UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS

FACULTAD 5
ENTORNOS VIRTUALES

"Interfaz para el Manejo de los Dispositivos de Entrada y Salida en Sistemas de Realidad Virtual."

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias en Informática

Autor: Yisel Nerys Cedeño Remón

Tutores: Msc. José Ignacio Guzmán Montoto

Ing. Yanoski Rogelio Camacho

Ciudad de la Habana

Junio, 2007

DECLARACIÓN DE AUTORÍA

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

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

_____	_____
Yisel Nerys Cedeño Remón	Yanoski Camacho Román
Firma del Autor	Firma del Tutor

DATOS DE CONTACTO

Nombre y Apellidos: Yanoski Rogelio Camacho Román
Edad: 26 años
Ciudadanía: cubano
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Informática
Categoría Docente: Profesor Instructor
E-mail: rcamacho@uci.cu

Graduado de la CUJAE, con tres años de experiencia en el tema de la Gráfica Computacional, y líder de un proyecto de Realidad Virtual en la Universidad de las Ciencias Informáticas.

Dedicatoria

A mis padres,

A mi familia,

A Adrián Carlos,

Al Comandante.

Agradecimientos

Agradezco a todas las personas que me ayudaron a la creación de este trabajo, a los tutores, al Tte Cor. Ing José E. Villar. A mis padres que me han apoyado en todo, a Adrián que a pesar de estar lejos siempre estuvo presente cuando lo necesité, a mis compañeros y amigos que me han apoyado y me han ayudado a comprender el tema de la Realidad Virtual.

Resumen

En los Sistemas de Realidad Virtual (SRV) el sistema de adquisición de datos cumple la tarea de transmitir la información del mundo real al mundo virtual. En el caso de los simuladores el uso de los sensores juega un papel importante por dar la posibilidad al usuario de sumergirse en el mundo virtual, lo que eleva la calidad del entrenamiento

La necesidad de este trabajo surge al introducirse el empleo de las tarjetas inteligentes para la adquisición de datos provenientes de diferentes tipos de sensores, La empresa Simuladores Profesionales (SIMPRO), al tener esta problemática requiere del desarrollo de un software para la atención a estas tarjetas y ofrecer respuestas a sistemas de tiempo real. La interfaz debe ser capaz de proporcionar los datos necesarios para realizar los cálculos pertinentes y luego ser visualizados en el simulador.

Este proyecto aborda el desarrollo de una interfaz de software para lograr la comunicación entre los sensores y la aplicación haciendo uso de la tarjeta inteligente Flash Lite. Para el desarrollo de esta interfaz se realizó un estudio de cómo otras herramientas de desarrollo manejan el tema de los sensores. Se investiga el uso de los sensores en el simulador y finalmente se propone una solución al problema planteado.

Contenido

Introducción

En la actualidad, La Realidad Virtual se plasma en una multiplicidad de sistemas, profesionales de otros campos, como la medicina, economía y exploración espacial, utilizan los laboratorios virtuales para el entrenamiento y la planificación en una gran variedad de funciones. Los cirujanos pueden realizar operaciones simuladas para ensayar las técnicas más complicadas, antes de una operación real. Los economistas exploran un modelo de acción de un sistema económico para poder entender mejor las complejas relaciones existentes entre sus distintos componentes. Los astronautas tienen la posibilidad de volar sobre la superficie simulada de un planeta desconocido y experimentar la sensación que tendrían si estuvieran allí. Los arquitectos pueden hacer que sus clientes, enfundados en cascos y guantes, visiten los pisos-piloto en un mundo de Realidad Virtual, dándoles la oportunidad de que abran las puertas o las ventanas y enciendan o apaguen las luces del apartamento. Por otra parte, permite la anticipación de errores de diseño y experiencias físicas con ambientes no construidos. En el ámbito científico, investigadores estudian moléculas complejas, desplazando grupos de átomos mediante un instrumento.

Un componente muy importante de todo simulador es el electromecánico, o sea, los dispositivos de entrada/salida que permiten la interacción del usuario con el sistema logrando mayor inmersión y dándole el realismo requerido a los simuladores.

