figura 3.3 Esquema de 6 grados de libertad

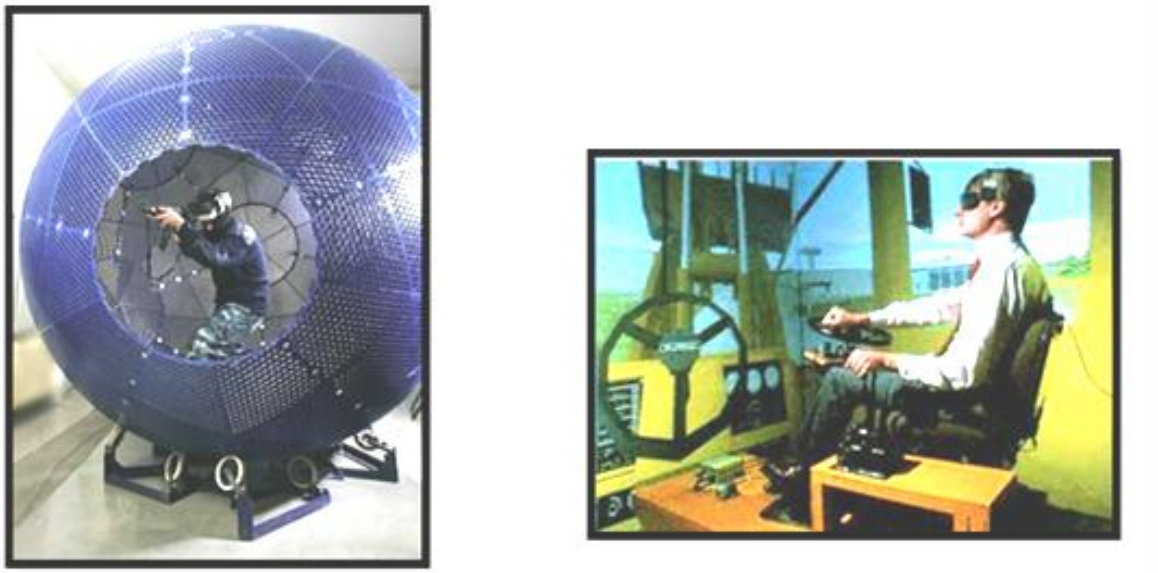

Figura 3.4 Sistemas de navegación que incorporan inmersión de alto grado

Por medio de nuevos y diversos sensores incorporados, y a la erradicación de botones en algunos casos, la navegación no se ve alterada y la sensación de pertenencia a mundos artificiales se acrecienta.

3.3.5 Sistemas de rastreo

Son aquellos sistemas que se encargan de registrar la posición, orientación o cambio de éstas en el espacio. Existen varias maneras de llevar a cabo esta tarea, ya sea por distintos sensores o por diferentes técnicas. Por lo general se hace una división de éstos dispositivos en cinco categorías: Mecánico, óptico, ultrasónico, electromagnético y sin origen, mismas que serán estudiadas más a fondo a continuación.

Mecánico

El sistema mecánico resulta ser el más rudimentario pero el más rápido en cuanto a respuesta (o de menor latencia), esto debido a que sucede en tiempo real por elementos mecánicos, como pueden ser engranes, palancas, poleas, etc. (figura 3.5) Son útiles para dar seguimiento a los movimientos y cambios de orientación de secciones del cuerpo, aunque siempre se recomienda utilizarlos sólo para el rastreo de una parte, como puede ser la cabeza o algún brazo. Estos dispositivos cuentan con algunas desventajas, tales como la limitada capacidad de movimiento para

el usuario y la oclusión que sucede entre brazos mecánicos si es que se intenta rastrear al mismo tiempo dos o más partes del cuerpo. Esto ocurre, por ejemplo, cuando se desea mover el brazo a una posición ocupada por el brazo de la cabeza y estos chocan.

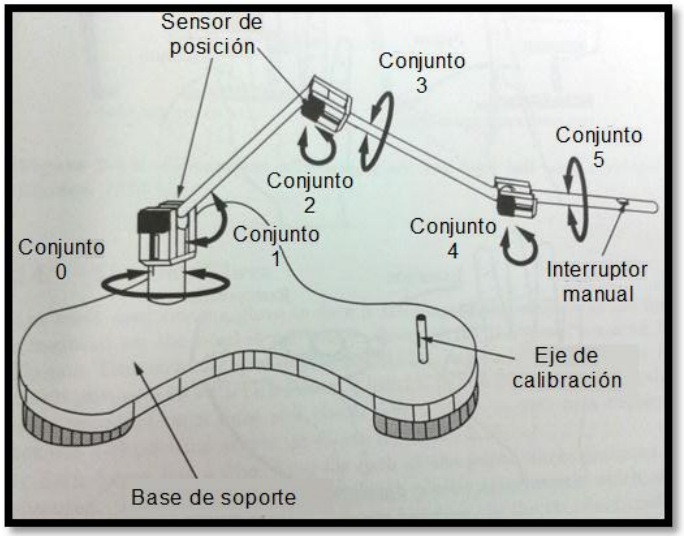

Figura 3.5 Ejemplo de dispositivo de rastreo mecánico

La tecnología de los sistemas de rastreo mecánicos también puede adaptarse para detectar el movimiento de los dedos, de la mano u otra extremidad creando dispositivos que contengan poleas y cables y que por medio de estos se haga una interpretación a nivel máquina de la información y se refleje en el mundo virtual. Cabe mencionar que esto ya sale de los llamados sistemas de rastreo, ya que no proporciona datos que sugieran alguna posición u orientación.

Óptico

Los sistemas ópticos son de los sistemas más utilizados debido a su alta velocidad de respuesta y a la facilidad con la que pueden llegar a implementarse. Puede llevarse a cabo con cámaras de video montadas en el entorno y que registren la posición del usuario por medio de algún elemento que distinga de la imagen tomada (usualmente son luces infrarrojas o material reflejante). Una desventaja de este sistema es que puede detectarse al usuario sólo en un plano, es decir dentro de dos ejes coordenados. Para tener un posicionamiento en tres dimensiones deben utilizarse por lo menos dos videocámaras. Otro punto en contra es que es susceptible a oclusión, por lo que deja de proporcionar la información adecuada si llega a haber algún elemento entre la cámara y el objeto a observar.

De igual manera, es posible detectar la posición y orientación de una persona montando la cámara en el sujeto en cuestión y colocando en el entorno luces u otro componente que ayude a una fácil división de los puntos de referencia.

Ultrasónico

Los sistemas de rastreo ultrasónico utilizan sonido a muy alta frecuencia (rebasando los 20,000 Hz) para que sean indetectables al oído humano, pero sea posible registrarlos por medio de micrófonos posicionados estratégicamente. Generalmente se requieren tres micrófonos para localizar a un elemento en un espacio tridimensional (el mundo real) y un dispositivo que genere la frecuencia que será detectada (altavoz). Al medir el retardo del sonido a cada micrófono, y tomando en cuenta la velocidad del sonido, es posible calcular la distancia de la fuente a los tres destinos. Esto resulta útil para encontrar en el espacio al elemento a registrar.

Se necesita tener control de los ruidos ambientales y estar aislado de ellos en el mayor grado posible, esto debido a la sensibilidad de los micrófonos. Por ejemplo, si llegan a sonar unas llaves en las cercanías del sistema, podría interpretarse mal la información y el rastreo sería erróneo. Además, cuenta con la misma desventaja de los sistemas ópticos: la oclusión, ya que si algún elemento se interpone en la vía de comunicación, el resultado se verá alterado.

Electromagnético

Como lo dice su nombre, este método ó sistema se basa en el cálculo de la posición por medio de campos magnéticos. Se utilizan varias bobinas que generan campos de bajo nivel en los límites del área de acción. Luego, un elemento receptor registra la información y es interpretado como un punto en el espacio. Además de poderse detectar la orientación del usuario gracias al censo de la fuerza de los campos.

Cabe mencionar que existe una clara desventaja en cuanto a la generación de los campos magnéticos, ya que hay que esperar a que sean creados. Y también está la cuestión de los efectos de cuerpos metálicos dentro del entorno de movimiento de la persona. Si existe algún elemento de metal afectará los campos creados y la información carecerá de significado útil. Su mayor ventaja es que no importa si algo se interpone entre el transmisor y el receptor, mientras esto no sea de metal. Además, debe de utilizarse a poca distancia los transmisores y receptores ya que los campos de bajo nivel operan razonablemente a menos de un metro.

Sin origen

En este caso no existe un rastreo espacial tal cual como se ha visto anteriormente, sino que este sistema se encarga de saber la orientación del usuario o de alguna sección del cuerpo. Suelen utilizarse elementos como giroscopios[2] o factores como el campo magnético de la tierra, además de acelerómetros. No se necesitan sensores como los antes mencionados, con una fuente y un destino, sino solo algunos dispositivos que indiquen algún tipo de movimiento relativo. Debido a lo anterior, debe de tomarse en cuenta la orientación inicial de la persona que navega el mundo virtual.

Al encontrar el sistema o los sistemas más adecuados a las condiciones de trabajo o a las partes que se desea localizar o rastrear, el rango de error deberá de ser muy pequeño. Suelen utilizarse dos sistemas, como puede ser la unión de dispositivos sin origen con videocámaras para encontrar los cambios de posición.

3.3.6 Elementos de programación

Panda3D

Es un motor de videojuegos que permite incluir gráficos, audio, detección de colisiones y además está encargado del renderizado[3] por medio de avanzadas técnicas. Permite desarrollar juegos tanto en lenguaje Python (se hablará de él más adelante) como en C++ y permite llevar a cabo un monitoreo de desempeño, donde es apreciable el gasto de recursos del computador.

Los modelos que son utilizados en Panda3D son creados previamente en programas de modelado como pueden ser Blender o Maya, y cargados en el ambiente virtual por medio de coordenadas cartesianas. Además pueden modificarse otras variables de los objetos como pueden ser la rotación y la escala.

Una característica importante es que ofrece soporte a dispositivos de entrada / salida, como el teclado y el ratón, ya que se tratan los eventos que ocurren a consecuencia de un acción de estos elementos.

[2] El giroscopio es un dispositivo mecánico formado esencialmente por un cuerpo con simetría de rotación que gira alrededor de su eje de simetría. Cuando se somete el giroscopio a un momento de fuerza que tiende a cambiar la orientación del eje de rotación su comportamiento es aparentemente paradójico ya que el eje de rotación, en lugar de cambiar de dirección como lo haría un cuerpo que no girase, cambia de orientación en una dirección perpendicular a la dirección "intuitiva".
[3] Se le llama *Renderizado* al proceso de generar una imagen 2D a partir de modelos tridimensionales por medio de cálculos complejos, tomando en consideración aspectos de iluminación y sombreado.

Panda3D fue desarrollado por Disney, pero actualmente se encuentra liberado (a partir del 2002) bajo licencia BSD. La Universidad Carnegie Mellon ha contribuido al pulir el motor y agregarle documentación y funciones de alto nivel.

Lenguaje Python

Es un lenguaje interpretado de alto nivel, orientado a objetos, que ofrece una sintaxis bastante sencilla y entendible. Posee Licencia de Código Abierto (Python Software Foundation License).

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado ligadura dinámica de métodos). Además es posible correr programas (o mejor dicho, interpretar código) de manera dinámica. Esto quiere decir que las instrucciones o comandos son suministrados en tiempo real en su modalidad de consola. Sin embargo, también es posible guardar las líneas de código en un archivo con extensión *py*, y dejar que Python lo interprete y corra en el sistema.

Comunicación en red

Existen diferentes maneras de llevar a cabo la transmisión de información entre computadoras. Estas formas de comunicarse son llamadas protocolos. Es posible encontrar dos tipos de protocolos para el transporte de datos: TCP y UDP (figura 3.6).

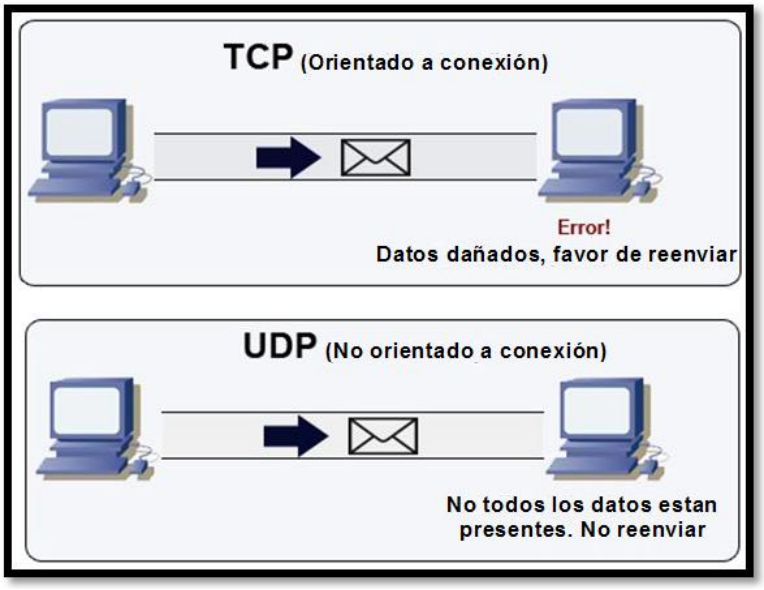

Figura 3.6 Protocolo TCP y UDP

Los protocolos de comunicación tomados en cuenta para los equipos conectados en red, que son parte de la instalación de la cabina de inmersión, son TCP y UDP. El protocolo TCP trabaja a nivel de la capa de transporte del modelo OSI, es orientado a conexión y cuenta con verificación por parte del receptor del mensaje. Si existe algún dato corrupto, se solicita la información de nueva cuenta. Esto se logra gracias a la recuperación de errores mediante los campos de secuencia y reconocimiento en el encabezado TCP. Si existe alguna falla, es enviado por parte del receptor el valor del segmento deseado para la retransmisión. De esta forma pueden recuperarse los segmentos erróneos y en ciertos casos, hasta todos los segmentos de la transmisión (figura 3.7).

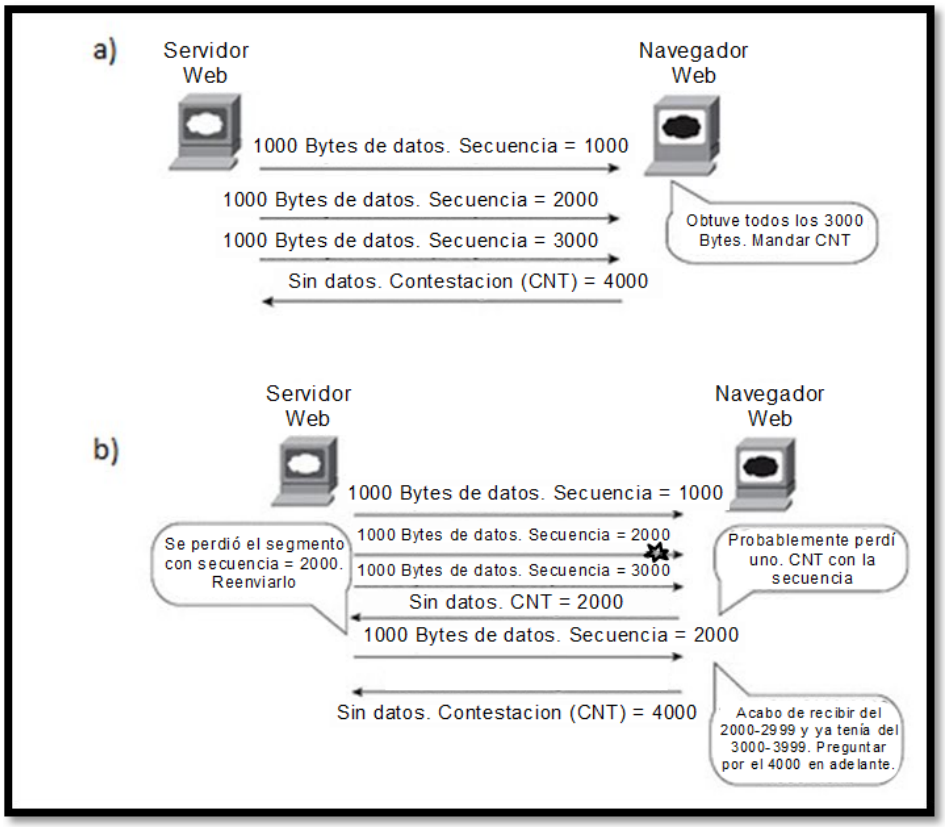

Figura 3.7 Recepción de segmentos con protocolo TCP a) sin errores y b) con errores

Por su parte, el protocolo UDP no está orientado a conexión y no requiere de respuesta. Esto es un punto débil ya que pueden existir errores en transmisión que no son registrados. Además, el receptor no conocerá los datos de la computadora emisora.

Mundo Virtual

Uniendo las características anteriores es posible llegar a definir lo que es un escenario virtual. No se trata más que de un mundo que “virtualmente” está ahí o que conceptualmente existe, conformado por los elementos básicos de la realidad virtual. En su mayoría, son regidos por leyes naturales. Sin embargo esto no es una regla, ya que pueden existir realidades fantásticas y llenas de elementos sobrenaturales. Ahora bien, ya que se utiliza una cabina de inmersión es necesario que el mundo creado también ayude a forjar esta ilusión.

Escenario de Prueba (Ambiente Boscoso)

Por medio de la observación de ambientes naturales, tanto de zonas templadas como en aquellas altamente húmedas, puede conseguirse aislar la "esencia" de un ecosistema y usarla de modelo en la creación de uno más personalizado. Haciendo mención específicamente a los entornos boscosos, existen factores que suelen repetirse, como la flora (arce, roble, fresno, haya) y la fauna (ardilla, lobo, oso, venados, zorro), con las variaciones correspondientes a su localización geográfica. Además están los elementos no vivos, como pueden ser las rocas, montañas o incluso nieve y agua.

A partir de esto se puede decir que un bosque contiene especies animales de sangre caliente, por lo general con pelaje, en el caso de mamíferos, y con vegetación de gran tamaño, exceptuando arbustos o hierbas, y de hoja caduca o perenne también dependiendo de su clima.

3.3.7 Elementos del dispositivo

PicBasic Pro

El compilador PicBasic Pro (PBP) es un lenguaje de programación de nueva generación que hace más fácil y rápido el programar microcontroladores Pic de Microchip Technology. El lenguaje Basic es mucho más sencillo de leer y escribir que el lenguaje ensamblador Microchip. Aunque no existen los mismos elementos que en lenguajes como C++ o Java, PicBasic Pro otorga una variada cantidad de funciones que facilita la comunicación del microcontrolador con otros dispositivos físicos, como pueden ser pantallas LCD, motores a pasos, servomotores e incluso contiene funciones que permiten llevar a cabo comunicación por puertos.

El PBP es similar al "BASIC STAMP II" y tiene muchas de las bibliotecas y funciones de los BASIC STAMP I y II. Como es un compilador real los programas se ejecutan más rápido y pueden ser mayores que sus equivalentes STAMP.

PBP no es tan compatible con los BASIC STAMP[4] como el compilador PicBasic pudiera llegar a ser con el BS I, esto debido a ciertos cambios en sentencias de control (se ha agregado IF...THEN...ELSE...ENDIF en lugar de IF… THEN (GOTO) de los Stamps).

[4] *BASIC Stamp* es un microcontrolador que posee un intérprete especializado de BASIC que se encuentra en su memoria ROM. Este microcontrolador es fabricado por Parallax, Inc.

Ya teniendo el programa escrito y después de configurar al compilador y obtener el archivo *.hex*, se puede proceder a programar el microcontrolador. Para esto hace falta contar con el dispositivo físico, y este mismo es independiente del PBP. Como último punto cabe hacer mención de su compatibilidad. PBP puede utilizarse bajo Windows 98/Me/NT/2000/XP/Vista/7, con lo que virtualmente podríamos considerarlo adecuado para cualquier sistema operativo de Microsoft que siga manejándose. Además, llega a soportar más de 300 tipos de microcontroladores PIC.

Visual C++

Desarrollado por Microsoft, Visual C++ es una adaptación del conocido lenguaje C++. Sin embargo, existen comandos que no son reconocidos entre estos lenguajes, como por ejemplo se puede mencionar al apuntador. Recordando la definición y utilización de un apuntador en C++, es posible apreciar que este tipo de variable se expresa con un asterisco (*), mientras que en VC++ se tiene que hacer esto con otro símbolo (^).

Visual C++ proporciona un ambiente gráfico de trabajo, en donde es sencillo programar y se tiene un fácil acceso a cada sección del sistema y varios comandos que son útiles al momento de compilar o construir un proyecto.

Puede encontrarse dentro del paquete de desarrollo de Visual Studio y dependiendo de la versión, también pueden encontrarse variaciones en la escritura de algunas instrucciones. Por esto, siempre es importante leer la documentación para actualizar o importar un programa entre versiones.

Caminadora

Como su nombre lo indica, es un dispositivo que permite caminar a su usuario en línea recta. Por medio de un camino o banda sin fin, generalmente colocada sobre rodillos alineados paralelamente, es capaz de simular un tramo amplio para poder ser recorrido en sitios cerrados en lugar de sitios abiertos y de gran extensión. Sus principios son muy evidentes, ya que parte de ruedas, acción y reacción de fuerzas y efectos gravitatorios (figura 3.8).

En inicio, las caminadoras siguen cierto estándar. La gran mayoría cuenta con elementos electrónicos que regulan velocidad, ángulo de inclinación del camino, resistencia y que además contabilizan los kilómetros recorridos y las calorías eliminadas. Sin embargo, para efectos de la tesis todos estos elementos extra no forman parte del prototipo.

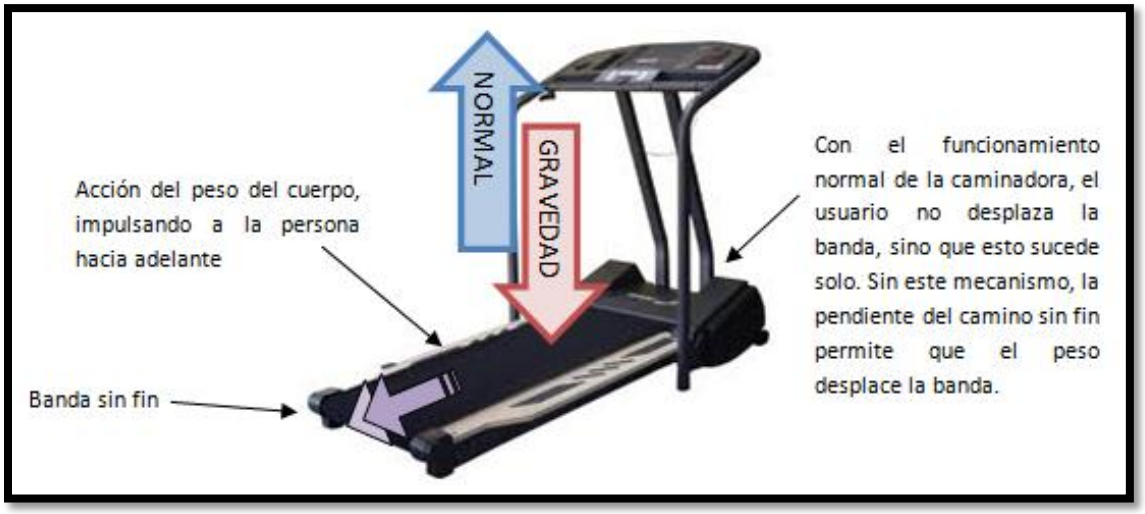

Figura 3.8 Principios de una caminadora

Microcontroladores

Los elementos encargados de ligar operaciones lógicas con impulsos o respuestas reales. Un microcontrolador es un circuito integrado. Esto quiere decir que en su interior es posible localizar las tres unidades funcionales de un computador: unidad central de procesamiento, memoria y periféricos de entrada y salida.

En el trabajo actual se recurre a la familia de PIC's, de tipo RISC, ya que cuentan con las características adecuadas (número apropiado de pines de entrada y salida), se tiene un programador universal con el cual es compatible y además información referente a su configuración.

Puertos

Son los medios de comunicación entre dos mundos: el real y el lógico. En otras palabras, es una interfaz que permite que exista una transferencia de datos de manera bidireccional de algún dispositivo o elemento físico al equipo computacional. Los puertos de mayor velocidad de transferencia resultan ser aquellos más cercanos a la tarjeta madre de la computadora, como el puerto PCI y PCI-Express. Ahora bien, durante la evolución de los equipos han surgido maneras de clasificar estas conexiones.

La división que se maneja a nivel de puertos es realizada al valorar la cantidad de información transmitida por unidad de tiempo. Debido a esto, es posible clasificarlos como:

- Puertos Paralelos. Son capaces de mandar varios bits a la vez debido a que cuentan con varias terminales o pines que transmiten sin necesidad de esperar que uno de ellos haya finalizado. Con esto, los datos no requieren de grandes velocidades para transmitir grandes cantidades de información, sin embargo esto siempre es deseable.
- Puertos Serie. Como su nombre lo indica, transmite los datos de manera serializada, es decir, uno tras otro. Uno de los defectos de los puertos serie iniciales era su lentitud en comparación con los puertos paralelos. Sin embargo, con el paso del tiempo, han ido apareciendo una multitud de puertos serie con una alta velocidad que los hace muy interesantes ya que tienen la ventaja de un menor cableado y solucionan el problema de la velocidad con un mayor apantallamiento. Son más baratos ya que usan la técnica del par trenzado; por ello, el puerto RS-232 e incluso multitud de puertos paralelos están siendo reemplazados por nuevos puertos serie como el USB, el Firewire o el Serial ATA.

Puerto USB

El puerto USB posee numerosas características. Algunas de ellas se enlistan a continuación:

- Un puerto USB permite conectar hasta 127 dispositivos.
- Se considera un estándar en las computadoras de última generación.
- Es totalmente Plug & Play, es decir, con sólo conectar el dispositivo éste es reconocido e instalado (sólo se requiere contar con los controladores correspondientes).
- Presenta una alta velocidad de transferencia en comparación con otros puertos (hasta 60 MB/s, en comparación con otros puertos serie o paralelo de 1 Mb/s.
- A través del cable USB es posible alimentar dispositivos externos.

3.4 Materiales y métodos

Los materiales que se utilizan son los siguientes:

- Cable USB
- Componentes necesarios para armar el circuito de conexión por el puerto USB y la circuitería requerida para los sensores utilizados, como pueden ser capacitores, resistencias, LEDs, botones, etc.
- Tres equipos de cómputo conectados en red
- Cabina de inmersión
- Caminadora

El método principal (impreso en el mismo título del desarrollo) que se utiliza es la comunicación a través del puerto USB. Ya son protocolos determinados que deben de seguirse, por lo que es considerado un método estándar.

Además, entra en consideración en este apartado la comunicación en red de los equipos computacionales. Dicha comunicación se lleva a cabo por medio de sockets UDP. La transferencia de información por este medio permite la actualización de las pantallas de acuerdo al movimiento o giro del usuario que utiliza la caverna. Debido a esto, entra a definirse esta comunicación como herramienta indispensable para llevar a cabo el funcionamiento de la cabina de inmersión y también forma parte primordial del sistema de despliegue de las tres ventanas al mundo.

Contando con las bases de los conocimientos a utilizar se puede realmente ligar un apartado con otro, hallando el canal de comunicación indicado, el protocolo, el comando, etc.