En La Universidad de las Ciencias Informáticas (UCI) existe un grupo de trabajo encargado del desarrollo de proyectos de realidad virtual que trabaja con profesionales de la empresa SIMPRO. Entre estos proyectos se encuentra el desarrollo conjunto de simuladores para el entrenamiento del personal. Estos equipos de tecnología de avanzada cuentan entre sus sistemas con el sistema de adquisición de datos, encargado de adquirir las acciones realizadas por el usuario y traducirlas a una interacción con el entorno virtual.

El sistema de adquisición de datos en estos medios de entrenamiento está compuesto generalmente por un grupo de sensores y un hardware encargado de la adquisición y transformación conveniente de estos datos a los necesarios para el funcionamiento adecuado del resto de los sistemas.

Después del análisis del sistema por parte del equipo de electrónicos, aparece la necesidad del desarrollo de una interfaz de software que permita la comunicación de la computadora con una tarjeta inteligente de adquisición de datos, la cual a su vez recibirá las señales provenientes de los sensores.

Se plantea entonces como **problema** de este trabajo la inexistencia de una interfaz de software que permita a una aplicación de realidad virtual interactuar con la tarjeta inteligente “Flash Lite” y facilite a los desarrolladores la programación del módulo de software dedicado a la adquisición de datos.

El **objeto de estudio** de este proyecto lo componen los Dispositivos de entrada y salida para los Sistemas de Realidad Virtual y el **campo de acción** es la adquisición de datos a través de tarjetas inteligentes y su comunicación con el puerto serie de una microcomputadora

Este trabajo tiene como **objetivo general** desarrollar una interfaz de software que facilite a los programadores de sistemas de realidad virtual el desarrollo de módulos que permitan interactuar con la tarjeta inteligente “Flash lite”.

Tareas planteadas:

- Investigar los principales tipos de sensores utilizados en aplicaciones de Realidad Virtual, así como de cuales dispone la empresa.
- Investigar el trabajo con sensores en algunas herramientas conocidas para el desarrollo de simuladores y juegos en el mundo.
- Estudiar el uso de La Flash Lite (FL) como nueva alternativa para el manejo y control de los sensores.
- Estudiar el uso del puerto serie como puerto de comunicación entre la computadora y La Flash Lite.
- Plantear una solución que dé respuesta al problema planteado.
- Seleccionar las herramientas y metodologías para desarrollar la aplicación.

- Realizar la ingeniería de software a la solución propuesta.
- Implementar la solución propuesta.

La tesis está estructurada en 4 Capítulos. En el capítulo 1 “Fundamentación Teórica” se realiza un estudio del uso de los dispositivos de E/S en los SRV, herramientas y técnicas utilizadas en el mundo para el manejo de los mismos. En el capítulo 2 “Soluciones técnicas” se describen los elementos técnicos seleccionados para dar respuesta al problema planteado. En el capítulo 3 “Características del sistema” se sientan las bases de los que resultará ser la biblioteca de clases, se crea un modelo de dominio y se capturan los requisitos funcionales y no funcionales. Se describen las principales funcionalidades del sistema. En el capitulo 4 “Diseño e Implementación” quedan establecidas las bases para el desarrollo del sistema se presenta un diagrama de clases y la interacción entre objetos de dichas clases a través de los diagramas de secuencias, Se muestra el diagrama de componentes y el diagrama de despliegue indicando como quedaría distribuido el sistema una vez que este hay sido terminado.

1

Capítulo 1 Fundamentación Teórica

Introducción

En este capítulo se exponen conceptos que son importantes analizar sobre el mundo de la realidad virtual. Temas tan interesante como la simulación y el uso del tiempo real en aplicaciones de realidad virtual. Se introduce al conocimiento de los dispositivos de Entrada /Salida que es el objeto de estudio de este trabajo. Se hace el estudio del trabajo de una de las herramientas existentes para el desarrollo de aplicaciones de Realidad Virtual, la World Toolkit de ella se analiza e el manejo que hace con los sensores y el uso del puerto serie como medio de comunicación entres los dispositivos y la computadora. Se introduce el término de Flash Lite (FL) como una nueva alternativa para la conexión con diferentes tipos de dispositivos en aplicaciones de realidad virtual.