Debido a las grandes cantidades de información acerca de la realidad virtual, se puede acercar uno de mejor forma a sus principios y comprender qué es lo que se necesita primero, luego que paso debería seguir, y así sucesivamente, cumpliendo con un trabajo que utilice la realidad virtual tal cual y como se define, con sus partes bien determinadas y un razonamiento dirigido a lo que se necesita mejorar o implementar.

No debe pasarse por alto que la información plasmada en el escrito es la parte medular del trabajo de investigación, ya que no solo cuenta con los conceptos primordiales para un desarrollo fundamentado, sino que se relaciona directamente con las pruebas durante el crecimiento de la tesis.

Capítulo IV. Desarrollo

4.1 Introducción

El desarrollo ha sido dividido en tres secciones que engloban a la totalidad de la tesis: Ambiente Virtual, Comunicación USB y Sensores. La primer parte une lo respectivo al funcionamiento de la realidad virtual, desde el desarrollo del mismo ambiente hasta la comunicación requerida para la navegación dentro de la cabina de inmersión, pasando a través de aspectos de transmisión de datos por red (software).

El segundo apartado cubre lo relacionado a los circuitos, la relación de los módulos y acoplamiento de los sensores tanto a las tablillas como a la caminadora misma, incluyendo su posicionamiento, las mediciones que realiza, interpretación, etc.

La última parte cubre los elementos críticos para llevar a cabo la comunicación vía puerto USB. Se exponen los puntos a considerar para la programación de los microcontroladores, su papel dentro del desarrollo y para finalizar su asociación a los elementos intangibles del sistema, es decir, los programas encargados de relacionar la información con la computadora.

Cabe mencionar que se puntualizará cada aspecto relevante de los puntos que se toquen a continuación, además de contar con diagramas que permitan una mejor comprensión.

4.2 Propuestas

Se ha enfocado el trabajo a la conexión por medio del puerto USB y al uso de un camino sin fin como control de desplazamiento de ambientes virtuales. Con esto se consigue una percepción completamente distinta de los controles de la caminadora, tomando movimientos más naturales para girar y avanzar, así como la selección de elementos.

Para conseguir aumentar la inmersión y mejorar la navegación, se plantean las siguientes propuestas:

- Reducir el índice de distracción del usuario a la pantalla. Cada vez que el usuario pierde de vista la proyección, regresa al mundo real y se concientiza de que es una ilusión. Esto ocurre muchas veces si se tienen controles que deben buscarse o localizarse para poder usarlos. Por eso mismo, existen algunos puntos susceptibles a mejorar. Esto ocurre en los controles de las consolas de videojuegos, donde los botones están bien ubicados por el jugador y no necesita mirar el mando después de cierto tiempo de experiencia.
- Mejorar la velocidad de comunicación. Si se llega a una relación de tiempo real, entonces el usuario puede sentir que sus acciones realmente están reflejándose frente a sus ojos. Además, cuando la respuesta sucede después de la maniobra ejecutada, por más pequeño que sea este retardo, el usuario llega a incomodarse y no disfruta de la aplicación.

4.3 Aportaciones

La conexión del dispositivo por el puerto USB es una de las contribuciones al trabajo de la cabina de inmersión. Con esto:

- Se aumenta la velocidad de transmisión de datos
- Se releva al puerto PS/2 de esta tarea, liberándolo para funciones aparte.

Por otra parte, también existen mejoras planteadas en cuanto a los controles de giro y desplazamiento. Como producto se conseguirá:

- Diseñar un control más cómodo y natural al usuario de la caminadora (giro y selección) cuya curva de aprendizaje sea elevada.
- Identificar la dirección del movimiento (delante y atrás), gracias a la adaptación de sensores.

4.4 Desarrollo

4.4.1 Ambiente virtual

Como parte medular de un simulador de realidad virtual se debe de tener un ambiente virtual. Hablando de ambientes dentro del computador en específico, en este tipo de entornos participan ciertos factores como son modelos artificiales, animaciones, efectos de luz y sombra y fenómenos

físicos que el diseñador desee. Debe tenerse en cuenta el propósito de dicho mundo, el cual es proporcionar al usuario un entorno adecuado para desplazarse y poder probar los controles desarrollados.

Para estos fines, se ha desarrollado un entorno boscoso en el cuál se integran los elementos recreados de manera sencilla pero satisfactoria a nivel de credibilidad del usuario. Con la misión de demostrar el funcionamiento del sistema de navegación, el ambiente artificial cuenta con una gran extensión para que no salga el usuario del entorno. Además, con los efectos de neblina no es posible apreciar las limitaciones y el mundo parece aún más grande.

Creación de modelos 3D

Los elementos que forma parte del ambiente simulado son por lo general tridimensionales (a menos que el propósito del entorno requiera de figuras planas). Suelen utilizarse varias formas simples para crear otras más complejas y así no tener que hacer parte por parte una malla que represente la estructura del objeto. Los modelos (figura 4.1) deben contar con los siguientes factores básicos:

- Un bajo nivel de detalle. Si los modelos contienen un alto grado de detalle, la carga computacional será mayor. Debido a que se busca mejorar el rendimiento, no debe de exagerarse con los polígonos que conforman a un cuerpo tridimensional. En el siguiente punto, se ve cómo contrarrestar la simplicidad de la forma con el gran detalle que proporciona una textura.
- Mapeado de textura. Para que los modelos puedan pasar por elementos del mundo real, se escoge una imagen que represente adecuadamente la superficie del objeto. Con esta imagen se “envolverá” al modelo para que, sin necesitar muchos triángulos en su estructura, se consiga gran realismo.

Para el entorno boscoso, se han creado elementos de vegetación (arbustos, hierbas y árboles como robles, hayas, etc.), además de rocas y una superficie con algunas variaciones de altura.

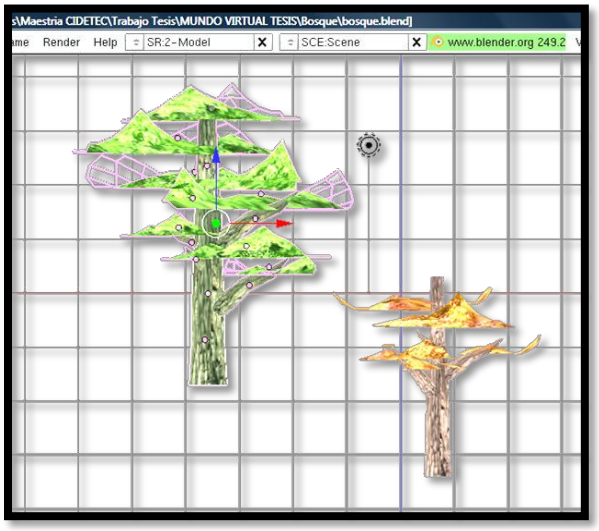

Figura 4.1 Modelos de árboles

Formando el mundo virtual

Para completar el cuadro, se lleva a cabo la integración de los elementos diseñados junto con algunos efectos visuales para agregar realismo (figura 4.2). Todo componente del ambiente generado debe contar, con respecto a los demás componentes, con una buena relación de proporciones, ya que es un aspecto esencial a la hora de crear mundos realistas y poder facilitar la inmersión. El tipo de iluminación también es fundamental para crear efectos que afecten estados de ánimo o sentimientos del usuario. En este caso, se recurre al efecto de neblina para crear primordialmente dos efectos: la sensación de encontrarse dentro de un ambiente de extensión muy grande o infinita, y proporcionar la ilusión de un clima frio.

En este punto, aún no se han programado cuestiones referentes a controles de desplazamiento, comunicación entre computadoras o tareas encargadas de la visualización del entorno de acuerdo al movimiento del navegante.

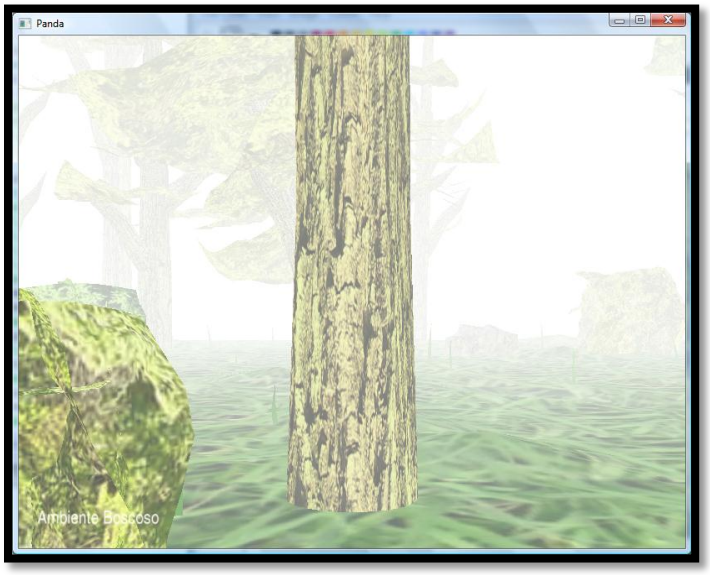

Figura 4.2 Ambiente conformado por la unión de modelos y efectos

TCP y UDP

Los protocolos de comunicación del sistema, como ya se ha hecho mención, son TCP y UDP. Mientras que uno se enfoca en establecer una conexión bidireccional, el otro solo se encarga del paso de mensajes sin requerir respuesta alguna. Por esta razón, la comunicación UDP resulta ser más veloz y la mejor opción para estos fines, ya que no resulta tan relevante revisar si los datos fueron transmitidos sin errores en comparación con la rapidez que el sistema necesita.

Posición absoluta y relativa

En cuanto a la transmisión de datos, debe de considerarse la información que será enviada a los clientes del sistema. Por un lado se tiene la opción de mandar el punto en el que se localiza el usuario y la dirección de visualización, con lo cual los equipos colocan la cámara en la coordenada indicada y no importa en donde haya estado antes el usuario. La otra opción existente es el comenzar el programa en un punto inicial en común en todas las computadoras y sólo transmitir el movimiento y la dirección del mismo, así como el giro.

Transmisión de la orden

Para mostrar claramente el proceso de circulación de las órdenes, en el siguiente diagrama (figura 4.3) es posible observar qué camino llevan las instrucciones a partir del exterior y finalizando

en los equipos de la red. Por medio de la red, la orden del servidor es transmitida hacia todos los equipos (figura 4.4). Sin embargo, debe de designarse algún momento en el que se llevará a cabo dicha transmisión. Dado que hacer esto a cada momento alentaría todo el sistema, se ha optado por llevar a cabo la comunicación cada vez que exista un cambio de movimiento o rotación.

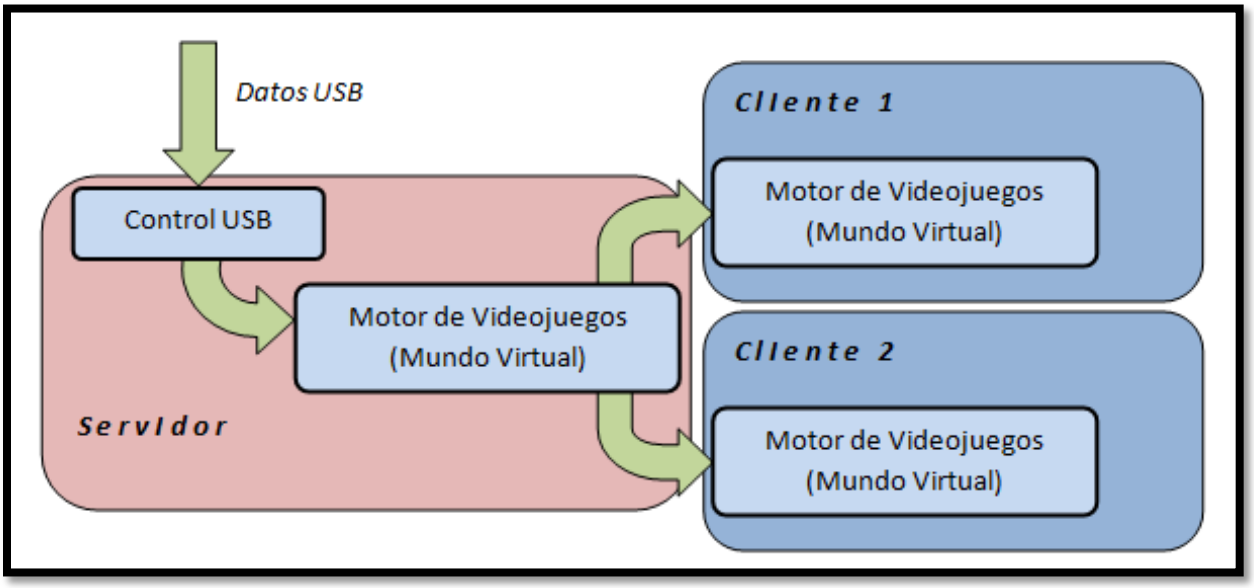

Figura 4.3 Diagrama de transmisión de datos

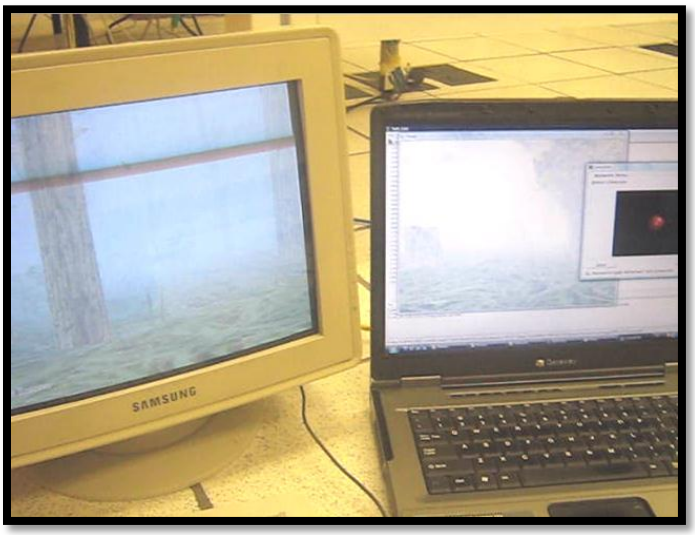

Figura 4.4 Comunicación entre equipos

El cliente da aviso al servidor de que está listo para recibir órdenes. Es entonces cuando, libremente, la computadora principal puede mandar los comandos a los otros equipos y esperar que

cumplan con las instrucciones. Dado que la instrucción consta sólo de una orden a la vez (es decir adelante, atrás, izquierda, derecha o selección), la operación es muy sencilla. Se facilita aún más si se considera que sólo se mandan órdenes de “seguir adelante” o “no girar”, por mencionar algunos ejemplos.

4.4.2 Componentes físicos

Diagramas de bloques y de flujo

Los elementos que se encargan de proporcionar las señales requeridas al sistema computacional son definidos por medio de diagramas de bloque, en donde se especifica cada sección de los módulos del control. Cuando se interconectan todos los módulos (figura 4.5), el funcionamiento es completo y el dispositivo es capaz de registrar las variaciones de posición y orientación en el ambiente simulado.

Gracias a esta representación gráfica del sistema, el trabajo es fácilmente comprendido independientemente de los componentes físicos con los que se ha contado, debido a que la lógica de funcionamiento no cambia de forma alguna. La visión particular de las secciones del dispositivo tiene las siguientes ventajas:

- Puede estudiarse a fondo por partes el sistema y mejorar la comprensión
- Es más sencillo proponer mejoras a futuro de una sección en especial, que de la totalidad del sistema

Más adelante, al revisar la unión de los módulos creados, es posible verificar la relación de los datos transmitidos desde el segmento de desplazamiento (figura 4.6), el módulo de rotación y selección (figura 4.8), el segmento de generación de órdenes (figura 4.10) y la sección de comunicación con el puerto USB (figura 4.12), hacia los demás bloques y el sentido de la comunicación misma.

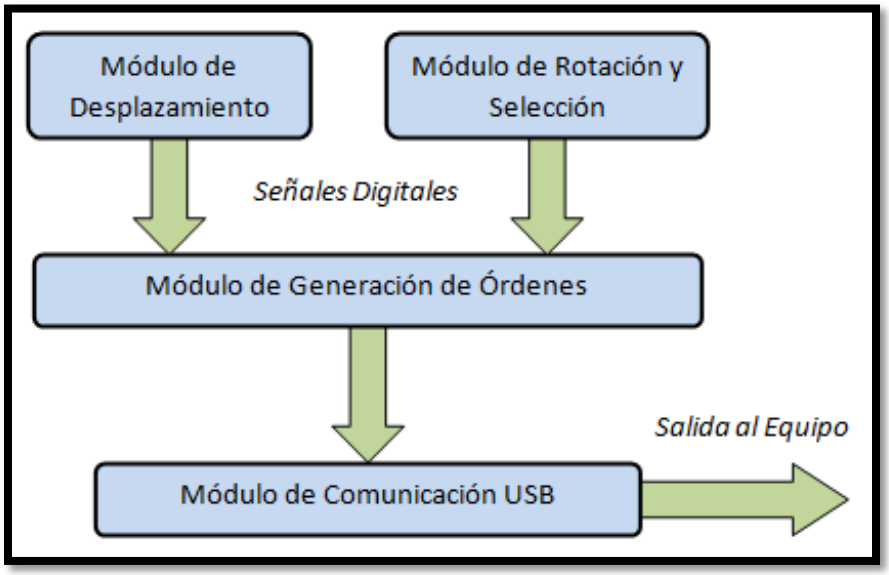

Figura 4.5 Diagrama de bloques general

Módulo de desplazamiento

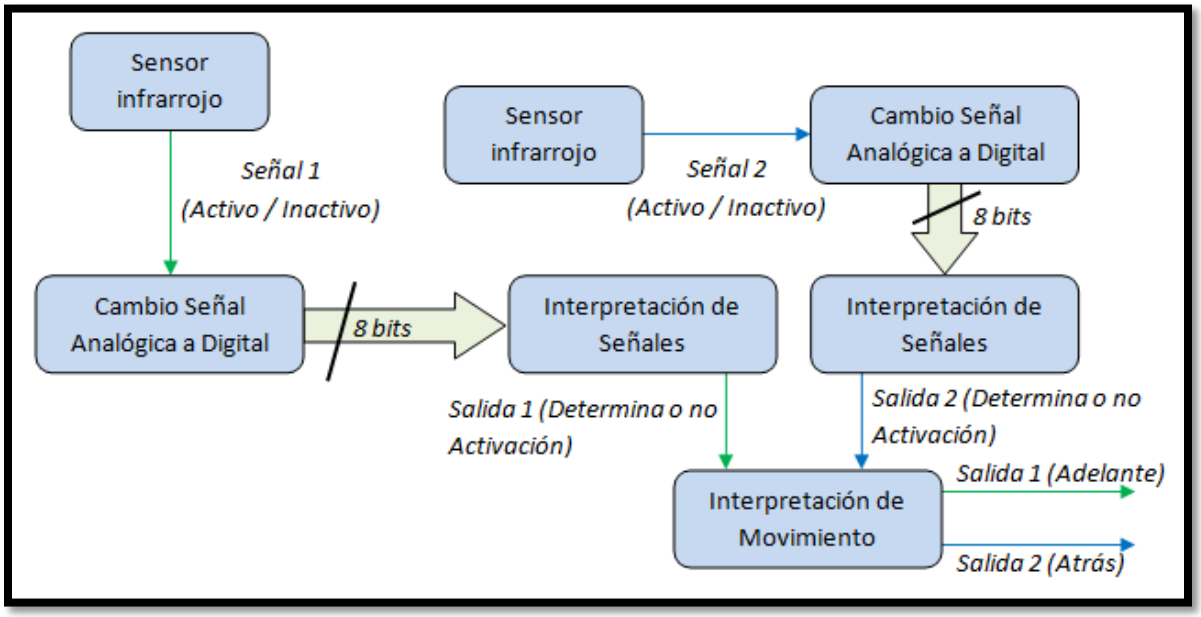

Figura 4.6 Diagrama de bloques del módulo de desplazamiento

La sección de desplazamiento está encargada de analizar las señales de los sensores (infrarrojos, en este caso), los cuales revisan las variables que se han considerado para el movimiento. Para la tesis actual se determinó la revisión del mecanismo de la caminadora, mismo que se estudiará en páginas posteriores. Después de obtener la señal de respuesta de los sensores,

se requiere un cambio a señal digital para ser considerada para los cálculos siguientes. A continuación, los datos resultantes entran en un pre-procesamiento para determinar sí, en consideración con ciertos valores mínimos de activación dados, se tomará su señal como un “1” o un “0”. Inmediatamente después se pasa al proceso de interpretación para distinguir la dirección hacia la cual se desplaza el usuario. Esto resulta en dos señales de salida (una para cada sentido) que habilitan o deshabilitan opciones del módulo de generación de órdenes. La Interpretación del movimiento se explica más claramente en el diagrama siguiente (figura 4.7):

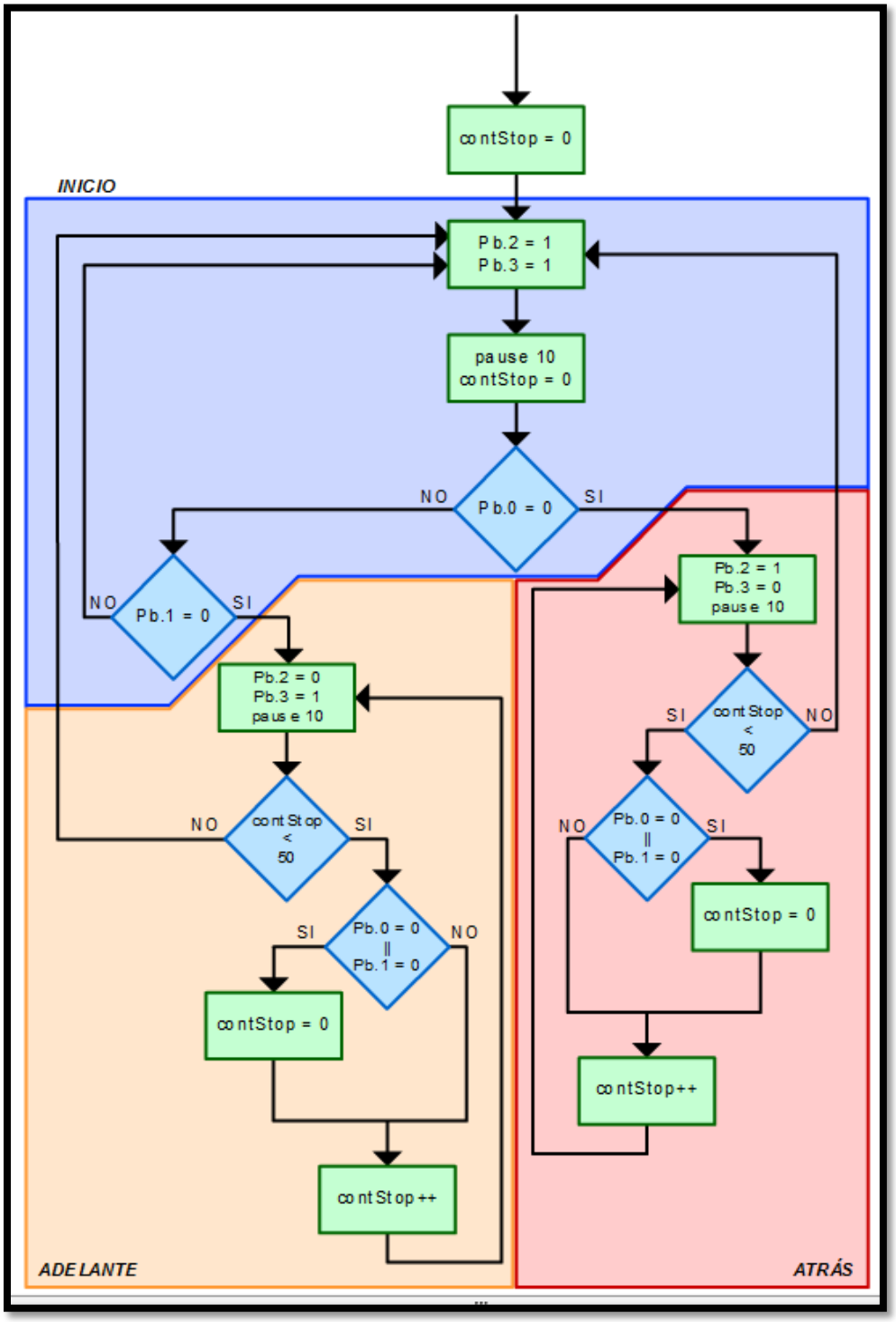

Figura 4.7 Diagrama de flujo del submódulo de interpretación de movimiento

En el anexo se dispone el código fuente para complementar la revisión del proceso de interpretación, mostrando información respecto a las entradas y a las salidas del submódulo e inicialización de variables de configuración.

Módulo de rotación y selección

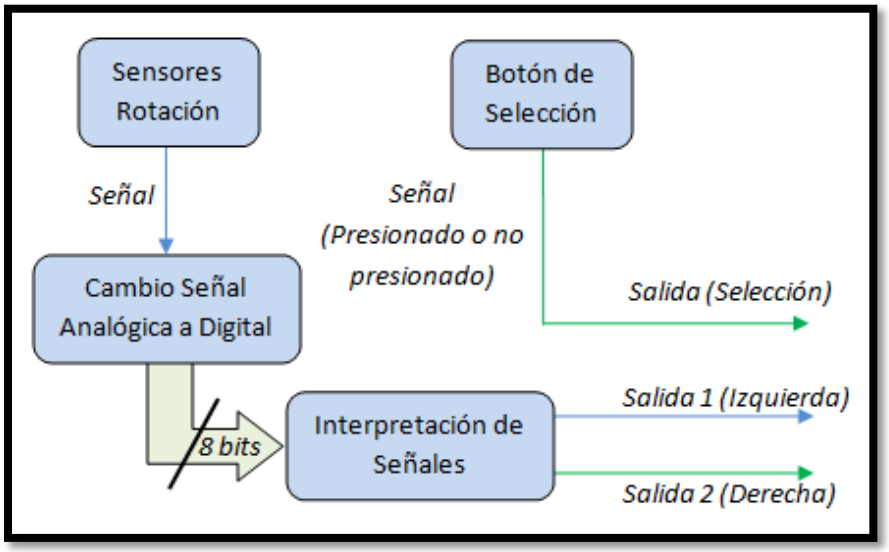

Figura 4.8 Diagrama de bloques del módulo de rotación y selección

Este módulo básicamente desempeña las mismas funciones que el segmento visto en el punto anterior. La diferencia que salta a la vista en primer lugar, es que sólo ocupa una señal del sensor para distinguir la orientación del usuario. Esto se debe a que en el trabajo se ha optado por utilizar un acelerómetro. El acelerómetro se encarga de generar un voltaje alto o uno bajo dependiendo de la orientación del dispositivo. Cuando el sensor refleja en el voltaje de salida su respuesta al estímulo físico (cambio de orientación), la conversión analógica – digital es el segundo paso. Ya que se cuenta con un valor en términos digitales, se recurre a la interpretación del microcontrolador (figura 4.9). Al finalizar el proceso, se mandan señales para el giro a la derecha y a la izquierda. En cuanto a la selección, debido a que se está utilizando un *push-button*, la señal pasa sin cambios a reflejar un 1 o un 0 lógicos.

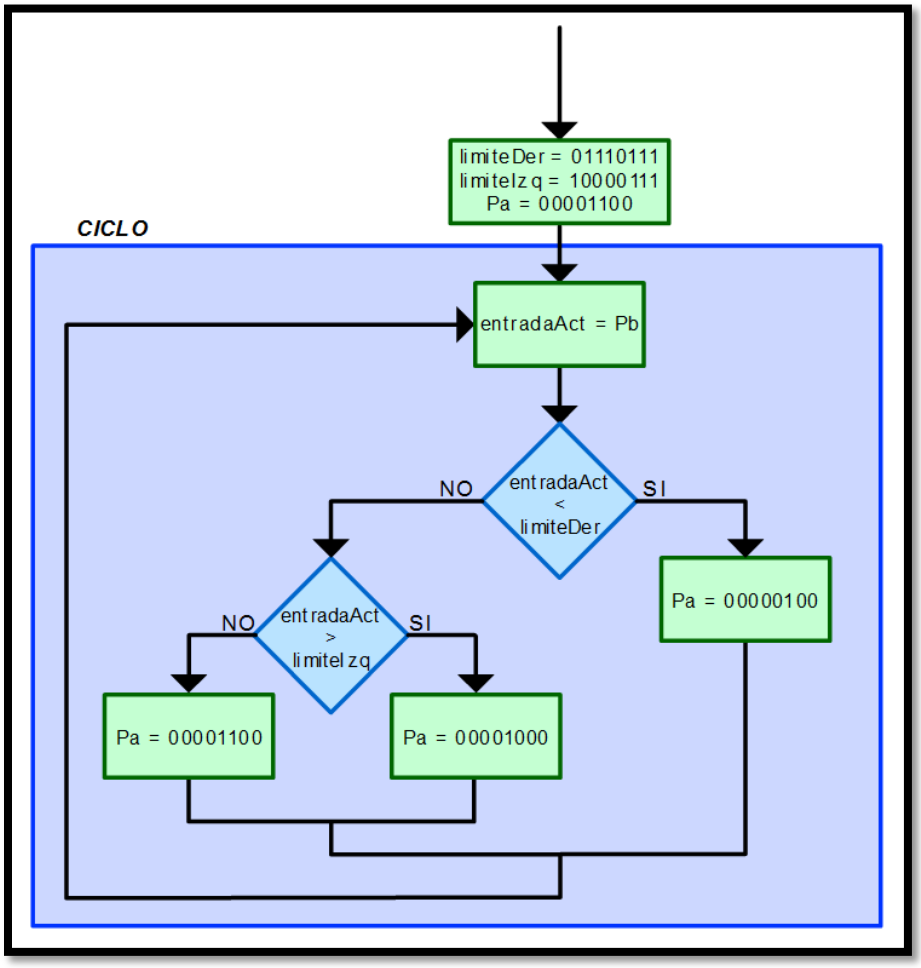

Figura 4.9 Diagrama de flujo del submódulo de interpretación de señales

Cabe mencionar que el uso de valores binarios en esta sección, no es coincidencia. Ya que se utilizan sólo un par de pines para llevar a cabo el cambio de orientación, el entendimiento del código, y del diagrama por ende, es más sencillo que si se manejaran valores decimales. Esto no repercute al iniciar su funcionamiento el sistema y primordialmente se recurre a esta notación en base dos para una visualización clara de lo que ocurre internamente. Los valores asignados a *limiteDer* y *limiteIzq* son determinados después de probar varios rangos y su respuesta en el mundo artificial.

Módulo de generación de órdenes

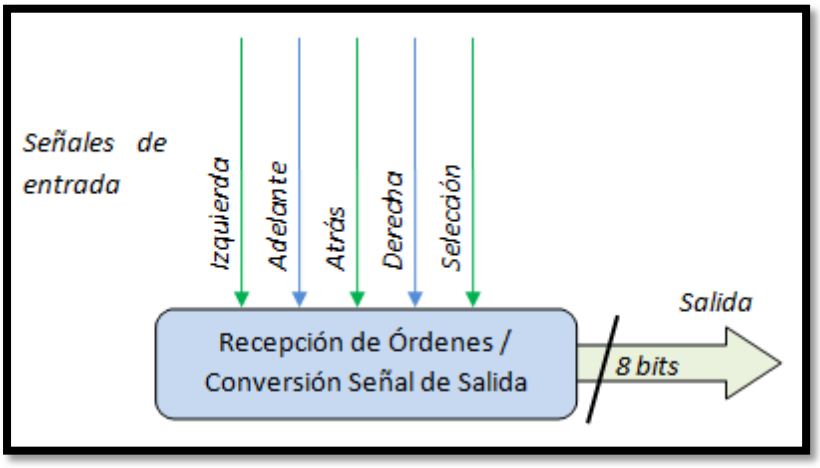

Figura 4.10 Diagrama de bloques del módulo de generación de órdenes

En el segmento de generación de órdenes se recogen las señales provenientes de los bloques anteriores, donde ya se han transformado en valores binarios, y los concentra para formar una instrucción que representa a estas señales (figura 4.11). La salida resultante será enviada por el puerto USB al equipo de cómputo gracias al siguiente módulo (comunicación USB).

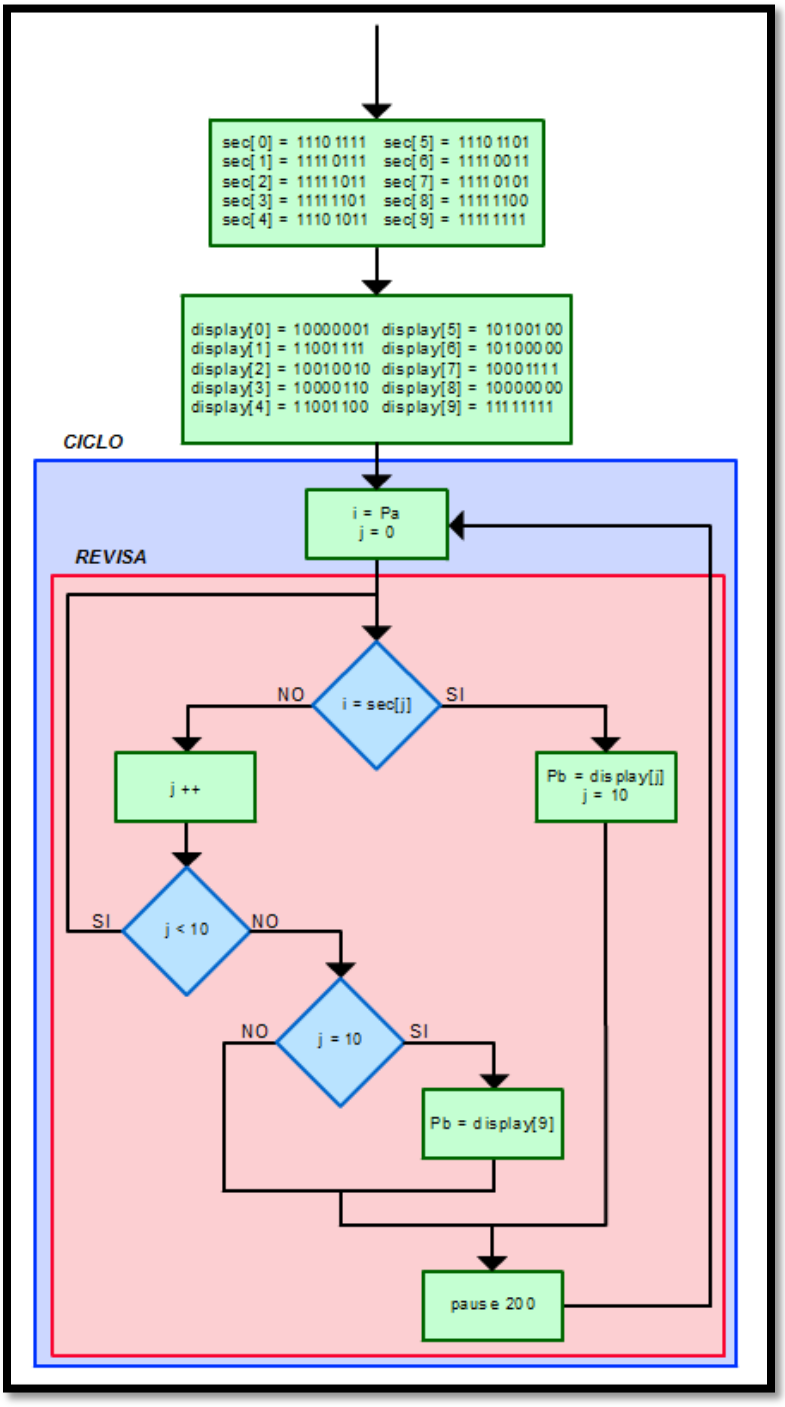

Figura 4.11 Diagrama de flujo del módulo de recepción de órdenes y conversión de la señal

El propósito de la sección es el de reducir las cinco señales posibles y sus combinaciones en simples valores numéricos que fueran sencillos de interpretar ya por la computadora y la programación enfocada a ello. Puede apreciarse en los diagramas que el programa es cíclico, es decir, no termina nunca debido a que el sistema utiliza el módulo durante todo su funcionamiento.

Cabe mencionar que los valores mostrados en el *display* son los mismos que las órdenes (traducidos a base diez más adelante al recibirse la información en el equipo de cómputo).

Módulo de comunicación USB

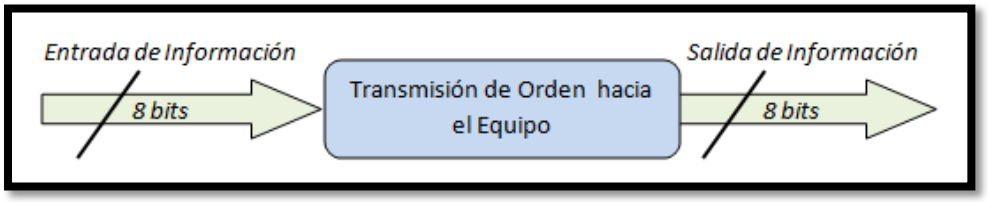

Figura 4.12 Diagrama de bloques del módulo de comunicación USB

Este módulo simplemente se encarga de la transmisión de los datos a su destino final: el equipo computacional. Por medio del proceso adecuado para la transferencia de la información, este módulo podría adaptarse a la comunicación por cualquier otro puerto ya que se muestra una conjunción del sistema físico a la computadora. En resumen, podría decirse que el único bloque de este apartado simboliza la traducción de información a la debida definición de los datos para que la comunicación se lleve a cabo.

Diagramas esquemáticos

Para poder diseñar la circuitería deben de estar definidos en un principio los componentes y su interconexión por medio de diagramas esquemáticos. A continuación se pasa directo a la implementación física del diseño y la explicación pertinente de cada circuito de acuerdo al módulo que represente.

El circuito consta de varias unidades que al final deben de integrarse (figura 4.13). Estas son:

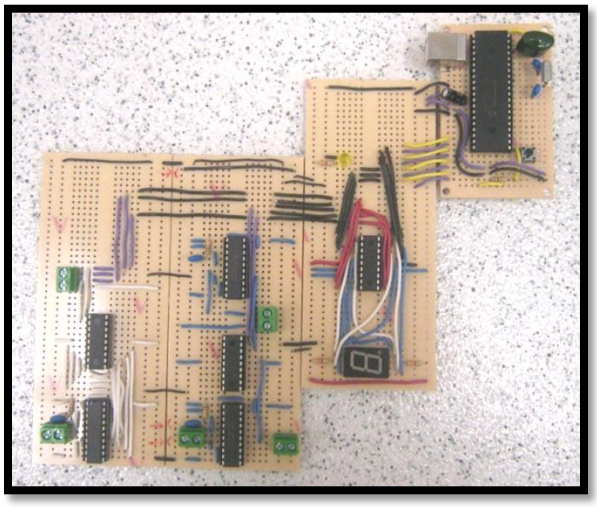

Figura 4.13 Circuito final con módulos integrados

- Módulo de desplazamiento. Este módulo recibe la información de los sensores montados en la caminadora y que se encargan de registrar el desplazamiento y dirección del mismo. Esta información más tarde es transmitida a la placa de generación de órdenes. Está conformado por dos submódulos idénticos (figura 4.14 y figura 4.15), los cuáles se encargan de convertir la señal de los sensores a valores binarios que pueda leer fácilmente el microcontrolador. A continuación se pueden apreciar ambos diagramas:

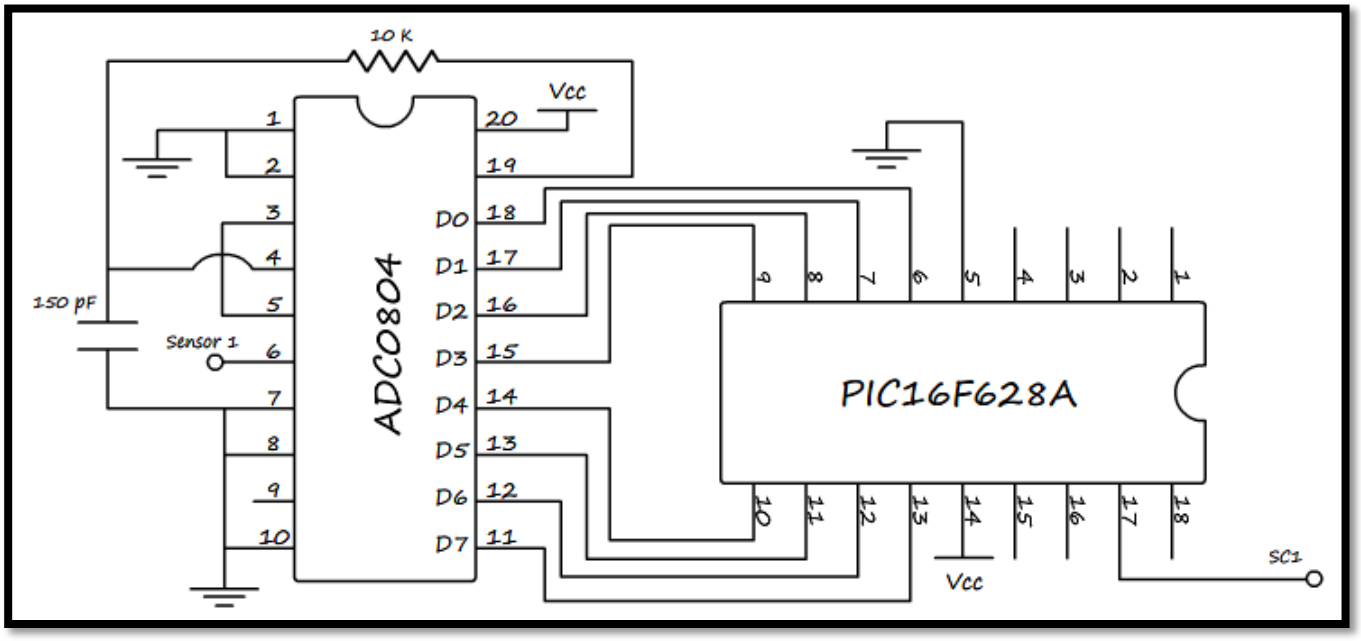

Figura 4.14 Submódulo de conversión del primer sensor de desplazamiento

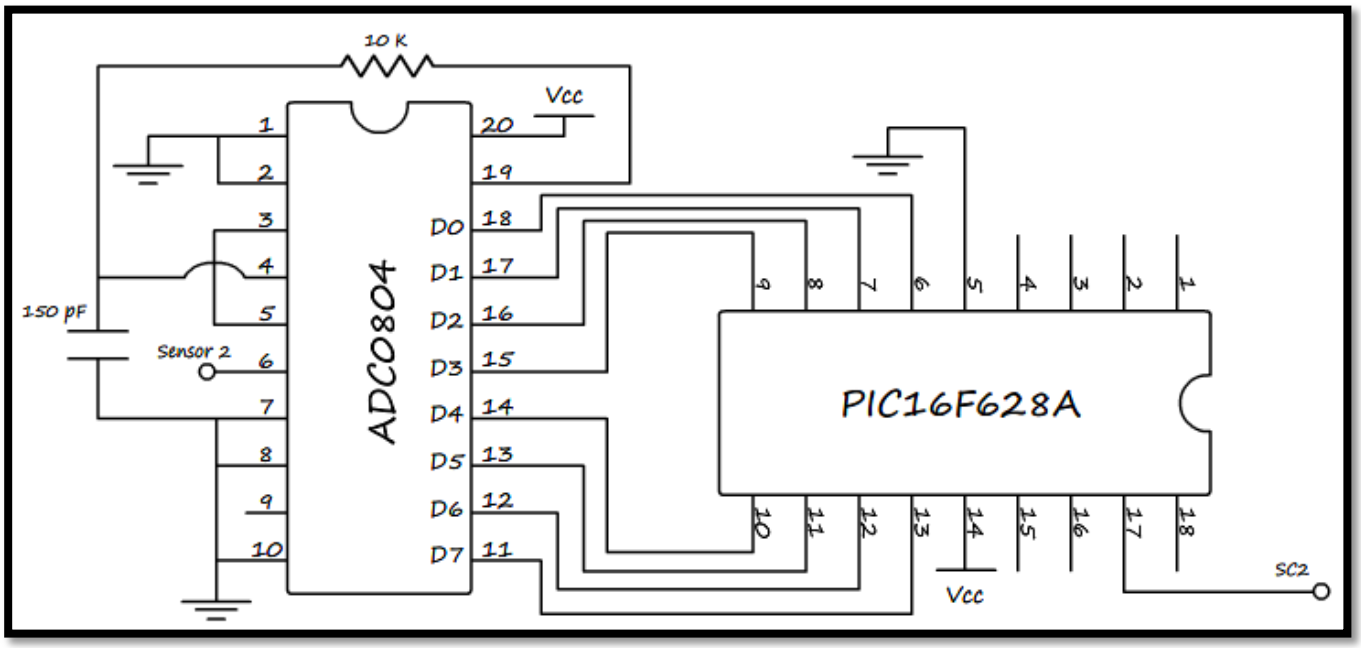

Figura 4.15 Submódulo de conversión del segundo sensor de desplazamiento

Las salidas de los submódulos van hacia la siguiente etapa, en la que la activación y/o desactivación de los sensores son interpretadas para determinar si el usuario se desplaza (hacia adelante o hacia atrás) o se mantiene inmóvil (figura 4.16). Este resultado es enviado hacia el módulo de generación de órdenes.

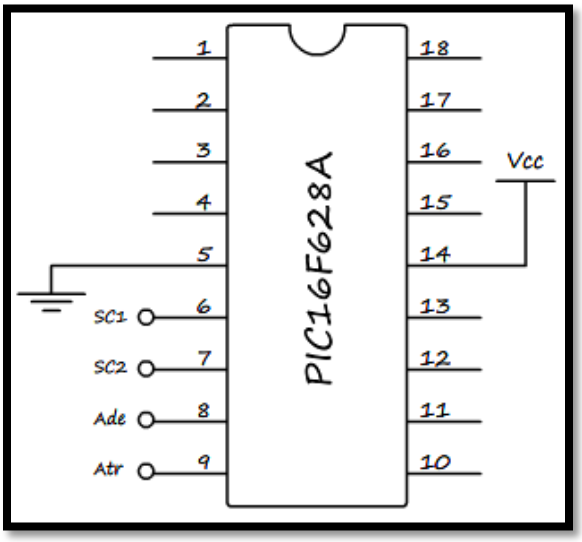

Figura 4.16 Etapa de interpretación de sub módulos

Al integrarse físicamente los componentes del módulo (figura 4.17) se obtiene una tablilla con todos los recursos necesarios para manejar las señales de la caminadora y transmitir a la siguiente sección si el usuario se movió o no, y si es que lo hizo hacia qué dirección ocurrió.

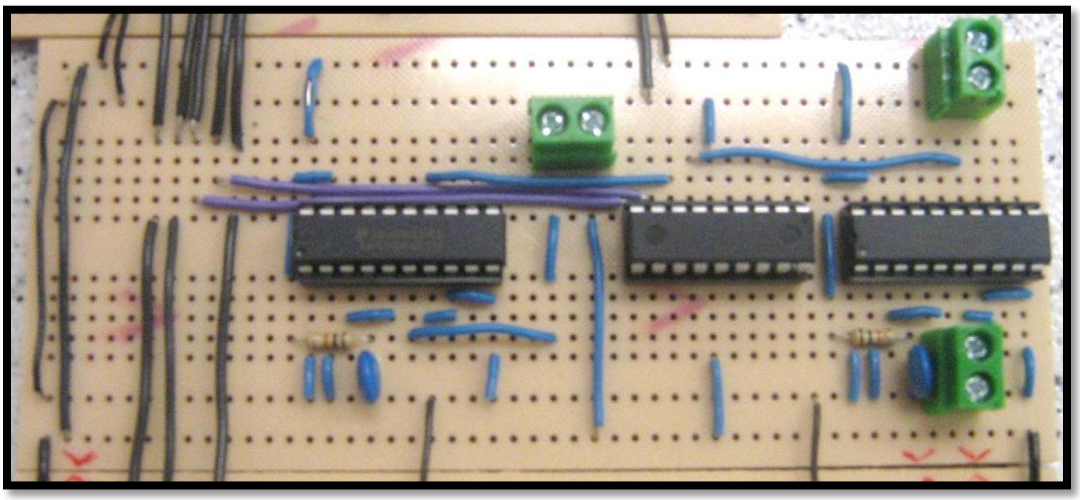

Figura 4.17 Placa de sensores de desplazamiento

- Módulo de sensores de rotación y selección. Aquí son traducidas las respuestas de los sensores de giro y de la opción de selección. De la misma forma que ocurre con la placa de desplazamiento, la información circula hacia la placa encargada de unir todas las órdenes y crear un comando para transmitir a la computadora.

En un principio, también debe considerarse una etapa de transformación de las señales con la ayuda de un microcontrolador (figura 4.18). Con esto, el trabajo se limita a valores binarios y es más entendible el código programado. En el mismo elemento programable, se obtienen las salidas correspondientes a los giros posibles del usuario (izquierda y derecha) que permiten identificar el cambio de dirección.

En cuanto a la selección, se recurre a dos botones conectados en paralelo. Con esto existe la posibilidad de presionar cualquier elemento, e igualmente, se identificará como selección en el sistema. Para evitar los rebotes que pudiera tener el control, se programó un anti rebotes en un microcontrolador que forma parte de otro módulo. Éste consta de un retardo del orden de milisegundos, con lo que la señal se estabiliza.

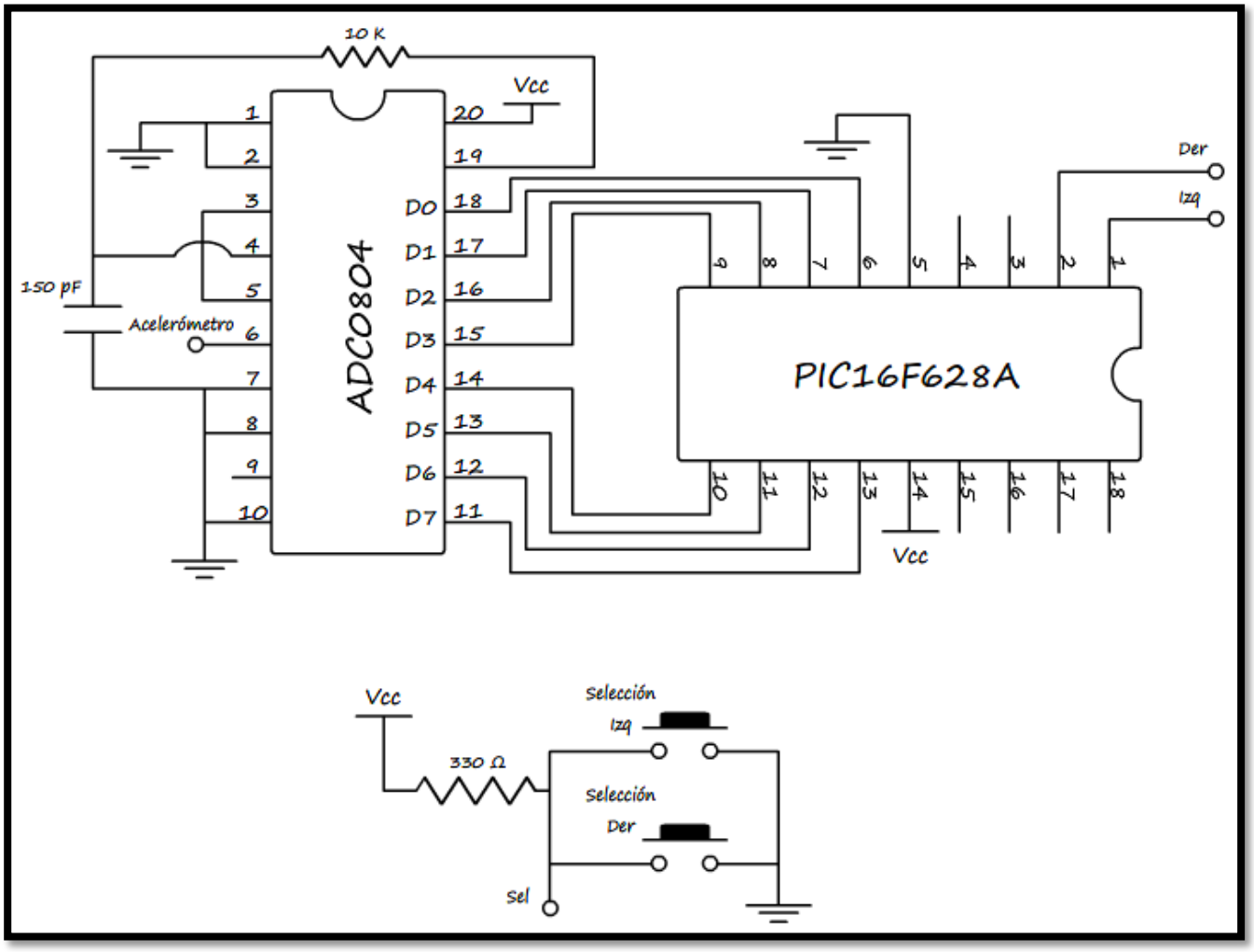

Figura 4.18 Módulo de conversión directa del acelerómetro y de la selección

En la placa (figura 4.19) no es posible apreciar en conjunto la representación de los diagramas esquemáticos, ya que parte del módulo (selección) se encuentra dentro del control de rotación de la caminadora. Sin embargo, es posible distinguir los elementos encargados del control de las señales del acelerómetro.

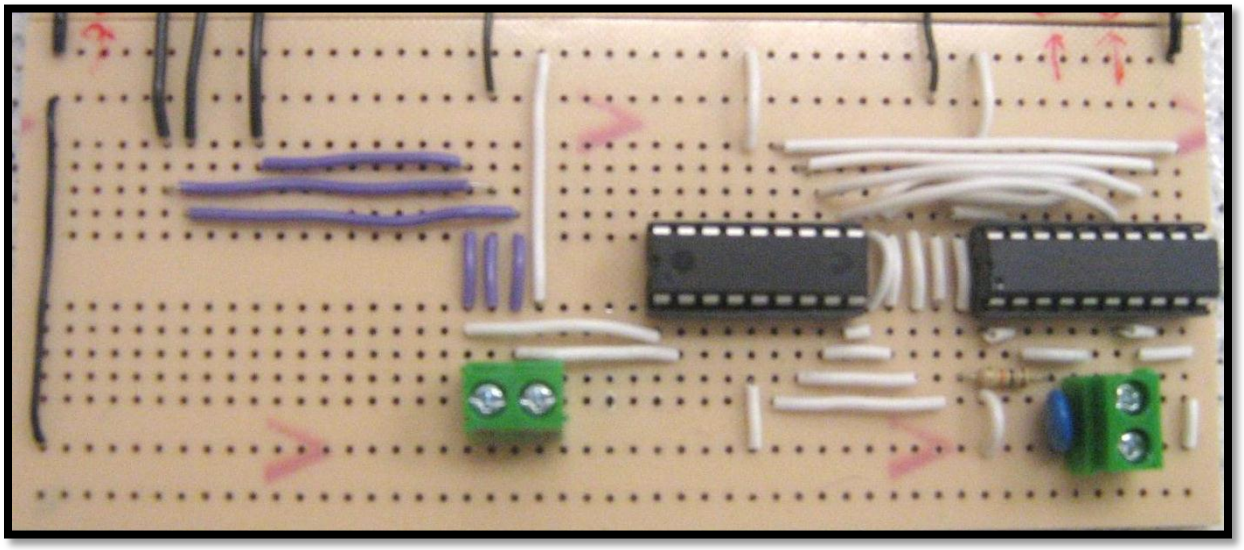

Figura 4.19 Placa de sensores de rotación y selección

- Módulo de generación de órdenes. Toma los datos provenientes de las placas de los sensores y los traduce de tal manera que existan nueve comandos posibles, mismos que son reflejados en un visualizador de siete segmentos[5] montado en este módulo y que pueden ser transferidos hacia la placa de comunicación USB.