1.1 Sistemas de Realidad Virtual.

Qué es la realidad virtual

Existen varios autores que a partir de la caracterización de lo que consideran Sistemas de Realidad Virtual han intentado dar una definición, citando algunos ejemplos:

Los sistemas de realidad virtual son sistema informático interactivo que ofrece una percepción sensorial al usuario de un mundo tridimensional sintético que suplanta al real.[1]

“Un SRV es aquel que da al usuario la experiencia de estar inmerso en un entorno sintético” (H.Fuchs). [1]

“La RV se caracteriza por ofrecer la ilusión de participación en un entorno sintético en vez de una observación externa del mismo. La RV es una experiencia inmersiva y multisensorial” (M. A. Gigante). [1]

“Un entorno virtual se caracteriza por ser interactivo, con un procesamiento especial de imagen, sonido y tacto, para convencer al usuario de que se encuentra inmerso en un espacio sintético” (S.R.Ellis). [1]

Se puede concluir que un SRV podría definirse como un sistema informático que simula eventos o procesos del mundo real, en un tiempo real durante la cual el usuario ingresa, a través del uso de sofisticados dispositivos de entrada, a "mundos" que aparentan ser reales, (como se muestra en la Fig. 1), utilizando un mecanismo de desarrollar la acústica el tacto y la visualización de un mundo virtual. Resultando inmerso en ambientes altamente participativos, de origen artificial. Esta tecnología permite la interacción y propone la inmersión de los sentidos en el mundo simulado.

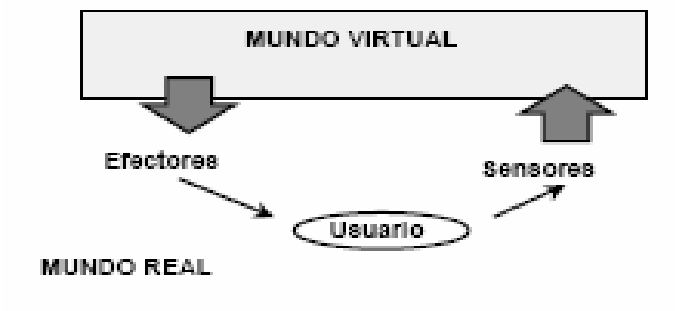

Figura 1 Representación gráfica del Sistema de Realidad Virtual.

Objetivos de la Realidad Virtual

La meta básica de la RV es producir un ambiente que sea indiferenciado a la realidad física crearlo con objetos, Poder presenciar un objeto o estar dentro de él. Definir las relaciones entre ellos y la naturaleza de las interacciones entre los mismos. Que varias personas interactúen en entornos que no existen en la realidad sino que han sido creados para distintos fines. En estos entornos el individuo solo debe preocuparse por actuar, ya que el espacio que antes se debía imaginar, es facilitado por medios tecnológicos. [2]

Características de un Sistema de Realidad Virtual (SRV):

- ✓ ***La inmersión****:* propiedad mediante la cual el usuario tiene la sensación de encontrarse dentro de un mundo tridimensional.
- ✓ ***Existencia de un punto de observación o referencia:*** permite determinar ubicación y posición de observación del usuario dentro del mundo virtual
- ✓ ***Navegación:*** propiedad que permite al usuario cambiar su posición de observación.
- ✓ ***Manipulación****:* característica que posibilita la interacción y transformación del medio ambiente virtual.
- ✓ ***Capacidad de síntesis:*** las imágenes se generan el tiempo real según la posición del usuario. Transformación proyectiva. Visibilidad.
- ✓ ***Ilusión de realidad:*** Factores físicos (estimulación real, estereoscopia, localización sónica, etc.). Factores psicológicos (credibilidad del mundo virtual, interacción natural, comportamiento, movilidad, experiencias compartidas)

Parámetros para Comparar SRV:

Para poder evaluar a un sistema realidad virtual como tal, es necesario que cumpla con ciertos elementos, la presencia de estos elementos nos garantizan en parte que se cumplan los objetivos de los SRV. A continuación se mencionan los principales que deben estar presentes en toda aplicación de la RV. Los parámetros en esencia son:

Simulación: Modelar un sistema, ésta tiene que ser lo suficientemente realista para convencer al usuario donde regirán una serie de reglas, no necesariamente iguales a las de la vida real.

Interacción: Tener control de la exploración del sistema creado; de no existir esta interacción el sistema no dejaría de ser una película o recorrido fijado a priori. Para lograrla existen diversas interfaces hombre-maquina, que van desde teclado y mouse hasta guantes o trajes sensoriales. Donde el usuario pueda mover objetos (además de a sí mismo) y modificarlos, y que tales acciones, produzcan cambios en el mundo artificial.

Percepción (Inmersión): El factor más importante, algunos SRV se dirigirán principalmente a los sentidos (visual, auditivo, táctil) por medio de elementos externos (Cascos de Visualización, Guantes de Datos, Cabinas, etc.); otros tratan de llegar directamente al cerebro, evitando así las interfaces sensoriales externas; y otros, los más simples recurrirán a la imaginación del hombre para experimentar una realidad virtual relativa.

***Generación de imágenes*:** en un SRV las imágenes son generadas dependiendo de la perspectiva que pretenda observarse. Esto es debido a la total libertad de movimientos que disfruta el usuario, y que hace imposible tener guardadas todas las imágenes correspondientes a todos los posibles puntos de vista. Los SRV poseen una base de datos con todos los elementos que componen el mundo virtual, a partir de la cual, éstos generan la información que se mostrara al usuario.

Tridimensionalidad: Debe existir realmente una dimensión de profundidad. Para conseguir este efecto los objetos del mundo virtual deben tener asociada una tercera dimensión que marque su profundidad en la pantalla.

Clasificación de los SRV.

En los SRV los elementos de hardware (HW) y de software (SW) se interrelacionan fuertemente. La base en todo SRV son los dispositivos de entrada y salida (E/S) ya que a partir de ellos, se conocen las limitaciones, el modo de presentación visual y el nivel de inmersión. Estas características sirven de bases para clasificarlos. A continuación es relacionan las características fundamentales de cada tipo de SRV con los Dispositivos de E/S mas comúnmente usados.

Atendiendo al modo de presentación visual (como le aparece al usuario)

Sistemas Inmersivos: Todo estímulo es generado por el SRV. El usuario no se ve a sí mismo. Estimulación separada de cada ojo. Esto son sistemas donde el usuario es sumergido en el mundo virtual sin formar parte de él, aunque siente todo los efectos provocados por él sistema. En estos se utilizan por lo general cascos de RV.

Sistemas Proyectivos: El usuario se introduce en un entorno en cuyas paredes se proyecta el mundo virtual. El usuario se ve a sí mismo y tiene la posibilidad de compartir experiencia, puede llegar en casos a ser multiusuario. El dispositivo esencial son los proyectores.

Sistemas Sobremesa: La proyección del mundo virtual se realiza en un monitor, puede existir además la estereoscopia. El usuario se ve a sí mismo. En estos sistemas se produce la perdida de sensación inmersiva. Los dispositivos fundamentales son el monitor, el teclado, el *mouse* y en algunos casos los *joystick*.

La inmersión en los sistemas de realidad virtual puede llegar a ser nula, parcial. La siguiente clasificación es basada en el nivel de inmersión de dichos sistemas.

SRV de escritorio: o también conocidos Sistemas Ventanas, *Windows on a World Systems (WoW)* Se han definido como SRV sin Inmersión. Algunos sistemas utilizan un monitor convencional para mostrar el mundo virtual. Estos sistemas tratan de hacer que la imagen que aparece en la pantalla luzca real y que los objetos, en ella representada actúen con realismo donde se engloban todas aquellas aplicaciones que muestran una imagen 2D ó 3D en el monitor en lugar de proyectarla en un “casco”. Los ejemplos típicos

son los simuladores de vuelos y la mayoría de los juegos para computadoras. Algunos tendrán interfaces sofisticadas, pero tienen en común el concepto de 3D en 2D.

Sistemas de Mapeo por Video: También es conocido por RV en segunda persona. Son aquellos sistemas en los cuales el usuario sabe que esta en el mundo virtual porque se ve a sí mismo dentro de la escena por medio de la proyección de su imagen en un fondo o ambiente, las cuales son visualizadas en la pantalla. Aplica la idea de "ver para creer" para inducir la sensación de presencia. Este enfoque se basa en la filmación, mediante cámaras de vídeo, de una o más personas y la incorporación de dichas imágenes a la pantalla del computador, donde podrán interactuar - en tiempo real – con otros usuarios o con imágenes gráficas generadas por el computador.

Sistemas de Telepresencia: Esta tecnología vincula sensores remotos en el mundo real con los sentidos de un operador humano. Los sensores utilizados pueden hallarse instalados en un robot. De esta forma el usuario puede operar el equipo como si fuera parte de él; utilizan generalmente elementos como cámaras, micrófonos, dispositivos táctiles y de fuerza con elementos de retroalimentación, ligados a elementos de control remoto para permitir al usuario manipular robots u otros dispositivos ubicados en lugares remotos mientras sé esta experimentando en forma virtual.