Como resultado de los módulos anteriores (desplazamiento y rotación - selección) se obtienen cinco salidas que toman el papel de entrada en el presente módulo. A manera de reducir las transmisiones independientes al módulo de comunicación USB, en este espacio se generan instrucciones que representan un conjunto de acciones conjuntadas representadas por un byte (ocho valores binarios).

Además, se cuenta con dos elementos ajenos al desempeño central del sistema: un LED de aviso, que enciende si el sistema se encuentra conectado a la computadora; y un elemento de despliegue que permite conocer la información que se transmite para verificar su funcionamiento. Para que este último elemento sea de fácil lectura, los valores de las instrucciones son representaciones de números para el *display* de siete segmentos (figura 4.20).

[5] El visualizador de siete segmentos, o "display", es un elemento electrónico que cuenta con ocho LEDs incorporados en su interior y sirve para representar números. Necesita un elemento que traduzca los números que se busca representar a ceros y unos que serán enviados a cada LED.

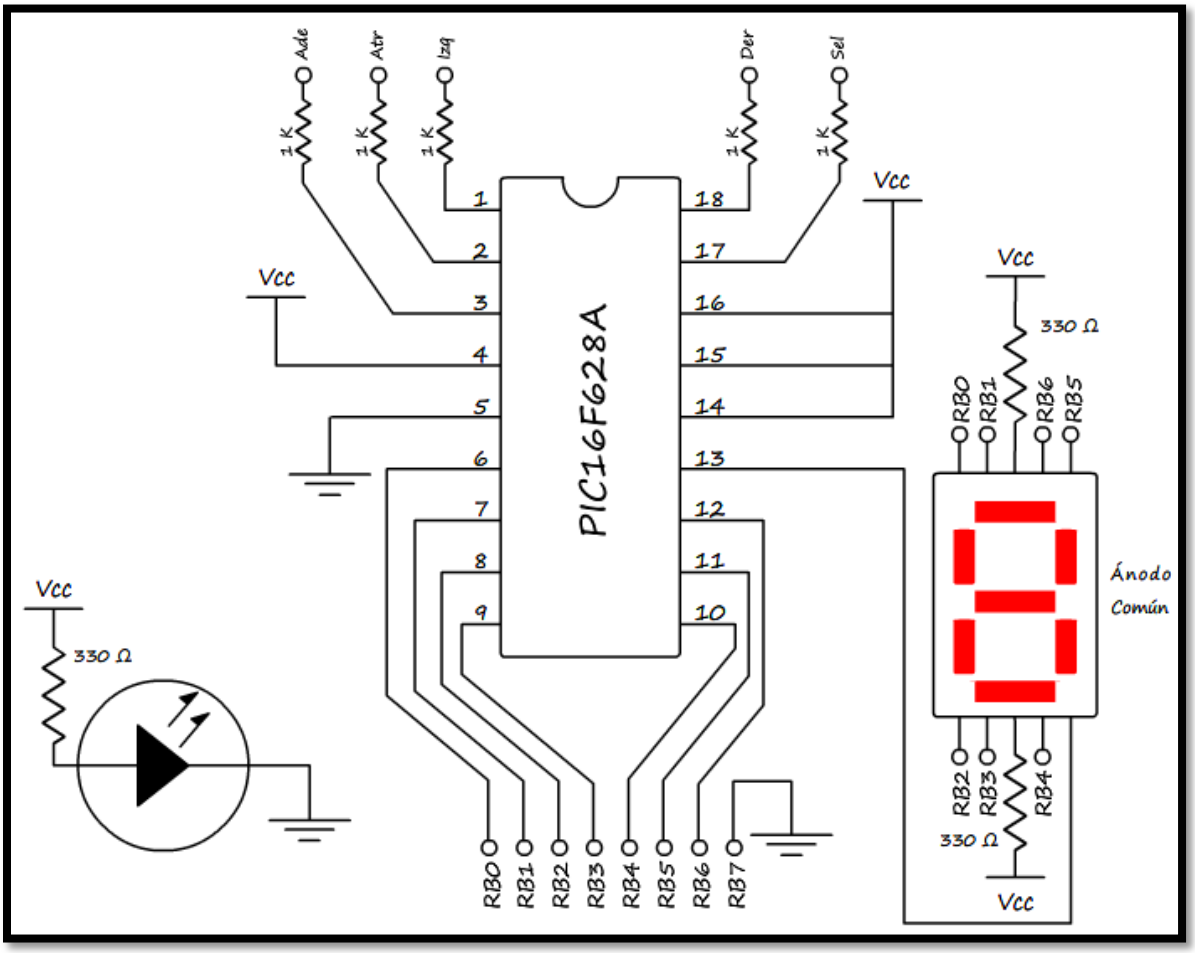

Figura 4.20 Módulo de generación de órdenes

Al ser montado en una tablilla (figura 4.21), el módulo pierde cierta comprensión debido a las uniones formadas. A pesar de ello, es posible distinguir cada sección antes mencionada.

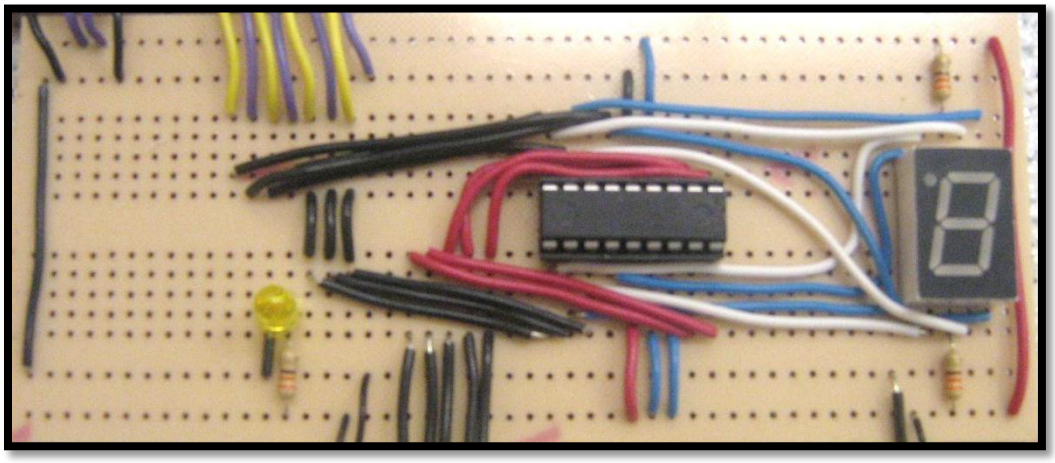

Figura 4.21 Placa de generación de órdenes

- Módulo de comunicación USB. Encargada de la transmisión de datos de la placa de generación de órdenes hacia la computadora que desempeña el papel de servidor. Recauda los datos y los envía hacia el equipo de cómputo para su interpretación a nivel de software. Esto se lleva a cabo gracias a las facilidades que ofrece el microcontrolador seleccionado (figura 4.22), el cual está dirigido a tomar este rol.

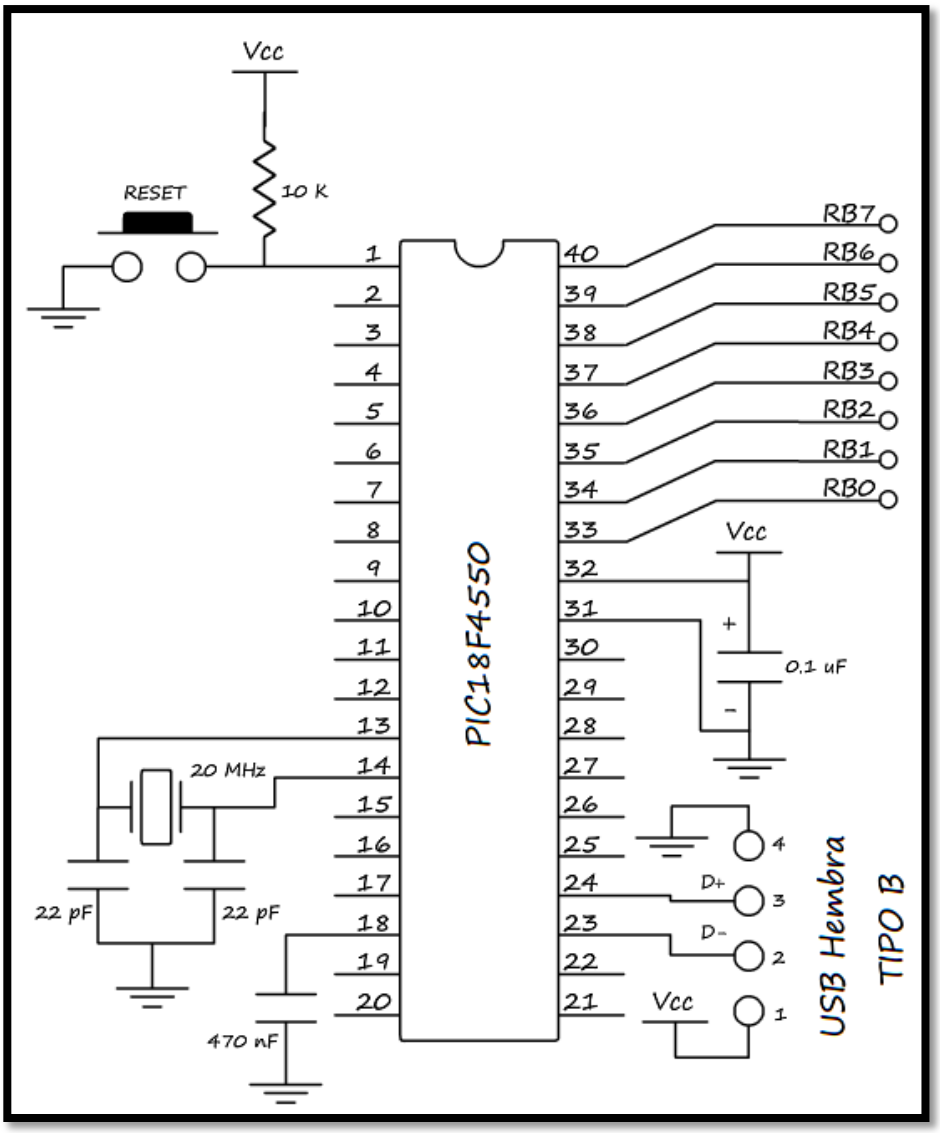

Figura 4.22 Módulo de comunicación USB

Su construcción está basada en la conexión básica de transferencia de datos del PIC18F4550 ya que no se requiere nada más profundo en cuanto a este punto se refiere. Además, cabe mencionar que el módulo de comunicación también se encarga de proporcionar la energía a todos los módulos que conforman al sistema. Gracias a la terminal que otorga 5 V provenientes del equipo, no es requerido ningún elemento externo que suministre el voltaje. Su representación final puede apreciarse en la figura 4.23.

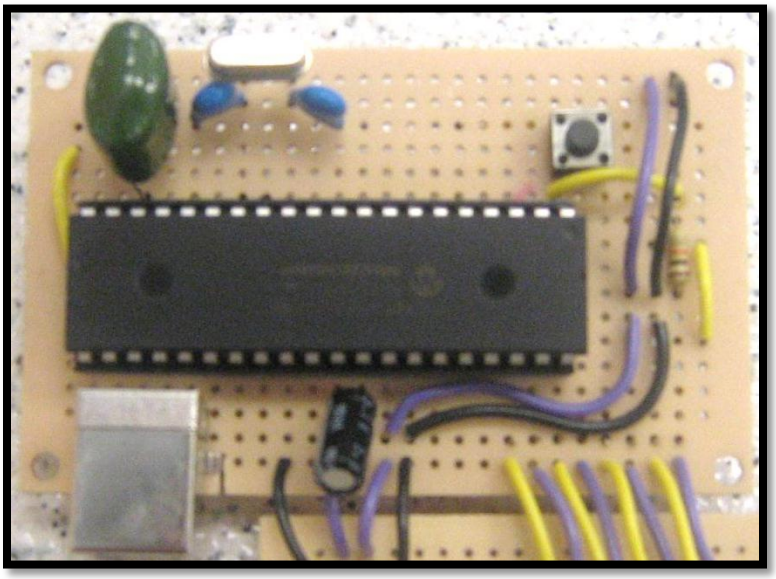

Figura 4.23 Placa de comunicación USB

Sensores

Por medio de los sensores se transforman ciertos impulsos o cambios en el entorno. Estos se reflejan en forma de cambios que se experimentan en el ambiente virtual. Para la aplicación de la caminadora y el mundo artificial son necesarios sensores para el desplazamiento, la rotación y la selección.

Sensores de desplazamiento

El movimiento natural de una persona en la caminadora es asociado a la cámara del mundo virtual. Esto es consecuencia de que los sensores lean el comportamiento de la banda sin fin y lo interpreten como movimiento en la dirección correspondiente. Sin embargo no se detecta propiamente a la banda, sino al comportamiento de la rueda del rodillo (figura 4.24) encargado de manejar dicha banda, debido a que es más sencillo de tratar y pueden incorporarse los sensores en él.

Figura 4.24 Rueda del rodillo de la caminadora

Para el manejo de este desplazamiento, se opta por un par de sensores infrarrojos. Esto con la finalidad de poder determinar la dirección de giro de la rueda. Además, en las pruebas se ha demostrado su correcto funcionamiento y que cuenta con ventajas claras como ser inmune a la luz ambiental. Se montan en una estructura (figura 4.25) creada especialmente para este propósito, que abarca ambos lados de la rueda. Mientras que por un lado se emite la luz infrarroja (una por cada sensor), en el otro extremo es recibida (figura 4.26).

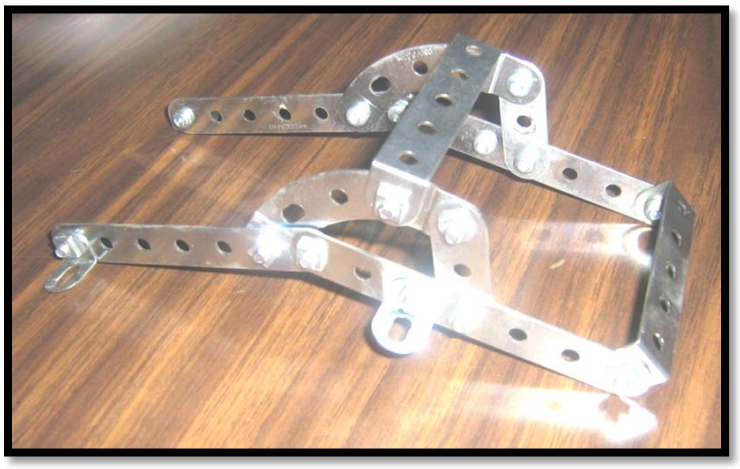

Figura 4.25 Estructura para montar los sensores infrarrojos

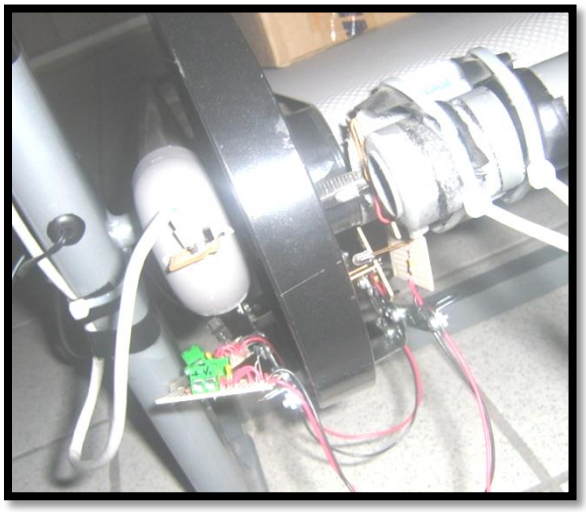

Figura 4.26 Elementos montados en la caminadora (sensores de desplazamiento)

Sensores de rotación

Para poder girar dentro del ambiente simulado se recurre al funcionamiento de un acelerómetro (figura 4.27). Éste toma en cuenta el efecto de la gravedad para proporcionar un voltaje equivalente, que después puede ser interpretado con un convertidor analógico – digital y un microcontrolador (placa de sensores de rotación y selección).

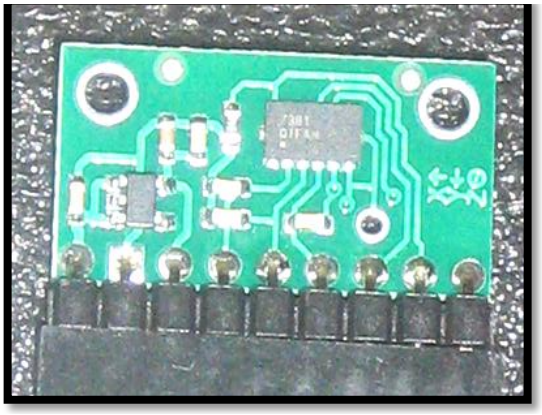

Figura 4.27 Acelerómetro encargado de la rotación

Selección

La selección es la parte más sencilla del sistema, ya que consta de dos botones colocados a los costados del control de giro (figura 4.28). Esto con el fin de que el usuario no tenga que buscar el elemento y presionarlo, sino que solo debe de entender el área que cubre cada botón y así seleccionar algún objeto de forma más natural.

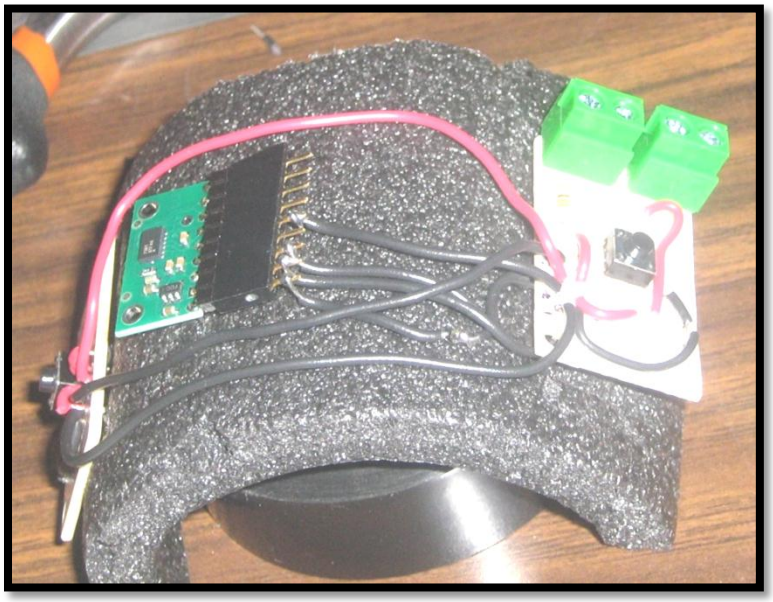

Figura 4.28 Botones situados a los extremos del control de giro

Al final, se cuenta con los dos controles del sistema finalizados (figura 4.29). El módulo de desplazamiento quedará ubicado en la parte inferior de la caminadora, invisible al usuario del sistema. Por otra parte, el control de giro y selección puede apreciarse en el brazo del aparato de ejercicio, por lo que el usuario puede disponer de él cuando lo requiera. Sin embargo, cabe mencionar que a pesar de que el control es visible, los botones o sensores se encuentran escondidos.

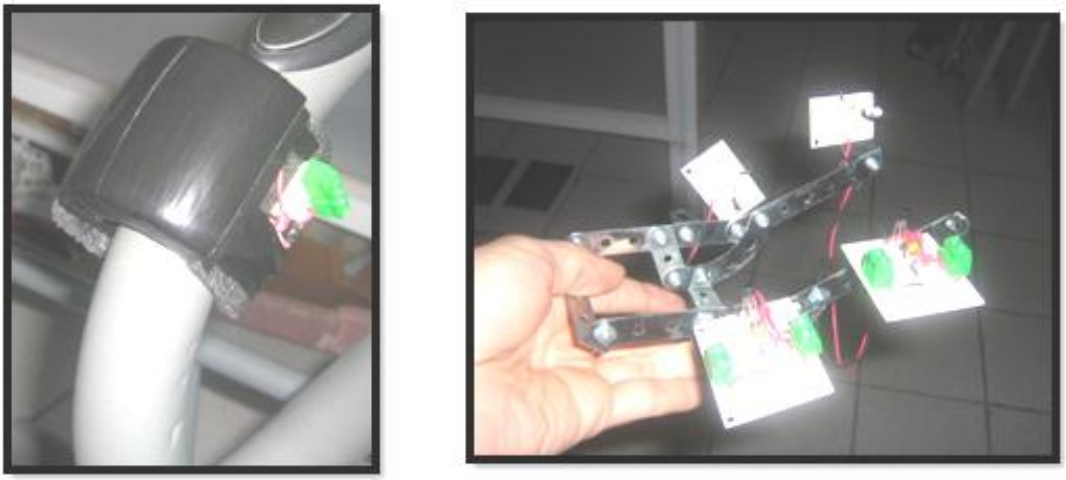

Figura 4.29 Módulos de control finalizados. Puede apreciarse el control de giro (izquierda) y el control de desplazamiento (derecha)

4.4.3 Comunicación USB

Para comunicar datos por este puerto, no sólo se necesita una parte encargada de esta labor, sino también proporcionar la información adecuada. Por la primer parte se encarga el PIC18F4550 y por la segunda harán la labor varios PIC16F628A, encargándose de transformar los datos recibidos en órdenes predefinidas que facilitan la interpretación en el programa. Primordialmente, se ha elegido el PIC16F628A por la experiencia adquirida y por el conocimiento en cuanto a su programación. En cuanto al microcontrolador PIC18F4550, es un elemento que forma parte de la misma familia (y por lo tanto comparte los mismos principios) y el aprendizaje se limitó a la comprensión de la transmisión por USB. Con esto, el enfoque pasa a la lógica de funcionamiento en lugar del entendimiento de otras tecnologías.

Programación del PIC16F628A

Tiene un papel fundamental de intermediario entre los sensores y el PIC encargado de la transmisión por USB, ya que recibe el voltaje suministrado por cada sensor, o de algún ADC8040[6], para luego tener como salida una activación o desactivación de algún elemento de control. El mismo rol de mediador podría desempeñarlo cualquier elemento programable con los pines de entrada/salida requeridos en cada módulo.

Otro microcontrolador de este tipo, que forma parte del circuito, es aquel encargado de pasar los valores de los controles a un número que represente la combinación de instrucciones suministrada por el sistema.

Programación del PIC18F4550

Este PIC es el encargado de transmitir los datos por medio del puerto USB hacia la computadora (figura 4.30). Cuenta con ocho pines del puerto B, mismo que son utilizados para recibir la información de las placas del sistema.

[6] El ADC8040 es una serie de convertidores analógico – digital que se encargan de recibir un voltaje y transformarlo en un valor binario de 8 bits, tomando de referencia un voltaje suministrado.

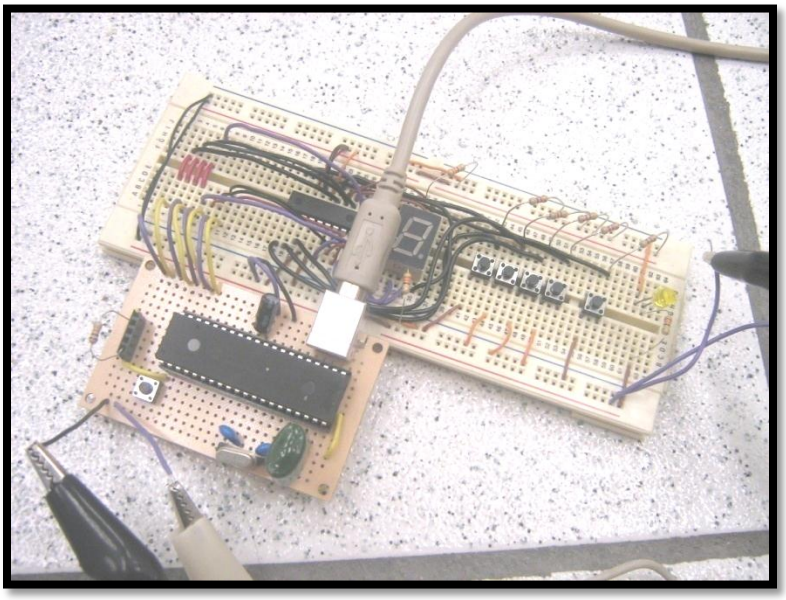

Figura 4.30 PIC18F4550 instalado para pruebas en protoboard

Gracias a este PIC, la comunicación vía puerto USB es posible. Proporciona una buena cantidad de pines de entrada / salida y puede ser programado de manera similar al PIC16F628A, con las mismas instrucciones pero se requieren diferentes características en el compilador.

Cabe mencionar que las consideraciones para elegir a esta familia de microcontroladores son variados, partiendo desde factores como el precio, la facilidad en cuanto a la programación y la documentación existente, hasta elementos más subjetivos como la experiencia en proyectos anteriores de comunicación por la misma tecnología USB.

Programa receptor de instrucciones

En el presente caso, no es posible llevar a cabo una conexión directa entre lenguaje Python y el microcontrolador PIC18. Por lo tanto, se requiere de otro elemento intermedio. El control USB se encarga de mandar datos hacia la computadora. A continuación, se encuentra un programa en VC++, que se ha dado por llamar *ControlUSB*, que traduce las señales provenientes del puerto en comandos que Python puede manejar de manera sencilla (figura 4.31).

Como un elemento extra, el programa muestra una animación de la dirección de desplazamiento y de la selección, en caso de que ocurra. Sin embargo, cuando se utilice el sistema, la pantalla puede esconderse para que el navegador no cuente con bloqueos en la pantalla.