***Sistemas Inmersivos*:** Permiten que el usuario pueda sentirse "sumergido" en el interior del mundo virtual. El fenómeno de inmersión puede experimentarse mediante 4 modalidades diferentes, dependiendo de la estrategia adoptada para generar esta ilusión. Ellas son:

- El operador aislado
- La cabina personal
- La cabina colectiva (*pods*, *group cab*)
- La caverna o cueva (*cave*)

Estos sistemas inmersivos se encuentran generalmente equipados con un casco-visor HMD. Este dispositivo está dotado de un casco o máscara que contiene recursos visuales, en forma de dos pantallas miniaturas coordinadas para producir visión estereoscópica y recursos acústicos de efectos tridimensionales.

Otra forma interesante de sistemas inmersivos se basa en el uso de múltiples pantallas de proyección de gran tamaño dispuestas ortogonalmente entre sí para crear un ambiente tridimensional o caverna (cave) en la cual se ubica a un grupo de usuarios. De estos usuarios, hay uno que asume la tarea de navegación, mientras los demás pueden dedicarse a visualizar los ambientes de RV dinamizados en tiempo real.

1.1.1 Técnicas de simulación en tiempo real

La principal característica que debe cumplir las aplicaciones informáticas de simulación en tiempo real es la interactividad, es decir que debe mostrar las imágenes que componen al mundo a mediad que este se va produciendo de manera que si ocurre algún cambio producido por el usuario se refleje de forma instantánea. El principal desafío desde el punto de vista grafico es optimizar el coste de los cálculos para realizar las visualizaciones.

Concepto de tiempo real

Por tiempo real se entiende la capacidad de un sistema de cálculo o simulación para proporcionar una respuesta en un tiempo igual al que necesitaría el sistema real simulado o calculado.

En términos teóricos el tiempo real se consigue cuando el tiempo necesario para realizar el cálculo o la simulación es menor que el tiempo que se simula o calcula. En términos estrictos, si las entradas al sistema son continuas el tiempo de cálculo debería ser instantáneo, lo cual es imposible. [6.2]

Este concepto se puede aplicar a distintas ramas de la Informática y proviene de la ingeniería de control. En general la caracterización como “tiempo real” se refiere a la existencia de determinadas restricciones sobre el comportamiento temporal de nuestro sistema. Dependiendo del tipo de aplicación, esas limitaciones serán de una clase u otra. El concepto de tiempo real, por tanto, puede aplicarse de forma más o menos estricta según lo duras que sean las restricciones impuestas. [6.1]

Desde un punto de vista práctico, el tiempo real se consigue cuando el tiempo de respuesta del simulador es lo suficientemente corto como para que la percepción del fenómeno se corresponda con fidelidad al sistema real. Uno de los hechos que más diferencian la realidad misma de la simulación consiste en que en la simulación se necesitan incrementos finitos de tiempo para realizar los cálculos y en la realidad

misma esto es de forma inmediata. El resultado es que nunca se alcanzará un tiempo de respuesta nulo. [6.2] (Ver figura 2)

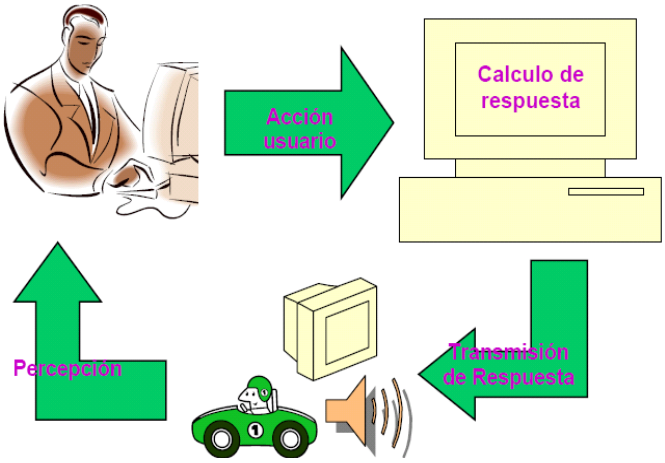

Figura 2 Ciclo de simulación

Creación de imágenes sintéticas con la suficiente rapidez como para que el usuario pueda interactuar con un entorno virtual. [2]

Un simulador de vuelo es un ejemplo de simulador de tiempo real, en el que la parte simulada del sistema (aéreo), está conectada a la parte real del sistema (piloto).