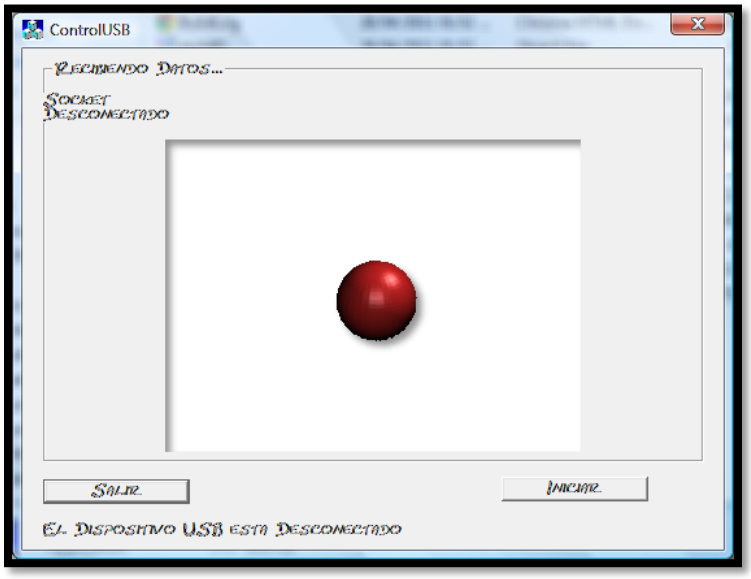

Figura 4.31 Pantalla de ControlUSB (intermediario)

Esta aplicación recibe la comunicación proveniente del microcontrolador. Mantiene en aviso al usuario de si el cable USB se encuentra conectado o si los sockets están activos. Al presionar el botón “Iniciar”, el sistema comienza a leer por el puerto indefinidamente. Aun así, la conexión puede finalizarse cuando se desee y el sistema no tendrá problemas para cerrarse.

La integración final del sistema puede apreciarse en la figura 4.32. Existen también algunos puntos a resaltar en cuanto al funcionamiento del sistema. Empezando por la separación de los elementos tangibles, es posible distinguir:

- Que se trata de un sistema de alta inmersión. Al ser considerado como una caverna (CAVE), el despliegue de la información se efectúa en el entorno, de manera envolvente. Esto aunado a efectos sonoros en estéreo y la correcta iluminación, produce en el usuario una sensación muy convincente de realidad. Entre más persuadido esté el usuario de que el entorno es real, se dice que el grado de inmersión es mayor.
- Los grados de libertad son suficientes. Esto significa que los grados de libertad con los que la cabina cuenta son suficientes para que su funcionamiento sea considerablemente bueno en escenarios de recorrido a pie. Sin embargo, es susceptible a mejoras con base a la sustitución de técnica de rastreo o de la misma instalación. Esto incrementa la inversión necesaria en el proyecto en gran medida, y con los resultados obtenidos actualmente se consigue un equilibrio aceptable.

- Sistema de rastreo. Para ubicar espacialmente a un usuario, es necesario contar con alguna técnica que permita esto. Para el caso de esta tesis, se puede separar al sistema en dos partes: rastreo óptico y sin origen. El primero se aplica al módulo de registro de movimiento, ya que por medio de sensores infrarrojos detecta los cambios. El segundo caso hace referencia al acelerómetro, que se encarga de monitorear la inclinación del dispositivo en relación a la gravedad o al movimiento de un usuario (éste se encuentra en el módulo de rotación). Por último también es importante hacer mención de la opción de “selección”, donde un botón que se encarga de esta función podría entrar en la categoría de los elementos de rastreo mecánicos, aunque propiamente no es útil para localizar al usuario espacialmente.

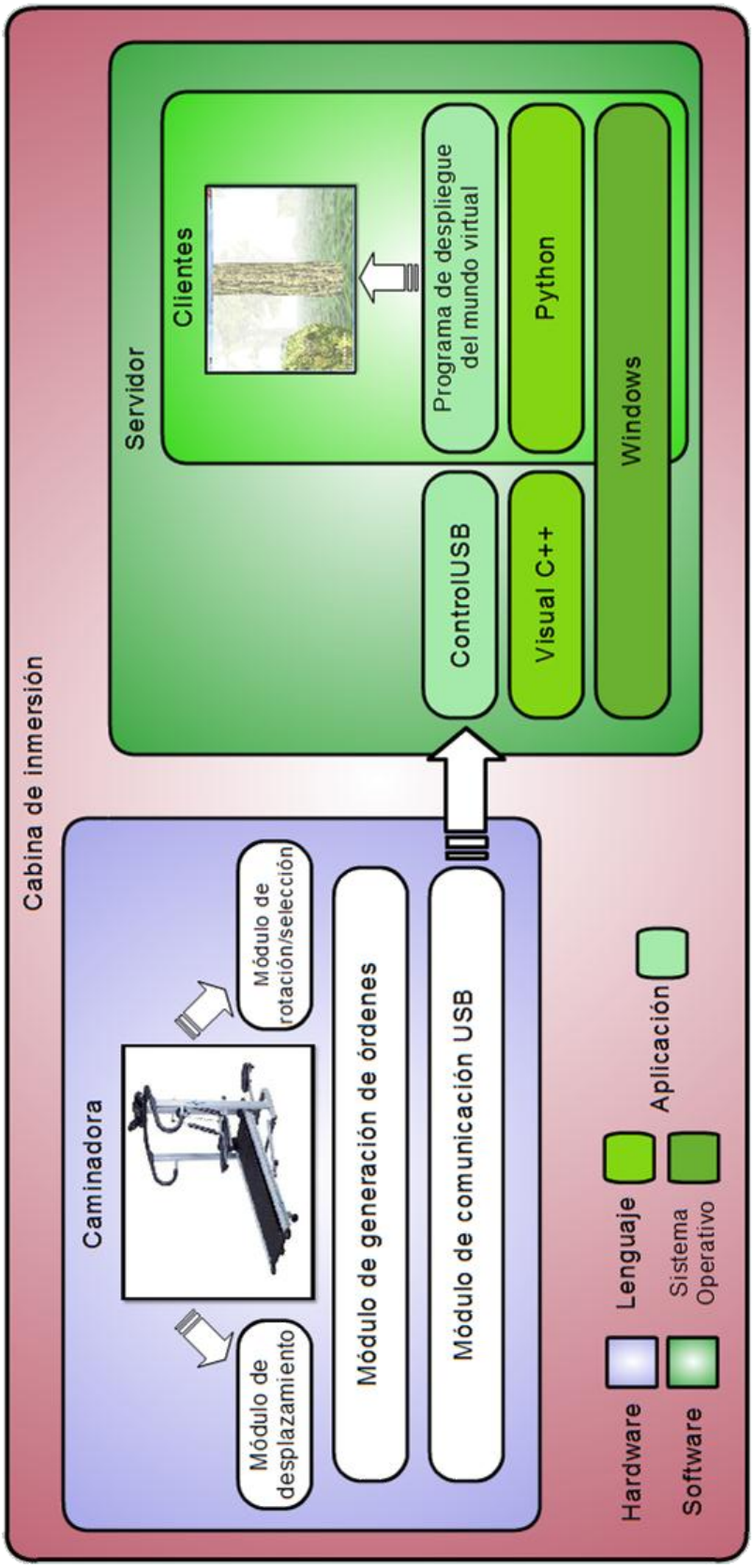

Figura 4.32 Diagrama estructural del sistema

Capítulo V. Experimentos y resultados

5.1 Introducción

Existen módulos o secciones (como la transmisión por puerto o la interpretación de los sensores) en donde es posible llevar a cabo una medición de rendimiento, velocidad, desempeño, etc. Por esto mismo, se realiza una comparativa con algunos factores variables con el fin de llegar a modificar el sistema, y con esto, obtener un mejor producto.

Para la tesis en cuestión, es necesario revisar la interacción de cada componente del sistema, incluido el usuario, con todos los demás. De esta forma se consigue un entorno equilibrado en el que todo módulo, probado de manera individual, cuenta con las entradas y salidas deseadas, desempeño adecuado y comodidad para el usuario a la hora de utilizar el dispositivo.

Ya que se tiene planteado el desarrollo, hay que escoger los elementos que más convengan a la construcción del prototipo. Por lo tanto, se puede dividir esta sección en las siguientes partes fundamentales: pruebas de comunicación entre equipos (en red), de transmisión de datos por el puerto USB y entre computadoras (órdenes), de desempeño de sensores tanto de desplazamiento como de rotación y pruebas de manejo por parte del usuario.

Se anexan imágenes de las pruebas y sus respectivos resultados para facilitar la comparación entre cada experimento. Cabe mencionar que en el caso de las tablas, se ha recurrido a una escala que simplifique la igualación. Los experimentos son calificados con cuatro posibles valores: ***Malo, Regular, Bueno y Excelente***. Esta calificación se determina en comparación con los límites de la escala, definidos en cada experimento. Los valores intermedios de la medición corresponden a su equivalente fraccionario de la escala utilizada.

5.2 Planteamiento de los experimentos

Para poder llevar a cabo algún experimento, se plantean las condiciones iniciales de cada uno, el proceso que se llevará a cabo, y que además será objeto de comparación, además de los resultados obtenidos en diferentes fases (velocidad de transmisión, funcionamiento adecuado de módulos, traducción correcta de comandos, recepción de información, etc.)

Más adelante se muestran los resultados de cada prueba y su correspondiente comparación entre cada experimento. Cabe hacer mención de que se deben realizar varias pruebas para asegurar un resultado ideal.

5.2.1 Comunicación en red

Existen una serie de dudas en cuanto a este tipo de comunicación, debido a que debe de ser veloz y llevar a cabo las funciones necesarias sin problema. Debido a esto, se pueden definir las variables a analizar en los experimentos, que en este caso son dos:

- Velocidad de transmisión de datos. Como se ha mencionado y recalcado, la rapidez con la que la información es recibida es crucial en este tipo de sistemas (interacción en tiempo real[7]). La calificación en este caso será tomando como peor nota una transmisión con retardo mayor a un segundo, y como mejor nota aquella que asemeje comunicación instantánea (diferencia del orden de milisegundos).
- Buen desempeño en conjunto (clientes y servidor). Ya en conjunto las computadoras deben de mostrar el ambiente simulado en perfecta sincronía, pero también debe verse un desplazamiento continuo de cada cuadro mostrado. Es decir, debe ser fluido el movimiento hacia cualquier dirección y para los cambios de orientación. Según el grado de fluidez (cuadros continuos mostrados de la escena en un flujo indetectable al ojo, de veinte cuadros hacia arriba, se considera de alta calificación, mientras que la notoria transición de cuadros igual o inferior a doce por segundo, resulta en un pésimo efecto) y la rapidez con la que se lleve a cabo la comunicación entre equipos (tomando como parámetro el mismo retardo del punto anterior), se puede encontrar un resultado favorable.

Para llegar a determinar el código definitivo del sistema, se plantea realizar experimentos con las siguientes combinaciones:

- Comunicación por protocolo TCP, transmisión de cambios de posición y dirección. Con un protocolo orientado a conexión se plantea el caso de transmitir la instrucción de avance, retroceso o giro hacia algún sentido a los clientes.

[7] Los sistemas en tiempo real se encargan de que exista respuesta casi instantánea, ya que se requiere una interacción automática.

- Comunicación por protocolo TCP, transmisión de posición y dirección absoluta. Con el mismo tipo de comunicación, se circulará a las demás computadoras una posición destino, en la cual cada equipo debe situar la cámara que visualiza el mundo.
- Comunicación por protocolo UDP, transmisión de cambios de posición y dirección. Cambiando el protocolo, se pretende medir qué tan eficiente resulta la transmisión de los cambios del servidor.
- Comunicación por protocolo UDP, transmisión de posición y dirección absoluta. Se efectúa la transmisión de la localización del usuario en el mundo virtual entre equipos, valiéndose del protocolo no orientado a conexión.

5.2.2 Comunicación instrucciones USB – Servidor

En la cuestión de manejo de datos hacia el equipo principal, existe el siguiente grupo de factores a considerar:

- Instrucciones a transmitir. Se refiere a la forma en que serán traducidas las órdenes provenientes del sistema de navegación de la caminadora.
- Desempeño de la simulación. Qué tanto repercutirá en la forma en que es visualizado el ambiente artificial con un número elevado de instrucciones.

En este apartado surgen las propuestas enlistadas a continuación, con el fin de determinar con qué tipo de comandos se alcanza un óptimo desempeño:

- Transmitir cada orden por separado. Cada orden del sistema será enviada individualmente al equipo, para ser reflejado en el ambiente virtual.
- Transmitir un conjunto de órdenes. En los casos en que exista una posible combinación o unión de instrucciones (como dar giro mientras se avanza), las órdenes pasarán a la computadora como una sola, que represente a las dos instrucciones.

5.2.3 Sensores para los controles

Para que el usuario sienta una mejora en los controles, debe de contarse con sensores que respondan a movimientos más naturales. Sin embargo, existen varios tipos de sensores que se

encargan de revisar diferentes estímulos del mundo real. Los sensores considerados en la tesis para efectuar pruebas son:

- Sensor infrarrojo. Registra una luz infrarroja que tiene frente a él a una distancia máxima de 15 cm. Si no detecta la luz, se obtendrá un voltaje de salida nulo. De lo contrario, arrojará 5 volts.
- Sensor de oclusión. Consta de un sensor infrarrojo y un LED puestos en un ángulo adecuado para que pueda ser detectada dicha luz por medio de un cuerpo reflejante. Opera a distancias cortas, del orden de los milímetros, y su respuesta se reduce a dejar pasar o no el voltaje suministrado.
- Acelerómetro. Es un sensor un poco más sofisticado, ya que detecta los cambios de orientación de uno a tres ejes (x, y o z). Existe una escala de la que depende el voltaje de salida. Si es muy pequeño, el movimiento es hacia la parte negativa del eje. En otro caso, se aproximará al voltaje de referencia.

De los elementos enlistados anteriormente, se seleccionará sólo un grupo para los controles. Para que sean escogidos los componentes adecuados se establecen algunas pruebas:

- Mediciones correctas. Si bien los sensores proporcionan una traducción de fenómenos físicos, a veces no pueden interpretarse adecuadamente para los controles que uno piensa. Si el sensor en cuestión, arroja datos que permitan una clara traducción a comandos del mundo virtual, la calificación del experimento será la mayor. En detrimento a la calidad de las mediciones (cuando la información del sensor se vuelve intermitente entre la respuesta adecuada y otra que no lo es, la calificación *Buena* y *Regular* dependen del orden de dicha intermitencia, siendo medible en minutos y en segundos respectivamente), se asignará una menor nota hasta el peor caso, considerado si la información del sensor no tiene ninguna relación con el fenómeno en cuestión.
- Buena instalación. Los componentes deben de poder fijarse a la caminadora de manera adecuada para censar los fenómenos correspondientes a su construcción. Cuando la instalación proporciona un campo de acción ideal para los sensores (no existen elementos que alteren las mediciones), se registrará el experimento con la nota más alta. Al tener el sensor colocado en posiciones que dificultan un reconocimiento adecuado de los cambios físicos que evalúa, podría reducir su calificación hasta el punto de no registrar ninguna variación (una *Buena* calificación significa una instalación sólida, pero difícil de conseguir por la estructura de

la caminadora; la evaluación *Regular* es aquella en la que la instalación es suceptible a desgaste o destrucción).

- Velocidad de respuesta de los sensores. Debe de existir una salida del sensor lo suficientemente rápida como para poder ser medida y traducida según la tarea a la que se enfoque. En el peor caso, la velocidad del sensor excederá el segundo, en cuanto a su transmisión. Si existe una respuesta instantánea, o con un retardo imperceptible, el resultado de la prueba contará con la mejor calificación.

5.2.4 Manejo del sistema

Unido a las pruebas del punto anterior, está el desempeño de los controles ya en manos de un usuario. Para hacer medible la comparación, se programan las siguientes variables de evaluación:

- Facilidad de uso. Ya que uno de los propósitos del trabajo de investigación es conseguir un alto grado de inmersión, los controles no deben de evitar esto. Se evaluará la naturalidad con la que pueden usarse los controles y la sencillez en su manejo. Si se determina que la utilización del sistema es a través de movimientos sencillos, será acreedor el experimento de la mejor nota (*Excelente*). Si durante la prueba, el control resultó ser poco intuitivo y de una clara dificultad para el usuario, se calificará como el experimento con peores efectos. Esta evaluación se realiza de forma subjetiva ya que depende de la experiencia del usuario y su elección de desempeño según la escala.
- Comodidad en su utilización. Aunque se cuente con un control de fácil uso y con una curva de aprendizaje pronunciada, existe el caso de que no sea muy cómodo para ciertos fines. Si se detecta molestia al manejar el sistema, llegando a incomodar al navegante del ambiente simulado a tal grado de causar dolor en articulaciones o movimientos difíciles, el resultado de la evaluación será “Mala”. Cuando el usuario refleje naturalidad y ninguna molestia al usar los sensores empotrados, se anotará una excelente calificación. De nuevo estos resultados dependen de la experiencia del sujeto que prueba el sistema.

La única especie de prueba a considerar, consiste en utilizar el sistema completo de navegación.

5.2.5 Navegación

Ya se han descrito anteriormente una serie de experimentos que involucran al entorno simulado y a los componentes físicos. Ahora bien, debe de considerarse además el funcionamiento de estos factores en conjunto.

Después de haber elegido los componentes adecuados para la comunicación y el desempeño de los elementos del sistema, se procede a evaluar la navegación, por lo que se fijan los siguientes aspectos a revisar (evaluados según la experiencia de cada usuario dentro de la escala de medición proporcionada, promediando todas las pruebas):

- Libertad. Cuando la sincronización está presente y además existe comodidad en los controles, el punto que corresponde ahora evaluar es qué tan bien puede uno desplazarse por el entorno, es decir, con cuanta libertad se cuenta en el mundo simulado. Al calificar el experimento, la valoración se hará en términos de movimientos permisibles en el ambiente virtual. Una excelente nota representa una amplia soltura y movimientos ágiles al momento de navegar, todo a gusto del usuario. Si esta libertad se encuentra reducida a tal grado que el usuario se siente incómodo, o los movimientos que desea efectuar no ocurren como se planean, la peor nota será anotada.
- Interacción. Se trata de revisar qué tan adecuada es la convivencia entre el usuario y el sistema. A pesar de medir la comodidad y facilidad de los controles, la interacción final con el entorno proporciona información acerca del desempeño. Si el usuario realmente percibe a los controles como una extensión de sus movimientos y utiliza los mismos de manera natural para efectuar el desplazamiento a través del entorno, será evaluada la prueba con la puntuación más alta. La calificación más baja será asignada a la prueba que refleje una pobre sensación de inmersión y poca naturalidad.

Cabe hacer mención de que los experimentos de este apartado se efectuaron tanto en el sistema elaborado en esta tesis como en el anterior desarrollo que involucra a la cabina de inmersión. Dicho trabajo es mencionado en el estado del arte.

5.3 Realización de experimentos

En el presente apartado se tratan las evaluaciones planteadas precedentemente. Al final de cada caso, se muestra un resumen general impreso en una tabla, con lo cual se facilita en entendimiento y la lógica en cada decisión referente a la tesis.

5.3.1 Comunicación en red

- Comunicación por protocolo TCP, transmisión de cambios de posición y dirección. La transmisión de la alteración de posición y/u orientación se consiguió. En cuanto a los factores a evaluar se obtuvo la siguiente información:

    - Velocidad de transmisión de datos. Fue el punto más afectado de los dos. Por el mismo protocolo, se tiene que mantener la sincronización a cada momento. Esto requería de una constante entrada / salida de información, o de lo contrario, volver a llevar a cabo la conexión cuando se requiriera. Esto causa un mayor retardo en el sistema.
    - Buen desempeño en conjunto (clientes y servidor). Aunque la transmisión de datos entre equipos no es la idónea, reaccionan bien las computadoras que fungen como cliente.

- Comunicación por protocolo TCP, transmisión de posición y dirección absoluta. También fue posible completarse la comunicación. Pero a diferencia del primer experimento, se obtuvieron las mediciones que se enlistan a continuación:

    - Velocidad de transmisión de datos. Existe el mismo problema que en el experimento anterior, ya que se tiene que establecer una conexión constante. Esto no permite que el cliente desempeñe alguna otra tarea hasta recibir la información del servidor.
    - Buen desempeño en conjunto (clientes y servidor). A diferencia de la computadora servidor, los clientes no muestran una fluidez en los cambios de posicionamiento y/u orientación, dando brincos de un punto a otro.

- Comunicación por protocolo UDP, transmisión de cambios de posición y dirección. Cambiando el protocolo existen ciertas variaciones. Ahora bien, los resultados cuentan que además, las similitudes apuntan a un perfil próximo a ser definitivo.

- Velocidad de transmisión de datos. Salta a la vista una clara ventaja: no se requiere mantener una conexión. El servidor es libre de mandar mensajes cuando tenga que mandar mensajes y no como una tarea obligada en cada ciclo del programa.
- Buen desempeño en conjunto (clientes y servidor). Un cambio constante en la escena proporciona al usuario una sensación de fluidez. Es decir, el ambiente virtual desempeña su tarea de despliegue con la misma calidad que ha hecho el servidor en todos los experimentos anteriores.

- Comunicación por protocolo UDP, transmisión de posición y dirección absoluta. Contando con la ventaja de un protocolo libre de conexión, sólo queda ver el funcionamiento del entorno en conjunto con las otras computadoras. A continuación puede apreciarse el resultado de la experimentación (Tabla 1):

  - Velocidad de transmisión de datos. Se aprecia de nuevo una comunicación si y solo si, esta es requerida. Esto se logra gracias a una función que siempre está ejecutándose y encolándose. Su papel es revisar si existe algún mensaje, y si lo hay, lo lee y manda a otra función encargada de interpretar la información.
  - Buen desempeño en conjunto (clientes y servidor). Con esto se comprueba la ineficacia de la transmisión de posiciones y direcciones absolutas. Se detecta el mismo índice de discontinuidad en los cuadros que conforman la escena. Debido a que los cambios no son tan veloces, el usuario puede ver que la imagen no es constante y hecha por los suelos la sensación de inmersión.

Tabla 1. Resultados de pruebas referentes a la comunicación en red

		Velocidad	*Desempeño*
TCP	Posición / Dirección Relativas	Buena	Bueno
TCP	Posición / Dirección Absolutas	Buena	Regular
UDP	Posición / Dirección Relativas	Excelente	Excelente
UDP	Posición / Dirección Absolutas	Excelente	Regular

5.3.2 Comunicación instrucciones USB - Servidor

- Transmitir cada orden por separado. Este caso consiste en dividir cada instrucción y mandarla por separado (aunque en un flujo constante). El experimento parte de considerar las siguientes órdenes:

  - Adelante. Movimiento hacia enfrente de la cámara de visualización del ambiente simulado aproximadamente cinco pasos.
  - Atrás. Movimiento hacia atrás, solo modificando el desplazamiento negativamente.
  - Izquierda. En el sistema, el comando refleja un cambio en la orientación de 10 grados por ciclo hacia la izquierda.
  - Derecha. Giro negativo en el entorno, que cambia la orientación hacia la derecha del usuario.
  - Selección. En el caso del presente trabajo, sólo se imprime un mensaje mencionando que se ha llevado a cabo la selección.

- Transmitir un conjunto de órdenes. Ya que se tiene definido el conjunto de instrucciones posibles del sistema, existe la posibilidad de unir ciertas órdenes en unas más complejas que requerirán, en teoría, de menor tiempo de transmisión y por lo tanto de mejor desempeño. Las combinaciones posibles son:

  - Adelante
  - Adelante/ Derecha
  - Adelante/ Izquierda
  - Atrás
  - Atrás/ Derecha.
  - Atrás/ Izquierda.
  - Izquierda
  - Derecha
  - Selección

La combinación de sentencias produce una acumulación mayor de instrucciones. Sin embargo, esto carece de importancia a simple vista. Ahora se cuenta con órdenes dirigidas al desplazamiento en diagonal. Esto se ha logrado gracias a la encapsulación de dos o más comandos simples.

Si se prosigue con la experimentación habrá dos variables: el desempeño (medido en cuanto a la visualización del movimiento) y si la cantidad de instrucciones afectará en algo al sistema. Puede existir una reducción de factores a considerar, únicamente el desempeño, ya que el número de

órdenes está ligado íntimamente con la transmisión combinada (9) o con la transmisión simple (5). Puede observarse el resultado a continuación (Tabla 2):

- Transmitir cada orden por separado. La comunicación ha sido veloz y los datos llegaron sin corrupción. Además, la visualización del entorno artificial fue continua y no surgió el caso esperado: una pausa entre avance y giro. Se tenía la hipótesis de que la escena haría una especie de *"zigzagueo"* al querer ir en diagonal. Sin embargo, si esto en realidad llega a ocurrir, no es apreciable para las personas.
- Transmitir un conjunto de órdenes. La transferencia de instrucciones como conjunto arroja resultados similares al experimento previo. La visualización es fluida, con buen desempeño y los controles no hacen aparente ningún efecto negativo. Ni siquiera parece afectar el número de instrucciones que casi duplica a la transmisión por separado, lo cual era de esperarse.

Tabla 2. Resultados de los experimentos referentes a la transmisión de órdenes Microcontrolador - Servidor y Servidor - Clientes

	Desempeño		
	Continuidad de movimiento	*Comunicación de Datos*	*Sincronización*
Órdenes por Separado	Excelente	Excelente	Excelente
Órdenes en Conjunto	Excelente	Excelente	Excelente

Para la evaluación del experimento, se muestra a continuación la manera en que se determina el puntaje:

- Continuidad de movimiento. Al proporcionar una serie de instrucciones en cola, la interpretación debe ser la adecuada para que el mundo virtual sea mostrado sin brincos en la imagen o con cambios drásticos. Un resultado "Excelente" es aquel que permita un flujo constante de escenas con respecto a las instrucciones enviadas (tiempo real); una salida con saltos perceptibles del ambiente refleja un fracaso en el experimento, siendo acreedor de la nota más baja (doce cuadros por segundo).
- Comunicación de datos. La información que es transmitida desde el exterior hacia el interior del equipo de cómputo, debe permanecer íntegra para asegurar su interpretación correcta. En este caso, solo se requiere una evaluación binaria. Cuando esta recopilación de datos llega completa a su destino, significa que los resultados fueron favorables (Excelente). De lo

contrario, cuando se obtiene el comando y no existe similitud, la calificación será negativa (Mala).

- Sincronización. Cuando una orden es enviada al equipo, debe existir un cambio instantáneo en el canal de salida (pantalla). Éste es el caso ideal de la prueba, donde parece que el ambiente simulado es navegado de acuerdo a los movimientos de la persona, como si se tratase del mundo real. El peor caso es aquel en que el retardo del comando y su reflejo en las computadoras hacen que se pierda la sensación tanto de inmersión como de realismo, llegando a demorarse más de un segundo. En el mejor de los casos, la respuesta es automática.

Por lo visto, en cuanto a desempeño no existen factores que ayuden a seleccionar una forma de proceder. El único factor que distingue los casos, como se hizo mención anteriormente, es el número de instrucciones que pueden manejar.

5.3.3 Sensores para los controles

En este apartado se considerarán mediciones tanto para el control de desplazamiento como para aquel encargado del giro en el entorno virtual. Con esto se consigue hacer una distinción de qué sensores serían adecuados para qué entorno de simulación, ya que no se puede suponer que daría igual el sentido de su uso.

- Sensor infrarrojo. En el control de giro, este sensor es colocado a un costado del usuario, en las cercanías de su extremidad, para que la presencia de su mano dentro del receptor refleje una señal de cambio de orientación (se requieren dos sensores. Es decir, uno para cada dirección).

  En el caso del desplazamiento, se aprovechan las condiciones de la caminadora y se hace uso de la rueda fusionada al camino sin fin. Por medio de un orificio en la misma, se hará pasar la luz hacia el receptor. Cuando sea bloqueado el flujo de iluminación, la señal reflejará movimiento.

  - Mediciones correctas. Dentro del contexto del control de giro, pudo verse una medición adecuada, ya que efectivamente se mandó un voltaje nulo al momento de existir iluminación incidente en el elemento sensor, consecuencia de cruzar y retirar la mano de

él. Ocurre algo similar para el control de desplazamiento, solo que la luz es bloqueada por la rueda del rodillo de la caminadora.

- Buena instalación. Para ambos casos la instalación pudro lograrse. En el primer experimento (giro) solo hizo falta fijar el componente al brazo de la caminadora. En la segunda prueba se tardó un poco más en poder instalar el dispositivo, pero se consiguió crear un soporte sencillo para esta.
- Velocidad de respuesta de los sensores. La respuesta es adecuada para el sensor de giro, ya que no puede reaccionar un ser humano en un tiempo tan breve como para evitar la salida del sensor. En cuanto al desplazamiento, el cambio de la salida tarda un poco más en llegar a cero, por lo que no se considera muy buena la rapidez.

- Sensor de oclusión. Resulta ser más complicado el experimento en este caso. Para el giro la instalación se asemeja a la prueba pasada, ya que el usuario tiene que tapar el sensor para que registre un cambio. De igual manera, se requieren o dos sensores o bien realizar dos pruebas separadas.

  Al acoplar este tipo de sensor al control de desplazamiento se hace uso de la rueda de nueva cuenta. En la orilla del disco de fija cinta blanca de aislar, que se pretende que refleje la suficiente luz para ser registrada, y el sensor es colocado a nomás de unos cuantos milímetros de la circunferencia.

  - Mediciones correctas. Se pudieron observar buenas mediciones en los dos experimentos propuestos, sin ninguna alteración a no ser que sea utilizado fuera de las condiciones necesarias (cercanía, contar con algún elemento reflejante, etc.). Para el giro se registra la oclusión con algún dedo y para la rueda de la caminadora (desplazamiento) se utiliza cinta blanca de aislar. Esta información es interpretable.
  - Buena instalación. En el dispositivo encargado de giro, se auxilió una vez más en el brazo que la caminadora ya posee, consiguiendo gran estabilidad. Sin embargo, para el movimiento hacia adelante y atrás, no se encontró una forma cómoda de instalar, más que utilizando un soporte de la propia caminadora.
  - Velocidad de respuesta de los sensores. Lentos resultados en el apartado de desplazamiento, ya que la rueda giraba a velocidades muy grandes. Para el elemento de cambio de dirección fue suficiente el tiempo que toma en regresar la información.

- Acelerómetro. Para ser implementado en el control de giro, se recurre al brazo de la caminadora por tercera vez. Esto debido a que es el elemento más cercano al usuario y en donde podría ser maniobrado un control adecuadamente. Cuando se desee girar hacia la derecha, el acelerómetro debe inclinarse hacia esa dirección. Ocurre lo opuesto si se requiere un cambio hacia la izquierda.

  Para la rueda no existen muchas formas seguras de fijar el dispositivo. Existen dos opciones: la primera es instalar el acelerómetro en el disco y medir la variación mientras se rota en conjunto; otra solución considerada es el colocar a muy corta distancia el sensor de la rueda, o el camino sin fin. Después debe existir contacto entre los cuerpos cercanos para que el acelerómetro “vibre”, por decirlo de algún modo, al existir movimiento.

  - Mediciones correctas. No ha existido hasta el momento alguna falla sobre este inciso. Al igual que los experimentos pasados, se recibieron muy buenos resultados. En los dos casos, el registro de alguna variación proviene del cambio de orientación del dispositivo.
  - Buena instalación. Sin problemas pudo colocarse el acelerómetro en el tubo que se localiza a un costado de la caminadora para el control de la orientación. Para el control del desplazamiento, no pudo diseñarse un artilugio que soportara el sensor de buena forma. La instalación que se planeaba realizar no fue posible llevarla a cabo (por vibración).
  - Velocidad de respuesta de los sensores. Existe una buena velocidad al probar el elemento con los controles de giro, debido a la relativamente lenta acción del usuario que pone en juego al sensor. Para el desplazamiento existió una velocidad de respuesta pobre, que se acerca bastante a la conseguida con los sensores infrarrojos en el mismo experimento.

  Por medio de la siguiente tabla (Tabla 3) es posible apreciar las mediciones:

Tabla 3. Resultados de las pruebas a los sensores de los controles

	Mediciones		Instalación		Velocidad de Respuesta	
	Giro	*Desplazamiento*	*Giro*	*Desplazamiento*	*Giro*	*Desplazamiento*
Sensor Infrarrojo	Excelente	Excelente	Excelente	Buena	Excelente	Buena
Sensor Oclusión	Excelente	Excelente	Excelente	Regular	Excelente	Regular
Acelerómetro	Excelente	Excelente	Excelente	Mala	Excelente	Regular

Las mediciones hacen visible la gran barrera que impone el elemento móvil de la caminadora, pues dificulta la instalación, y sin una buena instalación no durará mucho tiempo el control fijo y/o funcionando.

5.3.4 Manejo del sistema

Parte importante del sistema, si no es que de las más críticas. De este apartado depende la aceptación o rechazo del usuario al trabajo materializado, ya que si no proporciona la sensación de inmersión, perderá sentido que sea instalado en un cabina dirigida a este propósito.

Antes de continuar debe hacerse mención de los límites del experimento. Podría existir la suposición de que la prueba sea necesaria para los dos controles, de desplazamiento y giro. Sin embargo, dadas las variables a calificar (facilidad y comodidad de utilización), solo podría considerarse al control de cambios de orientación, debido a que el usuario no trata directamente con el segundo control.

- Sensor infrarrojo

    - Facilidad de uso. El sensor tiene un funcionamiento simple y sencillo para ser utilizado con las manos. Solo se requiere de un movimiento de brazo y muñeca para activar o desactivar la salida en alto (5 v).
    - Comodidad en su utilización. En esta cuestión, si existe una diferencia marcada, al contrario que en el caso de la facilidad de manejo, ya que para el usuario resulta algo incómodo el tener que buscar el área donde debe descansar la mano.

- Sensor de oclusión

    - Facilidad de uso. Se cuenta con un sensor de oclusión en las cercanías de la mano. Con tan solo colocar un dedo a una distancia muy pequeña puede recibirse una salida que más adelante será interpretada. Esto es una secuencia de pasos bastante simple que el navegante del ambiente recibe con agrado.
    - Comodidad en su utilización. No resulta muy cómodo su uso, ya que debe existir entre el elemento sensorial y el dedo muy poca distancia, sin llegar a cubrirlo completamente. Además, también debe de revisar la localización del componente.

- Acelerómetro

    - Facilidad de uso. No se necesita ninguna especie de conocimiento previo de utilización. Solo se trata de balancear el dispositivo, de manera similar al balanceo de los brazos al caminar (solo que en otro eje).
    - Comodidad en su utilización. Resultados excelentes en esta evaluación. No solo el usuario siente que el control de giro es muy natural, sino que pierde importancia el localizar algún elemento del mismo. No se cuenta con una sección particular que deba bloquearse como en los casos pasados. En esta prueba, el acelerómetro se mueve en conjunto.

En la Tabla 4 se pueden ver una representación de los resultados:

Tabla 4. Resultados de la experimentación con distintos controles de giro

	Facilidad	*Comodidad*
Sensor Infrarrojo	Excelente	Regular
Sensor Oclusión	Buena	Buena
Acelerómetro	Excelente	Excelente

Si consideramos que el experimento efectuado en el punto pasado, donde se toman en cuenta las mediciones de los sensores, comparte elementos con prueba actual, existe la posibilidad de fusionar los resultados y ampliar los parámetros de comparación para definir el camino que más convenga (Tabla 5).

Tabla 5. Reunión de resultados de los experimentos referentes a los controles de giro

	Mediciones	*Instalación*	*Velocidad de Respuesta*	*Facilidad*	*Comodidad*
Sensor Infrarrojo	Excelente	Excelente	Excelente	Excelente	Regular
Sensor Oclusión	Excelente	Excelente	Excelente	Buena	Buena
Acelerómetro	Excelente	Excelente	Excelente	Excelente	Excelente

5.3.5 Navegación

Para el experimento se requiere de la utilización de ambos sistemas, el actual y el que se está desarrollando, para medir las variables contempladas. Cada resultado ayuda a conocer las mejoras

existentes y también aquellas potenciales. Estas últimas, aunque no contempladas en la tesis, pueden incluirse en trabajos a futuro con el objetivo de fijar el paso siguiente al sistema en un posterior desarrollo.

- Sistema actual. Utiliza la caminadora y la cabina de inmersión de la misma forma que el trabajo en desarrollo. Por medio del puerto PS /2 que es utilizado para el ratón de la computadora, se lleva a cabo la comunicación del servidor con los elementos de la caminadora. Estos elementos pueden dividirse en dos: sensores de desplazamiento y de cambio de dirección. En el primer caso, se utiliza un sensor infrarrojo que revisa la recepción de luz para comunicar un avance. En cuanto al giro, un elemento mecánico montado en la parte delantera de la caminadora permite llevar a cabo esta acción.

    - Libertad. No se percibió alguna restricción en cuanto al movimiento en el ambiente simulado, ya que por medio de la banda sin fin el desplazamiento es el que el usuario desea y los giros son rápidos y hacia cualquier dirección. Sin embargo, si el usuario desea ir hacia atrás, el sistema no lo permite y tampoco la caminadora. A pesar de esto, no es frecuente recurrir a este tipo de desplazamiento. El usuario además puede permanecer sin movimiento y el sistema lo reconoce.
    - Interacción. Bastante buena, ya que el usuario puede caminar lo que desee y percibe al mundo artificial como si fuese real; además, el sistema permite alternar el control entre desplazamiento de usuario y del cursor en el entorno. Cuando se desee, la opción de interacción, entra en la escena y lleva a cabo su papel. Un punto que resulta algo incómodo es el intercambio entre el control de navegación y el de movimiento de cursor y también el mecanismo que genera el cambio de orientación.

- Sistema en desarrollo. Ya se ha revisado en qué consiste la tesis, así que solo se hace mención en esta sección de los elementos que varían entre los dos sistemas. Primero que nada está la comunicación a través del puerto USB, la cual aumenta la velocidad de transmisión de datos, libera el puerto del ratón para otras tareas y permite utilizar cualquier equipo como servidor (el puerto USB se encuentra en casi todos los equipos de cómputo). También se cambia el dispositivo que permite el giro y se fija en otra parte de la caminadora, obteniendo los siguientes resultados (Tabla 6):

- Libertad. Permite cualquier movimiento y cambio de orientación, incluido el desplazamiento hacia atrás. Sin embargo, por las mismas características de la caminadora (una pendiente que permite el avance de la banda sin fin) no es posible tener un cómodo movimiento de espalda. El giro es constante si el usuario lo permite y si se desea detener todo movimiento, también es posible.
- Interacción. A diferencia del sistema actual, no se puede intercambiar el control de navegación a cursor. Sin embargo, se agrega la opción de selección que permite llevar a cabo cierto grado de interacción con el ambiente. Este elemento puede crecer posteriormente y ampliarse su alcance. En cuanto a los controles de desplazamiento y cambio de dirección, resultaron una gran herramienta que proporciona sensación de inmersión de alto grado.

Tabla 6. Resultados de las pruebas entre sistemas

	Libertad	*Interacción*
Sistema Actual	Buena	Buena
Sistema en Desarrollo	Excelente	Buena

5.4 Evaluación de resultados

5.4.1 Comunicación en red

Tomando en cuenta los resultados impresos en la tabla, y las condiciones y comentarios apuntados anteriormente, la conclusión es que el modelo a seguir debe estar basado en comunicación por protocolo UDP y con paso de posición y dirección relativa.

Las mediciones que siguieron en calidad fueron de TCP/Relativo y UDP/Absoluto. En el primero de los casos, porque el desempeño fue muy pobre, apenas entrando en lo considerado “bueno”. El experimento de protocolo TCP con manejo de posiciones y rotación absoluta fue muy malo, ya que el protocolo alentaba el desempeño del muestreo del mundo virtual, debido a la conexión siempre existente. Aunado a esto, el paso de coordenadas y no de órdenes permitía apreciar saltos de un punto a otro en la pantalla.

5.4.2 Comunicación instrucciones USB - Servidor

Como se concluyó en la sección pasada, no existe más que un factor discontinuo entre los resultados: el número de instrucciones. Debido a que no cambia el desempeño se cambiamos de conjunto de órdenes, esto queda libre al desarrollador.

Tomando en cuenta la facilidad en utilizar pocos envíos, se opta por dejar atrás las cinco instrucciones simples y tener un registro de nueve. De esta manera, se reducirá la transmisión entre computadores y el proveniente del microcontrolador.

5.4.3 Sensores para los controles

Hasta el momento, se tienen resultados que no pueden entregar una conclusión clara en cuanto a los controles de giro, donde todos los experimentos fueron excelentes. Para la sección del sistema encargada del desplazamiento, tuvo una ligera ventaja el sensor infrarrojo en cuanto a instalación y respuesta. Los factores que fueron deficientes podrían llegar a mejorar con ciertas medidas a base de ADC (velocidad de respuesta).

En cuanto se revisen los resultados del experimento del manejo del sistema, que viene a continuación, se podrá concluir mejor qué elemento debe utilizarse para el control de cambios de dirección, ya que incluye mediciones referentes al desempeño en el manejo del dispositivo (facilidad de uso, comodidad).

5.4.4 Manejo del sistema

Gracias a los previos resultados y a las pruebas finales, se puede concluir que el sensor adecuado es aquel que registra la inclinación o movimiento. Si se toma en cuenta que obtuvo excelentes calificaciones en cada caso, no es posible pensar en alguna mejor decisión que tomar al elemento y volverlo un componente medular del control de orientación.

El acelerómetro lo tiene todo: comodidad en cuanto a su uso, buenas mediciones, velocidad adecuada, etc. Si también se aprecia la relación de su funcionamiento con el andar de una persona, es sencillo percibir por qué surgió el elemento de entre los demás sensores.

5.4.5 Navegación

Aunque los resultados se esperaban más alentadores, es natural pensar que no se ha alcanzado la interacción total de los sistemas con el usuario. Por una parte (sistema actual) no existe un control de giro natural y esto reduce la inmersión; del otro lado de la comparativa (sistema en desarrollo) no se tiene ninguna opción que permita usar el cursor para desencadenar eventos, manejar objetos, etc., y por eso ambos se evaluaron como “buenos”. En cuanto a la libertad, la única variable que le dio un puntaje más alto al trabajo de tesis en construcción, es que existe la posibilidad de un desplazamiento hacia atrás, añadiendo un poco más de independencia al usuario.

Las pruebas han sido satisfactorias en el sentido de que han producido tanto buenos como malos resultados, y esto es esencial para tomar decisiones en el desarrollo. Gracias a la consideración de tiempos de respuesta, velocidad de transmisión y funcionamiento adecuado, las modificaciones permitieron que el sistema se mantuviera en excelentes condiciones de desempeño.

Capítulo VI. Conclusiones y trabajo futuro

Se diseñó un sistema de navegación de realidad virtual basado en un camino sin fin utilizando una conexión USB como herramienta, además de que se consiguió un mayor grado de inmersión en los usuarios de realidad virtual con respecto al sistema anterior.

En cuanto a la definición de la solución tecnológica para el hardware y software a usar, se generaron las placas, se seleccionaron sensores, se optó por programación en Python, Visual C++, sockets e hilos.

Para la adecuación se revisaron los mecanismos tanto de la caminadora como del cuerpo al caminar, concluyendo con el uso del acelerómetro para control de giro y el sensor infrarrojo para el registro del desplazamiento.

Para la interfaz entre el usuario y el servidor, se diseñó un control sencillo pero que cumple con la función de giro, además de agregar la funcionalidad de selección sin requerir de la visualización del control.

Para finalizar con los objetivos particulares alcanzados, se ha probado el prototipo con un ritmo promedio de caminado y se ha mejorado su presentación y montaje con la caminadora.

Inicialmente, el sistema cuenta aún con mucho potencial y puede irse tratando de alcanzar en posteriores trabajos. Sin embargo, con la constante mejora de las tecnologías es muy probable que se requiera una actualización de igual manera constante.

Debido a las grandes cantidades de información acerca de la realidad virtual, se puede acercar uno de mejor forma a sus principios y comprender qué es lo que se necesita primero, luego que paso debería seguir, y así sucesivamente, cumpliendo con un trabajo que utilice la realidad virtual tal cual y como se define, con sus partes bien determinadas y un razonamiento dirigido a lo que se necesita mejorar o implementar.

Existen algunos puntos a resaltar en cuanto al funcionamiento del sistema. Empezando por la separación de los elementos tangibles, es posible distinguir:

- Que se trata de un sistema de alta inmersión. Al ser considerado como una caverna (CAVE), el despliegue de la información se efectúa en el entorno, de manera envolvente. Esto aunado a efectos sonoros en estéreo y la correcta iluminación, produce en el usuario una sensación muy convincente de realidad.
- Los grados de libertad son suficientes. Esto significa que los grados de libertad con los que la cabina cuenta permiten que su funcionamiento sea considerablemente bueno. Sin embargo, es susceptible a mejoras con base a la sustitución de técnica de rastreo o de la misma instalación. Esto incrementa la inversión necesaria en el proyecto en gran medida, y con los resultados obtenidos actualmente se consigue un equilibrio aceptable.
- Sistema de rastreo. Para ubicar espacialmente a un usuario, es necesario contar con alguna técnica que permita esto. Para el caso de esta tesis, se puede separar al sistema en dos partes: rastreo óptico y sin origen. El primero se aplica al módulo de registro de movimiento, ya que por medio de sensores infrarrojos detecta los cambios. El segundo caso hace referencia al acelerómetro, que se encarga de monitorear la inclinación del dispositivo en relación a la gravedad o al movimiento de un usuario (éste se encuentra en el módulo de rotación). Por último también es importante hacer mención de la opción de “selección”, donde un botón que se encarga de esta función podría entrar en la categoría de los elementos de rastreo mecánicos, aunque propiamente no es útil para localizar al usuario espacialmente.

Las pruebas han sido satisfactorias en el sentido de que han producido tanto buenos como malos resultados, y esto es esencial para tomar decisiones en el desarrollo. Gracias a la consideración de tiempos de respuesta, velocidad de transmisión y funcionamiento adecuado, las modificaciones permitieron que el sistema se mantuviera en excelentes condiciones de desempeño.

Al final se han llegado a concluir diversos puntos. Primero, que nada está dado empíricamente, en el caso de los trabajos de investigación. No pueden existir suposiciones que no sean susceptibles de duda, ya que el trabajo de un investigador es siempre estar cuestionando el entorno y el porqué de los fenómenos.

6.1 Aportes

El principal aporte de este trabajo es la utilización del puerto USB para los comandos de navegación en el mundo virtual. Antes de esto, se hacía por el puerto PS / 2 (puerto destinado al ratón de la computadora) y esto limitaba el uso de este dispositivo para otras funciones. Con esta

ventaja, los equipos que pueden utilizarse para el sistema prácticamente son todos los que existen, puesto que el USB ya es un estándar. También entra en este apartado la mención de los controles de navegación y su localización. Los controles están elaborados para que el usuario sea capaz de usarlos de forma natural y con poco conocimiento previo. Sin embargo, el verdadero aporte en este rubro viene con la instalación del control. Al encontrarse estratégicamente en el brazo de la caminadora, la persona que utiliza la cabina puede manipularlo fácilmente y con movimientos simples.

En cuestiones de lógica del producto, el programa que se encarga de recibir las órdenes del microcontrolador incorpora ciertas ventajas como son hilos y comunicación por medio de *sockets*. El primero, está enfocado a recibir los comandos paralelamente a la impresión del ambiente en la cabina, permitiendo una mayor agilidad al momento de utilizar el sistema.

6.2 Productos

Como parte de los resultados del trabajo de investigación, se encuentran los siguientes productos:

- Congresos (Internacional). 6° Congreso Internacional *Tendencias Tecnológicas en Computación* 2010.
- Publicaciones (nacional). *Mapeo de texturas en Blender*, Boletín Upiita (20), enero 2011.
- Prototipo

6.3 Trabajo futuro

Ya que se utilizó uno de los puertos con mayor velocidad hasta nuestros días, entra en escena un nuevo marco de trabajo: la comunicación inalámbrica. Si se cuenta con esta tecnología dentro del sistema se eliminarían los siguientes factores: dependencia de cables. A primera vista, es la clara ventaja que se obtiene. Sin los cables de por medio, la cabina se ve más limpia (sin cables por el suelo, la sensación de inmersión aumenta), la fijación de controles se vuelve más libre y existe la posibilidad de un cambio de localización; se reduce la probabilidad de accidentes en la cabina y el posible daño a la estructura del sistema.

A futuro, y con inversiones fuertes y trabajo riguroso, sería viable que algún mecanismo sustituya a la caminadora para aumentar aún más el efecto de realismo para el usuario, como puede ser un camino sin fin bidimensional y de movimiento automático. Si aunado a esto se posibilita el uso de cascos que permitan visión tridimensional, guantes que aumenten el número de capacidades del usuario y/o equipo de sonido envolvente, la cabina se convertirá en un aislamiento completo de la realidad para enviar a los usuarios a fantásticos viajes simulados.

En cuanto a los componentes electrónicos, es factible sustituir los PIC16F628A por otra clase de microcontrolador que integre un convertidor analógico digital y así empezar a reducir componentes, espacio, presentación e inversión. Para los elementos programables existen varias posibles mejoras, como la detección de colisiones, la manipulación de cuerpos en el mundo virtual y la determinación de un solo lenguaje de programación multiplataforma para aumentar el grado de portabilidad.

De las conclusiones anteriores, a corto plazo lo más viable que puede ser mejorado en cuanto al uso del sistema de navegación de ambientes virtuales, puede ser la comunicación tanto del exterior con el servidor, así como entre los equipos. Esto va a mejorar el tiempo de respuesta hacia los clientes y proporciona una posible utilización a distancia del sistema.

Referencias

[1] González Aspera A., Chávez Hernández G., “La Realidad Virtual Inmersiva en Ambientes Inteligentes de Aprendizaje, Un caso en la educación superior”, Revista de Comunicación y Nuevas Tecnologías *ICONO14*, Año 9, Vol. 2, pp. 122 – 137, España, Mayo 2011.

[2] Darken Rudolph P., Cockayne William R., Carmein David, “The Omni-Directional Treadmill: A Locomotion Device for Virtual Worlds”, en *UIST ’97*, Canadá, 1997.

[3] Ávila Valdés Noemí, “Interactividad y arte interactivo. La Realidad Virtual Inmersiva”, en *Arte, individuo y sociedad,* No. 15, pp. 163 – 168. Servicio de Publicaciones de la Universidad Complutense de Madrid, España, 2003.

[4] Domingues Diana, “TRANS-E, mi Cuerpo, mi Sangre”, en *7a Bienal de La Habana*, Cuba, 2001.

[5] Davies Char, "Osmose: Notes on Being in Immersive Virtual Space", *Digital Creativity*, Vol. 9, No. 2, Swets & Zeitlinger Publishers, Los Países Bajos, 1998.

[6] Lemoine Patrick, Gutiérrez Mario, Thalmann Daniel, Vexo Frederic, “The ‘Caddie Paradigm’: a Free-Locomotion Interface for Teleoperation”, Swiss Federal Institute of Technology at Lausanne, Suiza, 2004.

[7] Iwata Hiroo, Yano Hiroaki, Fukushima Hiroyuki, Noma Haruo, “CirculaFloor”, Universidad de Tsukuba, Japón, 2005.

[8] Olguín Carbajal Mauricio, Rivera Zarate Israel, Pozas Quiteria Oliver, “Desarrollo de un Sistema Inmersivo de Realidad Virtual basado en Cabina Multipersonal y Camino sin Fin”, en *Polibits*, No. 37, CIDETEC-IPN, México, 2008.

[9] Villar Nicolas et al., “Mouse 2.0: Multi-touch Meets the Mouse”, en *UIST ’09*, Canadá, 2009.

[10] Hilsendeger Andre, Brandauer Stephan, Tolksdorf Julia, Fröhlich Christian, “Navigation in Virtual Reality with the Wii Balance Board”, Universidad de Bielefeld, Alemania, 2009.

[11] Caballero David, “Cómo funciona el Wiimote”, 2006. [En línea]. Disponible: http://www.revogamers.net/article.php?articleId=35&page=4

[12] Pino Daniel, “PlayStation Move. Análisis, opiniones y a fondo”, 2010. [En línea]. Disponible: http://videojuegos.tuexperto.com/2010/07/13/playstation-move-analisis-opiniones-y-a-fondo

[13] Nurakhmed Nurislamovich Latypov, ”Method and Apparatus for Immersion of a User into Virtual Reality”, U.S. Patent 5,846,134, Diciembre 8, 1998.

[14] *Webster’s New Universal Unabridged Dictionary*. New York: Barnes & Noble Books, 1989.

[15] Burdea Grigore, Coiffet Philippe, *Virtual Reality Technology*. A Wiley – Interscience Publication. John Wiley & Sons, Inc., Edición traducida al inglés, 1994.

Anexos

Código principal ControlUSB

```
#include "stdafx.h"
#include "USBProject.h"
#include "USBProjectDlg.h"
#include "mcHID.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//codigo encargadod el despliegue en pantalla

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();
	enum { IDD = IDD_ABOUTBOX };
protected:
	virtual void DoDataExchange(CDataExchange* pDX);
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
//{{AFX_MSG_MAP(CAboutDlg)
// No hay manejador de mensajes
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
CUSBProjectDlg::CUSBProjectDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CUSBProjectDlg::IDD, pParent)
{
//variables usadas para la transmision
	opc = 99;
	port = 666;
	ipAddress = "localhost";
	Player = new CAnimateCtrl;
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CUSBProjectDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CUSBProjectDlg, CDialog)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_WM_CREATE()
	ON_MESSAGE(WM_HID_EVENT, OnHIDEvent)
	ON_BN_CLICKED(IDOK, &CUSBProjectDlg::OnBnClickedOk)
	ON_BN_CLICKED(IDC_BUTTON1, &CUSBProjectDlg::OnBnClickedButton1)
END_MESSAGE_MAP()

BOOL CUSBProjectDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	RECT Recto = { 110, 70, 360, 360 };
	Player->Create(WS_CHILD | WS_VISIBLE | ACS_TRANSPARENT | ACS_AUTOPLAY,Recto, this, 0x1884);
	Player->Open("C:\\ControlUSB\\VisualC-Mod\\nada.avi");
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	SetIcon(m_hIcon, TRUE);			// icono grande
	SetIcon(m_hIcon, FALSE);		// icono pequeño
	return TRUE;  // regresa TRUE a menos que se asigne el foco al control
}
```

```
void CUSBProjectDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

void CUSBProjectDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // contexto para el pintado de pantalla
		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
		// centra icono
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;
		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

HCURSOR CUSBProjectDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

/*
*********************************************************************************
* Nombre  : OnCreate
* Descripcion : Crea el dialogo y conecta a la DLL de HID
*********************************************************************************
*/
int CUSBProjectDlg::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
  if (CDialog::OnCreate(lpCreateStruct) == -1)
    return -1;

  // conecta a la DLL de HID
  CWnd *HostWnd = AfxGetMainWnd();
  if (LoadHID() == 0)
    Connect(HostWnd->m_hWnd);

  return 0;
}
```

```
/*
********************************************************************************
* Nombre  : OnHIDEvent
* Descripcion : Manejador de mensajes para todo evento del HID
********************************************************************************
*/
LRESULT CUSBProjectDlg::OnHIDEvent(WPARAM wParam, LPARAM lParam)
{
   CStatic* Label = (CStatic*) GetDlgItem(IDC_STATIC);
   CStatic* LabelSock = (CStatic*) GetDlgItem(IDC_STATIC2); // Conexion Socket
   char Text[0xFF];                    // buffer de texto
   UINT DevHandle;                     // manejador del dispositivo HID
   HIDBufferIn BufferIn;               // buffer de lectura del HID
//Arreglo del buffer de entrada 8 bits
   BufferIn[0]=0;
   BufferIn[1]=0;
   BufferIn[2]=0;
   BufferIn[3]=0;
   BufferIn[4]=0;
   BufferIn[5]=0;
   BufferIn[6]=0;
   BufferIn[7]=0;

   DevHandle = lParam;

   switch(wParam)
   {
            // Un dispositivo HID ha sido conectado...
                  case NOTIFY_PLUGGED:
                  {
                        // Si es nuetro HID...
                        if (GetVendorID(DevHandle) == VENDOR_ID && GetProductID(DevHandle) ==
PRODUCT_ID)
                        {
                              GetProductName(DevHandle,Text,0xFF);
                              //aviso de conexion
                              Label->SetWindowText(CString("El Dispositivo USB '") +
                                    CString(Text) +  CString("' esta Conectado"));
                              //establece conexion
                              sockClient.ConnectToServer(ipAddress,port);
                              //establece emnsaje a enviar
                              strcpy_s(sendMessage,256,"OK");
                              //envia mensaje
                              sockClient.SendData(sendMessage);
                              //aviso de conexion de socket
                              LabelSock->SetWindowText(CString("Socket Conectado"));
                              CStatic* boton1 = (CStatic*) GetDlgItem(IDC_BUTTON1);
                              CStatic* boton2 = (CStatic*) GetDlgItem(IDC_BUTTON2);
                        }
                        return 1;
                  }

                  // un HID ha sido desconectado
                  case NOTIFY_UNPLUGGED:
                  {
                        CStatic* boton1 = (CStatic*) GetDlgItem(IDC_BUTTON1);
                        CStatic* boton2 = (CStatic*) GetDlgItem(IDC_BUTTON2);
                        if (GetVendorID(DevHandle) == VENDOR_ID && GetProductID(DevHandle) ==
PRODUCT_ID)
```

```
// ha ocurrido algun evento con el HID
// esto ocurre despues de que todo mensaje de conexion o desconexion haya sido desplegado...
case NOTIFY_CHANGED:
{
	DevHandle = GetHandle(VENDOR_ID, PRODUCT_ID);
	SetReadNotify(DevHandle, TRUE);
	return 1;
}
case NOTIFY_READ:
{
	if(opc==1){
		//dependiendo la lectura, se escoje la orden
		Read(DevHandle, BufferIn);
		opc=BufferIn[2];
		switch(opc){
			case 1:  //Adelante
				strcpy_s(sendMessage,256,"UP");
				Player->Open("C:\\ControlUSB\\VisualC-Mod\\up.avi");
			break;
			case 79: //Atras
				strcpy_s(sendMessage,256,"DOWN");
				Player->Open("C:\\ControlUSB\\VisualC-Mod\\down.avi");
			break;
			case 18: //Izquierda
				strcpy_s(sendMessage,256,"LEFT");
				Player->Open("C:\\ControlUSB\\VisualC-Mod\\left.avi");
			break;
			case 6:  //Derecha
				strcpy_s(sendMessage,256,"RIGHT");
				Player->Open("C:\\ControlUSB\\VisualC-Mod\\right.avi");
			break;
			case 76: //Adelante - Izq
				strcpy_s(sendMessage,256,"ULEFT");
				Player->Open("C:\\ControlUSB\\VisualC-Mod\\uleft.avi");
			break;
			case 36: //Adelante - Der
				strcpy_s(sendMessage,256,"URIGHT");
				Player->Open("C:\\ControlUSB\\VisualC-Mod\\uright.avi");
			break;
			case 32: //Atras - Izq
				strcpy_s(sendMessage,256,"DLEFT");
				Player->Open("C:\\ControlUSB\\VisualC-Mod\\dleft.avi");
			break;
			case 15: //Atras - Der
				strcpy_s(sendMessage,256,"DRIGHT");
				Player->Open("C:\\ControlUSB\\VisualC-Mod\\dright.avi");
			break;
			case 0:  //Seleccionar
				strcpy_s(sendMessage,256,"SELECT");
				Player->Open("C:\\ControlUSB\\VisualC-Mod\\click.avi");
			break;
			case 127:	// Nada
				strcpy_s(sendMessage,256,"NADA");
				Player->Open("C:\\ControlUSB\\VisualC-Mod\\nada.avi");
			break;
		}
```

```
                                        //se manda la orden
                                        SendSomeData(opc);
                                        if (banderaRep!=opc){
                                                //solo se manda si la orden es diferente a la anterior
                                                banderaRep=opc;
                                                sockClient.SendData(sendMessage);
                                        }
                                }
                                opc=1;
                                return 1;
                        }
                }
                return 0;
}

//funcion encargada de la transmision
void CUSBProjectDlg::SendSomeData(int d)
{
  HIDBufferOut BufferOut;
  BufferOut[0] = 0;
  BufferOut[3] = d;
  WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
}

void CUSBProjectDlg::OnBnClickedOk()
{
        sockClient.CloseConnection();
        OnOK();
        OnClose();
}

//al presionar el boton de inicio, se avisa e inicializan variables
void CUSBProjectDlg::OnBnClickedButton1()
{
        SendSomeData(1);
        banderaRep=666;
        opc=1;
}
```

Código de interpretación de movimiento (Módulo de desplazamiento)

```
@ DEVICE MCLR_OFF, INTRC_OSC, WDT_OFF, LVP_OFF, BOD_OFF, PWRT_OFF, PROTECT_OFF

sensor1 var byte
sensor2 var byte
ver1 VAR BYTE
ver2 VAR BYTE
contStop var byte

CMCON = 7                'Entradas 0->sensor 1, 1->sensor 2
TRISB = $03              'Salidas 2->adelante, 3->atras
contStop = 0             ‘variable que controla el alto

Inicio:
PORTB.2 = 1
PORTB.3 = 1
pause 10
contStop = 0
‘si se active antes PORTB.0, se va hacia atras. Si primero se activa PORTB.1, se avanza
if (PORTB.0 = 0) then
   goto Atras
else
   if (PORTB.1 = 0) then
      goto Adelante
   endif
endif
‘es un ciclo infinito
goto inicio

'PASO ADELANTE
Adelante:
PORTB.2 = 0      ‘Salida Adelante
PORTB.3 = 1      ‘Salida Atrás
‘si el contador registra inactividad en 500 ms, se detiene el andar
pause 10
if (contStop < 50) then
   if (PORTB.0 = 0 or PORTB.1 = 0) then
      contstop = 0
   endif
   contStop = contStop + 1
   goto adelante
else
   goto inicio
endif
```

```
'PASO ATRAS
Atras:
PORTB.2 = 1                ‘Salida Adelante
PORTB.3 = 0                ‘Salida Adelante
‘si el contador registra inactividad en 500 ms, se detiene el andar
pause 10
if (contStop < 50) then
   if (PORTB.0 = 0 or PORTB.1 = 0) then
      contstop = 0
   endif
   contStop = contStop + 1
   goto atras
else
   goto inicio
endif

end
```

Código de interpretación de señales (Módulo de rotación y selección)

```
@ DEVICE MCLR_OFF, INTRC_OSC, WDT_OFF, LVP_OFF, BOD_OFF, PWRT_OFF, PROTECT_OFF

limiteIzq var byte
limiteDer var byte
entradaAct VAR BYTE

CMCON = 7
TRISA = $00         'Salidas   A2-> Izquierda, A3-> Derecha
TRISB = $FF         'Entradas
‘limites para registrar giro a la derecha o izquierda, determinados por experimentacion
limiteDer = %01110111
limiteIzq = %10000111
PORTA = %00001100
Ciclo:
   entradaAct = PORTB
   if (entradaAct < limiteDer) then
      PORTA = %00000100
   else
      if (entradaAct > limiteIzq) THEN
         PORTA = %00001000
      else
         PORTA = %00001100
      endif
   endif
   goto ciclo
end
```

Código de recepción de órdenes y conversión (Módulo de generación de órdenes)

```
@ DEVICE MCLR_OFF, INTRC_OSC, WDT_OFF, LVP_OFF, BOD_OFF, PWRT_OFF, PROTECT_OFF

i var byte
j var byte
sec var byte[10]
display VAR BYTE[10]

CMCON = 7
TRISA = $FF        'Entradas
TRISB = 0          'Salidas

'Acciones -> X,X,X,Ade,Atr,Izq,Der,Sel

sec[0]=%11101111     '239
sec[1]=%11110111     '247
sec[2]=%11111011     '251
sec[3]=%11111101     '253
sec[4]=%11101011     '235
sec[5]=%11101101     '237
sec[6]=%11110011     '243
sec[7]=%11110101     '245
sec[8]=%11111100     '254 252
sec[9]=%11111111     '255

‘numeros en el display
display[0]=%10000001   '%01111110
display[1]=%11001111   '%00110000
display[2]=%10010010   '%01101101
display[3]=%10000110   '%01111001
display[4]=%11001100   '%00110011
display[5]=%10100100   '%01011011
display[6]=%10100000   '%01011111
display[7]=%10001111   '%01110000
display[8]=%10000000   '%01111111
display[9]=%11111111   '%00000000
```

```
ciclo:
  i = PORTA
  j = 0
‘el ciclo busca que orden concuerda con la información de entrada y asigna su salida adecuada
  revisa:
    if (i = sec[j]) THEN
      PORTB = display[j]
      j = 10
    else
      j = j + 1
      if j < 10 then
        goto revisa
      else
        if j = 10 then
          PORTB = display[9]
        endif
      endif
    endif
‘antirrebotes
    pause 200
  goto ciclo
end
```

Código de transmisión de órdenes al equipo (Módulo de comunicación USB)

```
DEFINE OSC 48

i var byte

USBBufferSizeMax   con 8  ' tamaño maximo del buffer
USBBufferSizeTX    con 8  ' tamaño buffer entrada
USBBufferSizeRX    con 8  ' tamaño buffer salida

USBBuffer       Var Byte[USBBufferSizeMax]
USBBufferCount  Var Byte
TRISB = $FF

‘ciclo encargado de envio/recepcion
usbinit
ProgramStart:
  gosub DoUSBIn
  gosub DoUSBOut
  goto ProgramStart

DoUSBIn:
  USBBufferCount = USBBufferSizeRX            ' tamaño buffer RX
  USBService                                  ' mantiene viva la conexion
  USBIn 1, USBBuffer, USBBufferCount, DoUSBIn   'lee datos, si estan disponibles
  IF (USBBuffer[2] = 0) THEN
     i = 1
  ELSE
     i = 0
  ENDIF
  return

DoUSBOut:
  USBBufferCount = USBBufferSizeTX            ' tamaño buffer TX
  USBBuffer[1] = PORTB
  USBService                                  ' mantiene la conexion viva
  USBOut 1, USBBuffer, USBBufferCount, DoUSBOut ' si el bus esta disponible, transmite
  return
```

Código de despliegue del ambiente virtual e interacción de controles (Servidor)

```
import direct.directbase.DirectStart
from pandac.PandaModules import *
from direct.showbase import DirectObject
from direct.gui.DirectGui import *
from direct.gui.OnscreenImage import OnscreenImage
from direct.task import Task
from pandac.PandaModules import ConnectionManager
from pandac.PandaModules import QueuedConnectionReader
from pandac.PandaModules import ConnectionWriter
from direct.distributed.PyDatagram import PyDatagram
from direct.distributed.PyDatagramIterator import PyDatagramIterator

#import socket
import SocketServer
import sys
import math
import threading

#Posicion inicial de la camara (Hacia enfrente)
base.disableMouse()
base.camera.setPosHpr(0,0,1,0,0,0)

#valores iniciales
orden = "NADA"
ordAnterior = "NADA"
PORTNO = 666

#clase que maneja la recepción de ordenes
class handler(SocketServer.DatagramRequestHandler):
    def handle(self):
        global orden
        orden = self.rfile.readline().rstrip()

class CBosque(DirectObject.DirectObject):
    def __init__(self,px,py):
        #Declaracion variables de avance y giro
        self.movX = 0
        self.movY = 0
        #Distancia a desplazarse
        self.paso = 5
        #self.banderaPaso = 1
        self.tiempoAnterior = 0
```

```
        #Orientacion de la camara y su torque
        self.oriCam = 0
        self.velGiroCam = 10
        #Movimiento de la camara y su velocidad
        self.posCam = Vec3(0,0,1)
        self.velCam = Vec3(0,0,0)
        #Eventos (pruebas)
        self.accept('arrow_up',self.UP)
        self.accept("arrow_up-up", self.UPU)
        self.accept('arrow_down',self.DOWN)
        self.accept("arrow_down-up", self.DOWNU)
        self.accept('arrow_left',self.LEFT)
        self.accept("arrow_left-up", self.LEFTU)
        self.accept('arrow_right',self.RIGHT)
        self.accept("arrow_right-up", self.RIGHTU)
        #Variables basicas del mundo
        self.title = OnscreenText(text="Ambiente Boscoso",style=1, fg=(1,1,1,1),pos=(-1,-0.9), scale = 0.07)
        base.setBackgroundColor(1,1,1)
        #Niebla
        nube = Fog("niebla")
        nube.setColor(1,1,1)
        nube.setExpDensity(0.1)
        render.setFog(nube)
        #Iluminacion
        render.setLightOff()
        dlight = DirectionalLight('dlight')
        dlight.setColor(VBase4(0.2, 1, 0.45, 1))
        dlnp = render.attachNewNode(dlight)
        dlnp.setHpr(0, -60, 0)
        render.setLight(dlnp)
        ambientLight = AmbientLight('ambientLight')
        ambientLight.setColor(Vec4(1, 1, 0.5, 0.5))
        ambientLightNP = render.attachNewNode(ambientLight)
        render.setLight(ambientLightNP)
        #Elementos del mundo (las líneas comentadas son ampliaciones del mundo)
        self.cargaElementos(px,py)
#       self.cargaElementos(px+59.4,py)
#       self.cargaElementos(px-59.4,py)
#       self.cargaElementos(px,py+59.4)
#       self.cargaElementos(px+59.4,py+59.4)
#       self.cargaElementos(px-59.4,py+59.4)
#       self.cargaElementos(px,py-59.4)
```

```
        #   self.cargaElementos(px+59.4,py-59.4)
        #   self.cargaElementos(px-59.4,py-59.4)

            #Numero de Puerto 9990
            self.num_port = 9990
            #Direccion donde se mandara informacion (Clientes)
            self.target = NetAddress()
            self.target.setHost('148.204.67.222',self.num_port)
            self.target2 = NetAddress()
            self.target2.setHost('148.204.67.146',self.num_port)
            #Maneja la Conexion
            self.cManager = ConnectionManager()
            #Sera el encargado de mandar los datagramas
            self.cWriter = ConnectionWriter(self.cManager,0)
            #Leera los datagramas entrantes
            self.cReader = QueuedConnectionReader(self.cManager,0)

            #Abre la Conexion
            self.UDPsocket = self.cManager.openUDPConnection(self.num_port)
            self.cReader.addConnection(self.UDPsocket)

            #Hilo encargado de recibir mensajes del ControlUSB
            t = threading.Thread(target=self.leeOrdenes, args=())
            t.start()

        def leeOrdenes(self):
            self.s = SocketServer.UDPServer(('',PORTNO), handler)
            taskMgr.add(self.movVC,'Movimiento')
            self.s.serve_forever()

    #dependiendo de la orden, se cambian variables y se retransmite la informacion a los clientes
        def movVC(self,task):
            global orden,ordAnterior
            control=orden
            if ordAnterior!=orden:
                if control == "UP":     #Adelante
                    self.movX = 0
                    self.movY = 1
                    self.escribe(1,"UP")
                elif control == "DOWN":   #Atras
                    self.movX = 0
                    self.movY = -1
                    self.escribe(1,"DOWN")
```

```
            elif control == "LEFT":   #Izquierda
                self.movX = 1
                self.movY = 0
                self.escribe(1,"LEFT")
            elif control == "RIGHT":   #Derecha
                self.movX = -1
                self.movY = 0
                self.escribe(1,"RIGHT")
            elif control == "ULEFT":   #Adelante-Izquierda
                self.movY = 1
                self.movX = 1
                self.escribe(1,"ULEFT")
            elif control == "URIGHT":   #Adelante-Derecha
                self.movY = 1
                self.movX = -1
                self.escribe(1,"URIGHT")
            elif control == "DLEFT":   #Atras-Izquierda
                self.movY = -1
                self.movX = 1
                self.escribe(1,"DLEFT")
            elif control == "DRIGHT":   #Atras-Derecha
                self.movY = -1
                self.movX = -1
                self.escribe(1,"DRIGHT")
            elif control == "SELECT":   #Seleccionar
                print "Seleccion"
                self.escribe(1,"SEL")
            elif control == "OK":   #Listo
                print "El Control esta Conectado"
                taskMgr.add(self.lee,'Lee')
                taskMgr.add(self.actualiza, 'Actualiza')
            else:                   #Nada
                self.movY = 0
                self.movX = 0
                self.escribe(1,"NADA")
            ordAnterior=control
        return task.cont

    #se encarga de tratar la información y como responder
    def tratarInfo(self):
        myIterator = PyDatagramIterator(self.datagram)
        msgID = myIterator.getUint8()
```

```
        if msgID == 0:
            messageToPrint = myIterator.getString()
            print messageToPrint
            self.escribe(0,"148.204.67.46")

    #lee los datos
    def lee(self,task):
        if self.cReader.dataAvailable():
            self.datagram=NetDatagram()
            if self.cReader.getData(self.datagram):
                self.tratarInfo()
        return task.cont

    #manda datos a los clientes
    def escribe(self,n,evento):
        myPyDatagram=NetDatagram()
        myPyDatagram.addUint8(n)
        myPyDatagram.addString(evento)
        self.cWriter.send(myPyDatagram,self.UDPsocket,self.target)
        self.cWriter.send(myPyDatagram,self.UDPsocket,self.target2)

    def free(self):
        #Termina las Conexiones
        self.cManager.closeConnection(self.UDPsocket)

    #se modifican los valores de posicion y orientacion constantemente dependiendo del tiempo transcurrido
    def actualiza(self,Task):

        tiempoAct = Task.time - self.tiempoAnterior
        self.oriCam = self.oriCam + (self.velGiroCam * self.movX * tiempoAct)
        dX = math.sin(math.radians(self.oriCam))*self.paso
        dX = math.sqrt(dX*dX)
        dY = math.cos(math.radians(self.oriCam))*self.paso
        dY = math.sqrt(dY*dY)
        if self.oriCam % 360 <= 180:
            dX = -dX
        if self.oriCam % 360 >= 90 and self.oriCam % 360 <= 270:
            dY = -dY
        self.posCam = self.posCam + (Vec3(dX,dY,0) * self.movY * tiempoAct)
            #self.comunicarPos()
        base.camera.setPos(self.posCam)
        base.camera.setH(self.oriCam)
        self.tiempoAnterior = Task.time
        return Task.cont
```

```
#controles para prueba
def UP(self):
        self.movY = 1
        self.escribe(1,"UP")

    def UPU(self):
        self.movY = 0
        self.escribe(1,"SUP")

    def DOWN(self):
        self.movY = -1
        self.escribe(1,"DOWN")

    def DOWNU(self):
        self.movY = 0
        self.escribe(1,"SDOWN")

    def LEFT(self):
        self.movX = 1
        self.escribe(1,"LEFT")

    def LEFTU(self):
        self.movX = 0
        self.escribe(1,"SLEFT")

    def RIGHT(self):
        self.movX = -1
        self.escribe(1,"RIGHT")

    def RIGHTU(self):
        self.movX = 0
        self.escribe(1,"SRIGHT")

#carga el mundo virtual
    def cargaElementos(self,px,py):
        self.cargaSuelo(px,py)
        self.cargaRocas(px,py)
        self.cargaVegetacion(px,py)

    def cargaVegetacion(self,px,py):
        self.cargaArboles1(px,py)
        self.cargaArboles2(px,py)
        self.cargaArboles12(px,py)
        self.cargaArboles22(px,py)
```

```
        self.cargaArboles3(px,py)
        self.cargaTroncos(px,py)
        self.cargaArbustos2(px,py)
        self.cargaArbustos1(px,py)
        self.cargaArbustos12(px,py)

    #Arboles

    def cargaArboles1(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbol1")
            i.reparentTo(render)
            i.setScale(2, 2, 2.5)
            listaArb.append(i)
        listaArb[0].setPos(x,y+28,-0.8)
        listaArb[0].setHpr(24,23,0)
        listaArb[1].setPos(x-5,y+25,-0.8)
        listaArb[1].setHpr(10,10,0)
        listaArb[2].setPos(x+30,y+20,-0.8)
        listaArb[2].setHpr(40,-12,0)
        listaArb[3].setPos(x+20,y+10,0)
        listaArb[3].setHpr(56,0,0)
        listaArb[4].setPos(x-30,y+10,0)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x-10,y,-0.8)
        listaArb[5].setHpr(7,10,0)
        listaArb[6].setPos(x-20,y-10,0)
        listaArb[6].setHpr(15,0,0)
        listaArb[7].setPos(x+28,y-15,-0.8)
        listaArb[7].setHpr(20,20,0)
        listaArb[8].setPos(x-22,y-30,-0.8)
        listaArb[8].setHpr(50,-17,0)
        listaArb[9].setPos(x+15,y-27,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x,y+20,-0.8)
        listaArb[10].setHpr(45,-25,0)
        listaArb[11].setPos(x-16,y-12,0)
        listaArb[11].setHpr(88,0,0)

    def cargaArboles2(self,x,y):
        listaArb = []
```

```
        for i in range(12):
            i = loader.loadModel("arbol2")
            i.reparentTo(render)
            i.setScale(1.5, 1.5, 2.5)
            listaArb.append(i)
        listaArb[0].setPos(x-25,y-20,-0.8)
        listaArb[0].setHpr(24,5,0)
        listaArb[1].setPos(x-10,y-25,0)
        listaArb[1].setHpr(10,0,0)
        listaArb[2].setPos(x+28,y-22,-1)
        listaArb[2].setHpr(40,4,0)
        listaArb[3].setPos(x+12,y-16,-0.5)
        listaArb[3].setHpr(56,0,0)
        listaArb[4].setPos(x-22,y-1,-0.8)
        listaArb[4].setHpr(87,-8,0)
        listaArb[5].setPos(x+10,y,-0.8)
        listaArb[5].setHpr(7,-12,0)
        listaArb[6].setPos(x-5,y+5,-0.5)
        listaArb[6].setHpr(15,0,0)
        listaArb[7].setPos(x-20,y+20,-0.5)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x+20,y+25,-0.5)
        listaArb[8].setHpr(50,13,0)
        listaArb[9].setPos(x+27,y+14,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x+27,y+27,-0.8)
        listaArb[10].setHpr(55,15,0)
        listaArb[11].setPos(x-23,y+30,-0.5)
        listaArb[11].setHpr(19,0,0)

    def cargaArboles12(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbol1")
            i.reparentTo(render)
            i.setScale(2, 2, 5)
            listaArb.append(i)
        listaArb[0].setPos(x-25,y+27,0)
        listaArb[0].setHpr(24,0,0)
        listaArb[1].setPos(x-7,y+29,0)
        listaArb[1].setHpr(10,0,0)
        listaArb[2].setPos(x-1,y+27,0)
```

```
        listaArb[2].setHpr(40,0,0)
        listaArb[3].setPos(x+7,y+17,0)
        listaArb[3].setHpr(56,0,0)
        listaArb[4].setPos(x+15,y+17,0)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x-7,y+12,0)
        listaArb[5].setHpr(7,0,0)
        listaArb[6].setPos(x-28,y+7,0)
        listaArb[6].setHpr(15,0,0)
        listaArb[7].setPos(x+23,y+2,0)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x+25,y-10,0)
        listaArb[8].setHpr(50,0,0)
        listaArb[9].setPos(x-24,y-14,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x-18,y-16,0)
        listaArb[10].setHpr(45,0,0)
        listaArb[11].setPos(x+10,y-20,0)
        listaArb[11].setHpr(88,0,0)

    def cargaArboles22(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbol2")
            i.reparentTo(render)
            i.setScale(2, 2, 4)
            listaArb.append(i)
        listaArb[0].setPos(x+22,y+30,0)
        listaArb[0].setHpr(24,0,0)
        listaArb[1].setPos(x+12,y+27,0)
        listaArb[1].setHpr(10,0,0)
        listaArb[2].setPos(x-18,y+15,0)
        listaArb[2].setHpr(40,0,0)
        listaArb[3].setPos(x+11,y+14,0)
        listaArb[3].setHpr(56,0,0)
        listaArb[4].setPos(x,y+7,0)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x+15,y+2,0)
        listaArb[5].setHpr(7,0,0)
        listaArb[6].setPos(x-20,y,0)
        listaArb[6].setHpr(15,0,0)
        listaArb[7].setPos(x-3,y-17,0)
        listaArb[7].setHpr(20,0,0)
```

```
        listaArb[8].setPos(x-30,y-17,0)
        listaArb[8].setHpr(50,0,0)
        listaArb[9].setPos(x+25,y-19,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x+3,y-24,0)
        listaArb[10].setHpr(55,0,0)
        listaArb[11].setPos(x-15,y-27,0)
        listaArb[11].setHpr(19,0,0)

    def cargaArboles3(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbol3")
            i.reparentTo(render)
            i.setScale(1.5, 1.5, 3)
            listaArb.append(i)
        listaArb[0].setPos(x+30,y,0)
        listaArb[0].setHpr(24,0,0)
        listaArb[1].setPos(x+25,y+7,0)
        listaArb[1].setHpr(10,0,0)
        listaArb[2].setPos(x+22,y-8,0)
        listaArb[2].setHpr(40,0,0)
        listaArb[3].setPos(x+17,y+30,0)
        listaArb[3].setHpr(56,0,0)
        listaArb[4].setPos(x+10,y+25,0)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x+7,y-22,0)
        listaArb[5].setHpr(7,0,0)
        listaArb[6].setPos(x+1,y-27,0)
        listaArb[6].setHpr(15,0,0)
        listaArb[7].setPos(x-15,y-20,0)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x-14,y+30,0)
        listaArb[8].setHpr(50,0,0)
        listaArb[9].setPos(x-25,y-5,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x-24,y+13,0)
        listaArb[10].setHpr(123,0,0)
        listaArb[11].setPos(x-28,y+23,0)
        listaArb[11].setHpr(11,0,0)
```

```
def cargaTroncos(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("tronco")
            i.reparentTo(render)
            i.setScale(1.25, 1.25, 3)
            listaArb.append(i)
        listaArb[0].setPos(x+2,y+30,0)
        listaArb[0].setHpr(24,5,0)
        listaArb[1].setPos(x-10,y+27,0)
        listaArb[1].setHpr(10,2,0)
        listaArb[2].setPos(x+25,y+20,0)
        listaArb[2].setHpr(40,3,0)
        listaArb[3].setPos(x-20,y+10,0)
        listaArb[3].setHpr(56,10,0)
        listaArb[4].setPos(x-24,y+5,0)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x+21,y+6,0)
        listaArb[5].setHpr(7,4,0)
        listaArb[6].setPos(x-8,y-2,0)
        listaArb[6].setHpr(15,2,0)
        listaArb[7].setPos(x+20,y-12,0)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x-13,y-16,0)
        listaArb[8].setHpr(50,1,0)
        listaArb[9].setPos(x+15,y-20,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x-5,y-27,0)
        listaArb[10].setHpr(123,2,0)
        listaArb[11].setPos(x-10,y-30,0)
        listaArb[11].setHpr(11,4,0)

    #Rocas

    def cargaRocas(self,x,y):
        listaRoc = []
        for i in range(12):
            i = loader.loadModel("roca1")
            i.reparentTo(render)
            i.setScale(0.5, 0.5, 0.5)
            listaRoc.append(i)
        listaRoc[0].setPos(x-30,y+20,-0.5)
        listaRoc[0].setHpr(24,5,0)
        listaRoc[1].setPos(x-17,y+25,-0.5)
```

```
        listaRoc[1].setHpr(10,2,0)
        listaRoc[2].setPos(x+3,y+23,-0.5)
        listaRoc[2].setHpr(40,3,0)
        listaRoc[3].setPos(x+14,y+22,-0.5)
        listaRoc[3].setHpr(56,10,0)
        listaRoc[4].setPos(x+28,y+23,-0.5)
        listaRoc[4].setHpr(87,0,0)
        listaRoc[5].setPos(x+17,y+7,-0.5)
        listaRoc[5].setHpr(7,4,0)
        listaRoc[6].setPos(x-5,y,-0.5)
        listaRoc[6].setHpr(15,2,0)
        listaRoc[7].setPos(x+27,y-5,-0.5)
        listaRoc[7].setHpr(20,0,0)
        listaRoc[8].setPos(x-25,y-10,-0.5)
        listaRoc[8].setHpr(50,1,0)
        listaRoc[9].setPos(x+16,y-16,-0.5)
        listaRoc[9].setHpr(93,0,0)
        listaRoc[10].setPos(x+5,y-30,-0.5)
        listaRoc[10].setHpr(123,2,0)
        listaRoc[11].setPos(x+30,y-30,-0.5)
        listaRoc[11].setHpr(11,4,0)

    #Arbustos

    def cargaArbustos1(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbusto1")
            i.reparentTo(render)
            i.setScale(1, 1, 1.3)
            listaArb.append(i)
        listaArb[0].setPos(x-25,y-25,-0.3)
        listaArb[0].setHpr(24,5,0)
        listaArb[1].setPos(x-2,y-25,-0.3)
        listaArb[1].setHpr(10,2,0)
        listaArb[2].setPos(x+25,y-25,-0.3)
        listaArb[2].setHpr(40,3,0)
        listaArb[3].setPos(x+14,y-13,-0.3)
        listaArb[3].setHpr(56,10,0)
        listaArb[4].setPos(x-5,y-5,-0.3)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x+12,y-2,-0.3)
        listaArb[5].setHpr(7,4,0)
```

```
        listaArb[6].setPos(x-30,y,-0.3)
        listaArb[6].setHpr(15,2,0)
        listaArb[7].setPos(x-2,y+5,-0.3)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x-20,y+5,-0.3)
        listaArb[8].setHpr(50,1,0)
        listaArb[9].setPos(x+30,y+10,-0.3)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x+10,y+20,-0.3)
        listaArb[10].setHpr(123,2,0)
        listaArb[11].setPos(x-23,y+18,-0.3)
        listaArb[11].setHpr(11,4,0)

    def cargaArbustos12(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbusto1")
            i.reparentTo(render)
            i.setScale(2, 2, 2)
            listaArb.append(i)
        listaArb[0].setPos(x-30,y+27,-0.3)
        listaArb[0].setHpr(24,5,0)
        listaArb[1].setPos(x-18,y+30,-0.3)
        listaArb[1].setHpr(10,2,0)
        listaArb[2].setPos(x+6,y+25,-0.3)
        listaArb[2].setHpr(40,3,0)
        listaArb[3].setPos(x+29,y+17,-0.3)
        listaArb[3].setHpr(56,10,0)
        listaArb[4].setPos(x+7,y+14,-0.3)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x-18,y+8,-0.3)
        listaArb[5].setHpr(7,4,0)
        listaArb[6].setPos(x-2,y-2,-0.3)
        listaArb[6].setHpr(15,2,0)
        listaArb[7].setPos(x+15,y-3,-0.3)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x-28,y-3,-0.3)
        listaArb[8].setHpr(50,1,0)
        listaArb[9].setPos(x-12,y-13,-0.3)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x-23,y-21,-0.3)
        listaArb[10].setHpr(123,2,0)
        listaArb[11].setPos(x+22,y-21,-0.3)
        listaArb[11].setHpr(11,4,0)
```

```
def cargaArbustos2(self,x,y):
    listaArb = []
    for i in range(12):
        i = loader.loadModel("arbusto2")
        i.reparentTo(render)
        i.setScale(1.5, 1.5, 1.5)
        listaArb.append(i)
    listaArb[0].setPos(x+20,y-30,-0.3)
    listaArb[0].setHpr(24,5,0)
    listaArb[1].setPos(x+10,y-25,-0.3)
    listaArb[1].setHpr(10,2,0)
    listaArb[2].setPos(x-12,y-22,-0.3)
    listaArb[2].setHpr(40,3,0)
    listaArb[3].setPos(x,y-15,-0.3)
    listaArb[3].setHpr(56,10,0)
    listaArb[4].setPos(x-20,y-5,-0.3)
    listaArb[4].setHpr(87,0,0)
    listaArb[5].setPos(x+11,y+3,-0.3)
    listaArb[5].setHpr(7,4,0)
    listaArb[6].setPos(x+27,y+3,-0.3)
    listaArb[6].setHpr(15,2,0)
    listaArb[7].setPos(x-9,y+4,-0.3)
    listaArb[7].setHpr(20,0,0)
    listaArb[8].setPos(x-5,y+15,-0.3)
    listaArb[8].setHpr(50,1,0)
    listaArb[9].setPos(x+20,y+20,-0.3)
    listaArb[9].setHpr(93,0,0)
    listaArb[10].setPos(x-13,y+24,-0.3)
    listaArb[10].setHpr(123,2,0)
    listaArb[11].setPos(x-23,y+23,-0.3)
    listaArb[11].setHpr(11,4,0)

#Suelo del Bosque (59.4 x 59.4 mts)

def cargaSuelo(self,x,y):
    distBqs = 19.8
    listaSuelo = []
    for i in range(9):
        i = loader.loadModel("suelo")
        i.reparentTo(render)
        listaSuelo.append(i)
    listaSuelo[0].setPos(x-distBqs, y-distBqs, -0.5)
    listaSuelo[1].setPos(x, y-distBqs, -0.5)
```

```
        listaSuelo[2].setPos(x+distBqs, y-distBqs, -0.5)
        listaSuelo[3].setPos(x-distBqs, y, -0.5)
        listaSuelo[4].setPos(x, y, -0.5)
        listaSuelo[5].setPos(x+distBqs, y, -0.5)
        listaSuelo[6].setPos(x-distBqs, y+distBqs, -0.5)
        listaSuelo[7].setPos(x, y+distBqs, -0.5)
        listaSuelo[8].setPos(x+distBqs, y+distBqs, -0.5)

bosque = CBosque(0,0)

run()
```

Código de despliegue del ambiente virtual e interacción de controles (Cliente)

```
import direct.directbase.DirectStart
from pandac.PandaModules import *
from direct.gui.DirectGui import *
from direct.gui.OnscreenImage import OnscreenImage
from direct.showbase import DirectObject
from direct.task import Task
from pandac.PandaModules import ConnectionManager
from pandac.PandaModules import QueuedConnectionReader
from pandac.PandaModules import ConnectionWriter
from direct.distributed.PyDatagram import PyDatagram
from direct.distributed.PyDatagramIterator import PyDatagramIterator

import socket
import sys
import math

#posicion y orientacion de camara y desabilitacion del raton
base.camera.setPosHpr(0,0,1,45,0,0)
base.disableMouse()

class CBosque(DirectObject.DirectObject):
    def __init__(self,px,py):
        self.movX = 0
        self.movY = 0
        self.paso = 5
        self.tiempoAnterior = 0
        self.oriCam = 45                #Ángulo Inicial (Izq)
        self.oriCamOrig = 0
        self.posCam = Vec3(0,0,1)
        self.velCam = Vec3(0,0,0)
        self.velGiroCam = 10

        self.title = OnscreenText(text="Ambiente Boscoso",style=1, fg=(1,1,1,1),pos=(-1,-0.9), scale = 0.07)
        base.setBackgroundColor(1,1,1)

        #Niebla

        nube = Fog("niebla")
        nube.setColor(1,1,1)
        nube.setExpDensity(0.1)
        render.setFog(nube)
```

```
        #Iluminacion

        render.setLightOff()
        dlight = DirectionalLight('dlight')
        dlight.setColor(VBase4(0.2, 1, 0.45, 1))
        dlnp = render.attachNewNode(dlight)
        dlnp.setHpr(0, -60, 0)
        render.setLight(dlnp)
        ambientLight = AmbientLight('ambientLight')
        ambientLight.setColor(Vec4(1, 1, 0.5, 0.5))
        ambientLightNP = render.attachNewNode(ambientLight)
        render.setLight(ambientLightNP)

    #carga el escenario
        self.cargaElementos(px,py)
      #   self.cargaElementos(px+59.4,py)
      #   self.cargaElementos(px-59.4,py)
      #   self.cargaElementos(px,py+59.4)
      #   self.cargaElementos(px+59.4,py+59.4)
      #   self.cargaElementos(px-59.4,py+59.4)
      #   self.cargaElementos(px,py-59.4)
      #   self.cargaElementos(px+59.4,py-59.4)
      #   self.cargaElementos(px-59.4,py-59.4)

        #Numero de Puerto 9990
        self.num_port = 9990
        self.target = NetAddress()
        self.target.setHost('148.204.67.46',self.num_port)

        #Maneja la Conexion

        self.cManager = ConnectionManager()

        #Sera el encargado de mandar los datagramas

        self.cWriter = ConnectionWriter(self.cManager,0)

        #Leera los datagramas entrantes

        self.cReader = QueuedConnectionReader(self.cManager, 0)

        #Abre la Conexion

        self.UDPsocket = self.cManager.openUDPConnection(self.num_port)
```

```
        self.cReader.addConnection(self.UDPsocket)
        self.escribe(0,"Cliente Izq Listo!")

        taskMgr.add(self.lee,"Lee")
        taskMgr.add(self.actualiza,"Actualiza")

    #funcion que actualiza posición y orientación de los clientes
    def actualiza(self,Task):
        tiempoAct=Task.time - self.tiempoAnterior
        self.oriCam = self.oriCam + (self.velGiroCam * self.movX * tiempoAct)
        self.oriCamOrig = self.oriCamOrig + (self.velGiroCam * self.movX * tiempoAct)
        dX = math.sin(math.radians(self.oriCamOrig))*self.paso
        dX = math.sqrt(dX*dX)
        dY = math.cos(math.radians(self.oriCamOrig))*self.paso
        dY = math.sqrt(dY*dY)
        if self.oriCamOrig % 360 <= 180:
            dX = -dX
        if self.oriCamOrig % 360 >= 90 and self.oriCamOrig % 360 <= 270:
            dY = -dY
        self.posCam = self.posCam + (Vec3(dX,dY,0) * self.movY * tiempoAct)
        base.camera.setPos(self.posCam)
        base.camera.setH(self.oriCam)
        self.tiempoAnterior = Task.time
        return Task.cont

    #Se encarga del manejo de los datos recibidos
    def tratarInfo(self):
        myIterator = PyDatagramIterator(self.datagram)
        msgID = myIterator.getUint8()
        if msgID == 0:
            messageToPrint = "Bienvenido,",myIterator.getString()
            print messageToPrint
        if msgID == 1:
            mensaje = myIterator.getString()
            if mensaje == "UP":
                self.movX = 0
                self.movY = 1
            elif mensaje == "DOWN":
                self.movX = 0
                self.movY = -1
            elif mensaje == "LEFT":
                self.movX = 1
                self.movY = 0
```

```
        elif mensaje == "RIGHT":
            self.movX = -1
            self.movY = 0
        elif mensaje == "ULEFT":
            self.movX = 1
            self.movY = 1
        elif mensaje == "URIGHT":
            self.movX = -1
            self.movY = 1
        elif mensaje == "DLEFT":
            self.movX = 1
            self.movY = -1
        elif mensaje == "DRIGHT":
            self.movX = -1
            self.movY = -1
        elif mensaje == "SEL":
            print "Seleccion"
        else:
            self.movX = 0
            self.movY = 0

def lee(self,task):
    if self.cReader.dataAvailable():
        self.datagram=NetDatagram()
        if self.cReader.getData(self.datagram):
            self.tratarInfo()
    return task.cont

def escribe(self,n,evento):
    myPyDatagram=NetDatagram()
    myPyDatagram.addUint8(n)
    myPyDatagram.addString(evento)
    self.cWriter.send(myPyDatagram,self.UDPsocket,self.target)

def free(self):
    #Termina las Conexiones
    self.cManager.closeConnection(self.UDPsocket)

def cargaElementos(self,px,py):
    self.cargaSuelo(px,py)
    self.cargaRocas(px,py)
    self.cargaVegetacion(px,py)
```

```
def cargaVegetacion(self,px,py):
        self.cargaArboles1(px,py)
        self.cargaArboles2(px,py)
        self.cargaArboles12(px,py)
        self.cargaArboles22(px,py)
        self.cargaArboles3(px,py)
        self.cargaTroncos(px,py)
        self.cargaArbustos2(px,py)
        self.cargaArbustos1(px,py)
        self.cargaArbustos12(px,py)

    #Arboles

    def cargaArboles1(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbol1")
            i.reparentTo(render)
            i.setScale(2, 2, 2.5)
            listaArb.append(i)
        listaArb[0].setPos(x,y+28,-0.8)
        listaArb[0].setHpr(24,23,0)
        listaArb[1].setPos(x-5,y+25,-0.8)
        listaArb[1].setHpr(10,10,0)
        listaArb[2].setPos(x+30,y+20,-0.8)
        listaArb[2].setHpr(40,-12,0)
        listaArb[3].setPos(x+20,y+10,0)
        listaArb[3].setHpr(56,0,0)
        listaArb[4].setPos(x-30,y+10,0)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x-10,y,-0.8)
        listaArb[5].setHpr(7,10,0)
        listaArb[6].setPos(x-20,y-10,0)
        listaArb[6].setHpr(15,0,0)
        listaArb[7].setPos(x+28,y-15,-0.8)
        listaArb[7].setHpr(20,20,0)
        listaArb[8].setPos(x-22,y-30,-0.8)
        listaArb[8].setHpr(50,-17,0)
        listaArb[9].setPos(x+15,y-27,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x,y+20,-0.8)
        listaArb[10].setHpr(45,-25,0)
        listaArb[11].setPos(x-16,y-12,0)
        listaArb[11].setHpr(88,0,0)
```

```
def cargaArboles2(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbol2")
            i.reparentTo(render)
            i.setScale(1.5, 1.5, 2.5)
            listaArb.append(i)
        listaArb[0].setPos(x-25,y-20,-0.8)
        listaArb[0].setHpr(24,5,0)
        listaArb[1].setPos(x-10,y-25,0)
        listaArb[1].setHpr(10,0,0)
        listaArb[2].setPos(x+28,y-22,-1)
        listaArb[2].setHpr(40,4,0)
        listaArb[3].setPos(x+12,y-16,-0.5)
        listaArb[3].setHpr(56,0,0)
        listaArb[4].setPos(x-22,y-1,-0.8)
        listaArb[4].setHpr(87,-8,0)
        listaArb[5].setPos(x+10,y,-0.8)
        listaArb[5].setHpr(7,-12,0)
        listaArb[6].setPos(x-5,y+5,-0.5)
        listaArb[6].setHpr(15,0,0)
        listaArb[7].setPos(x-20,y+20,-0.5)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x+20,y+25,-0.5)
        listaArb[8].setHpr(50,13,0)
        listaArb[9].setPos(x+27,y+14,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x+27,y+27,-0.8)
        listaArb[10].setHpr(55,15,0)
        listaArb[11].setPos(x-23,y+30,-0.5)
        listaArb[11].setHpr(19,0,0)

    def cargaArboles12(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbol1")
            i.reparentTo(render)
            i.setScale(2, 2, 5)
            listaArb.append(i)
        listaArb[0].setPos(x-25,y+27,0)
        listaArb[0].setHpr(24,0,0)
        listaArb[1].setPos(x-7,y+29,0)
        listaArb[1].setHpr(10,0,0)
```

```
        listaArb[2].setPos(x-1,y+27,0)
        listaArb[2].setHpr(40,0,0)
        listaArb[3].setPos(x+7,y+17,0)
        listaArb[3].setHpr(56,0,0)
        listaArb[4].setPos(x+15,y+17,0)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x-7,y+12,0)
        listaArb[5].setHpr(7,0,0)
        listaArb[6].setPos(x-28,y+7,0)
        listaArb[6].setHpr(15,0,0)
        listaArb[7].setPos(x+23,y+2,0)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x+25,y-10,0)
        listaArb[8].setHpr(50,0,0)
        listaArb[9].setPos(x-24,y-14,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x-18,y-16,0)
        listaArb[10].setHpr(45,0,0)
        listaArb[11].setPos(x+10,y-20,0)
        listaArb[11].setHpr(88,0,0)

    def cargaArboles22(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbol2")
            i.reparentTo(render)
            i.setScale(2, 2, 4)
            listaArb.append(i)
        listaArb[0].setPos(x+22,y+30,0)
        listaArb[0].setHpr(24,0,0)
        listaArb[1].setPos(x+12,y+27,0)
        listaArb[1].setHpr(10,0,0)
        listaArb[2].setPos(x-18,y+15,0)
        listaArb[2].setHpr(40,0,0)
        listaArb[3].setPos(x+11,y+14,0)
        listaArb[3].setHpr(56,0,0)
        listaArb[4].setPos(x,y+7,0)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x+15,y+2,0)
        listaArb[5].setHpr(7,0,0)
        listaArb[6].setPos(x-20,y,0)
        listaArb[6].setHpr(15,0,0)
        listaArb[7].setPos(x-3,y-17,0)
        listaArb[7].setHpr(20,0,0)
```

```
        listaArb[8].setPos(x-30,y-17,0)
        listaArb[8].setHpr(50,0,0)
        listaArb[9].setPos(x+25,y-19,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x+3,y-24,0)
        listaArb[10].setHpr(55,0,0)
        listaArb[11].setPos(x-15,y-27,0)
        listaArb[11].setHpr(19,0,0)

    def cargaArboles3(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbol3")
            i.reparentTo(render)
            i.setScale(1.5, 1.5, 3)
            listaArb.append(i)
        listaArb[0].setPos(x+30,y,0)
        listaArb[0].setHpr(24,0,0)
        listaArb[1].setPos(x+25,y+7,0)
        listaArb[1].setHpr(10,0,0)
        listaArb[2].setPos(x+22,y-8,0)
        listaArb[2].setHpr(40,0,0)
        listaArb[3].setPos(x+17,y+30,0)
        listaArb[3].setHpr(56,0,0)
        listaArb[4].setPos(x+10,y+25,0)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x+7,y-22,0)
        listaArb[5].setHpr(7,0,0)
        listaArb[6].setPos(x+1,y-27,0)
        listaArb[6].setHpr(15,0,0)
        listaArb[7].setPos(x-15,y-20,0)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x-14,y+30,0)
        listaArb[8].setHpr(50,0,0)
        listaArb[9].setPos(x-25,y-5,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x-24,y+13,0)
        listaArb[10].setHpr(123,0,0)
        listaArb[11].setPos(x-28,y+23,0)
        listaArb[11].setHpr(11,0,0)
```

```
def cargaTroncos(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("tronco")
            i.reparentTo(render)
            i.setScale(1.25, 1.25, 3)
            listaArb.append(i)
        listaArb[0].setPos(x+2,y+30,0)
        listaArb[0].setHpr(24,5,0)
        listaArb[1].setPos(x-10,y+27,0)
        listaArb[1].setHpr(10,2,0)
        listaArb[2].setPos(x+25,y+20,0)
        listaArb[2].setHpr(40,3,0)
        listaArb[3].setPos(x-20,y+10,0)
        listaArb[3].setHpr(56,10,0)
        listaArb[4].setPos(x-24,y+5,0)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x+21,y+6,0)
        listaArb[5].setHpr(7,4,0)
        listaArb[6].setPos(x-8,y-2,0)
        listaArb[6].setHpr(15,2,0)
        listaArb[7].setPos(x+20,y-12,0)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x-13,y-16,0)
        listaArb[8].setHpr(50,1,0)
        listaArb[9].setPos(x+15,y-20,0)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x-5,y-27,0)
        listaArb[10].setHpr(123,2,0)
        listaArb[11].setPos(x-10,y-30,0)
        listaArb[11].setHpr(11,4,0)

    #Rocas

    def cargaRocas(self,x,y):
        listaRoc = []
        for i in range(12):
            i = loader.loadModel("roca1")
            i.reparentTo(render)
            i.setScale(0.5, 0.5, 0.5)
            listaRoc.append(i)
        listaRoc[0].setPos(x-30,y+20,-0.5)
        listaRoc[0].setHpr(24,5,0)
```

```
        listaRoc[1].setPos(x-17,y+25,-0.5)
        listaRoc[1].setHpr(10,2,0)
        listaRoc[2].setPos(x+3,y+23,-0.5)
        listaRoc[2].setHpr(40,3,0)
        listaRoc[3].setPos(x+14,y+22,-0.5)
        listaRoc[3].setHpr(56,10,0)
        listaRoc[4].setPos(x+28,y+23,-0.5)
        listaRoc[4].setHpr(87,0,0)
        listaRoc[5].setPos(x+17,y+7,-0.5)
        listaRoc[5].setHpr(7,4,0)
        listaRoc[6].setPos(x-5,y,-0.5)
        listaRoc[6].setHpr(15,2,0)
        listaRoc[7].setPos(x+27,y-5,-0.5)
        listaRoc[7].setHpr(20,0,0)
        listaRoc[8].setPos(x-25,y-10,-0.5)
        listaRoc[8].setHpr(50,1,0)
        listaRoc[9].setPos(x+16,y-16,-0.5)
        listaRoc[9].setHpr(93,0,0)
        listaRoc[10].setPos(x+5,y-30,-0.5)
        listaRoc[10].setHpr(123,2,0)
        listaRoc[11].setPos(x+30,y-30,-0.5)
        listaRoc[11].setHpr(11,4,0)

    #Arbustos

    def cargaArbustos1(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbusto1")
            i.reparentTo(render)
            i.setScale(1, 1, 1.3)
            listaArb.append(i)
        listaArb[0].setPos(x-25,y-25,-0.3)
        listaArb[0].setHpr(24,5,0)
        listaArb[1].setPos(x-2,y-25,-0.3)
        listaArb[1].setHpr(10,2,0)
        listaArb[2].setPos(x+25,y-25,-0.3)
        listaArb[2].setHpr(40,3,0)
        listaArb[3].setPos(x+14,y-13,-0.3)
        listaArb[3].setHpr(56,10,0)
        listaArb[4].setPos(x-5,y-5,-0.3)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x+12,y-2,-0.3)
```

```
        listaArb[5].setHpr(7,4,0)
        listaArb[6].setPos(x-30,y,-0.3)
        listaArb[6].setHpr(15,2,0)
        listaArb[7].setPos(x-2,y+5,-0.3)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x-20,y+5,-0.3)
        listaArb[8].setHpr(50,1,0)
        listaArb[9].setPos(x+30,y+10,-0.3)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x+10,y+20,-0.3)
        listaArb[10].setHpr(123,2,0)
        listaArb[11].setPos(x-23,y+18,-0.3)
        listaArb[11].setHpr(11,4,0)

    def cargaArbustos12(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbusto1")
            i.reparentTo(render)
            i.setScale(2, 2, 2)
            listaArb.append(i)
        listaArb[0].setPos(x-30,y+27,-0.3)
        listaArb[0].setHpr(24,5,0)
        listaArb[1].setPos(x-18,y+30,-0.3)
        listaArb[1].setHpr(10,2,0)
        listaArb[2].setPos(x+6,y+25,-0.3)
        listaArb[2].setHpr(40,3,0)
        listaArb[3].setPos(x+29,y+17,-0.3)
        listaArb[3].setHpr(56,10,0)
        listaArb[4].setPos(x+7,y+14,-0.3)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x-18,y+8,-0.3)
        listaArb[5].setHpr(7,4,0)
        listaArb[6].setPos(x-2,y-2,-0.3)
        listaArb[6].setHpr(15,2,0)
        listaArb[7].setPos(x+15,y-3,-0.3)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x-28,y-3,-0.3)
        listaArb[8].setHpr(50,1,0)
        listaArb[9].setPos(x-12,y-13,-0.3)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x-23,y-21,-0.3)
        listaArb[10].setHpr(123,2,0)
```

```
        listaArb[11].setPos(x+22,y-21,-0.3)
        listaArb[11].setHpr(11,4,0)

    def cargaArbustos2(self,x,y):
        listaArb = []
        for i in range(12):
            i = loader.loadModel("arbusto2")
            i.reparentTo(render)
            i.setScale(1.5, 1.5, 1.5)
            listaArb.append(i)
        listaArb[0].setPos(x+20,y-30,-0.3)
        listaArb[0].setHpr(24,5,0)
        listaArb[1].setPos(x+10,y-25,-0.3)
        listaArb[1].setHpr(10,2,0)
        listaArb[2].setPos(x-12,y-22,-0.3)
        listaArb[2].setHpr(40,3,0)
        listaArb[3].setPos(x,y-15,-0.3)
        listaArb[3].setHpr(56,10,0)
        listaArb[4].setPos(x-20,y-5,-0.3)
        listaArb[4].setHpr(87,0,0)
        listaArb[5].setPos(x+11,y+3,-0.3)
        listaArb[5].setHpr(7,4,0)
        listaArb[6].setPos(x+27,y+3,-0.3)
        listaArb[6].setHpr(15,2,0)
        listaArb[7].setPos(x-9,y+4,-0.3)
        listaArb[7].setHpr(20,0,0)
        listaArb[8].setPos(x-5,y+15,-0.3)
        listaArb[8].setHpr(50,1,0)
        listaArb[9].setPos(x+20,y+20,-0.3)
        listaArb[9].setHpr(93,0,0)
        listaArb[10].setPos(x-13,y+24,-0.3)
        listaArb[10].setHpr(123,2,0)
        listaArb[11].setPos(x-23,y+23,-0.3)
        listaArb[11].setHpr(11,4,0)

    #Suelo del Bosque (59.4 x 59.4 mts)

    def cargaSuelo(self,x,y):
        distBqs = 19.8
        listaSuelo = []
```

```
        for i in range(9):
            i = loader.loadModel("suelo")
            i.reparentTo(render)
            listaSuelo.append(i)
        listaSuelo[0].setPos(x-distBqs, y-distBqs, -0.5)
        listaSuelo[1].setPos(x, y-distBqs, -0.5)
        listaSuelo[2].setPos(x+distBqs, y-distBqs, -0.5)
        listaSuelo[3].setPos(x-distBqs, y, -0.5)
        listaSuelo[4].setPos(x, y, -0.5)
        listaSuelo[5].setPos(x+distBqs, y, -0.5)
        listaSuelo[6].setPos(x-distBqs, y+distBqs, -0.5)
        listaSuelo[7].setPos(x, y+distBqs, -0.5)
        listaSuelo[8].setPos(x+distBqs, y+distBqs, -0.5)

bosque = CBosque(0,0)

run()
```