1.2 Simulación

1.2.1 Concepto de simulador

Un simulador puede usar cualquier combinación de sonido, vista, movimiento y olor para hacer sentir las experiencias de una situación real. Algunos videojuegos son buenos ejemplos de simuladores inferiores. [5]

En términos generales, un simulador es la representación por modelos matemáticos (MM) de un proceso en una computadora. Estos modelos muestran una realidad en la cual la complejidad de los eventos o situaciones pueden controlarse. Algunas características principales de la simulación son sus herramientas

audiovisuales e interactivas, como por ejemplo el diseño gráfico de las imágenes tridimensionales de personas o personajes y de espacios o lugares, que dan la sensación de que estás en un escenario real. [1]

En resumen un simulador es un dispositivo que permite que los usuarios se emerjan en un SRV. Utiliza modelos matemáticos físicos y gráficos para lograr el realismo que requieren los sistemas virtuales para lograr un realismo.

1.2.2 Surgimiento de los simuladores

Los simuladores surgen a partir de los videojuegos como respuesta a la necesidad de entrenarse en diferentes campos y han evolucionado más rápido que ellos. Actualmente, la tecnología que usan los simuladores, supera por mucho lo que podemos tener en casa.

Los simuladores poseen características que los hacen diferentes a los videojuegos, pues su principal objetivo es hacer creer al usuario que se encuentra en una situación real, esto implica tener gráficas muy precisas y dispositivos especiales como lentes que permiten ver el panorama en 180° o guantes con los que se puede sentir como si se tocaran los objetos. [7].

1.3 Hardware.

En los SRV el hardware se encarga no solo de mandar señales al usuario sino también de recibir señales de este. El hardware consiste de dispositivos físicos que forman parte de un sistema de RV y son los que estimulan al usuario en distintas maneras. Estos estímulos son los que le permiten alimentar los sentidos del usuario, y así, inducirlo a un mundo creado para él. Por tener las características de poder recibir y enviar informaciones los dispositivos son conocidos como de Entrada y Salida (Input */Output*) [10]

Aunque los dispositivos tradicionales de Entrada salida (E/S) como el computador (Teclado, Mouse, Monitor, Joystick) ofrecen grandes facilidades para el manejo de información, están limitados cuando las aplicaciones van ganando en complejidad y sofisticación, como es el caso de la RV que requiere de otros dispositivos para poder aprovechar todo su potencial.

A través de los dispositivos de E/S se facilita la comunicación hombre-máquina. A través de los dispositivos de entrada el usuario puede transmitir sus órdenes al SRV, indicándole que desea desplazarse, cambiar el punto de vista, o interactuar con algún objeto del mundo virtual; y los dispositivos de salida, permiten que el usuario se "sienta" inmerso en el mundo virtual creado. [10]

A estos dispositivos se les conoce como sensores de los cuales se tienen varias definiciones

Sensor: Dispositivo que convierte un parámetro físico (como temperatura, presión, flujo, velocidad, posición) en una señal eléctrica. En algunos casos se le considera un sinónimo de traductor, pero un verdadero sensor contiene un sistema de acondicionamiento de la señal, de manera que es mucho más sencillo realizar una medición. [4]

Sensor: Un elemento eléctrico/mecánico/químico capaz de convertir una característica del entorno en una medida cuantitativa

Los sensores son en realidad unos elementos físicos que pertenecen a un tipo de dispositivo llamado transductor. Los transductores son unos elementos capaces de transformar una variable física en otra diferente. Los sensores son un tipo concreto de transductores que se caracterizan porque son usados para medir la variable transformada. La magnitud física que suele ser empleada por los sensores como resultado suele ser la tensión eléctrica, debido a la facilidad del trabajo con ella.

Desde el punto de vista de la forma de la variable de salida, podemos clasificar los sensores en dos grupos: analógicos, en los que la señal de salida es una señal continua, analógica; y digitales, que transforman la variable medida en una señal digital, a modo de pulsos o bits. En la actualidad los sensores más empleados son los digitales, debido sobre todo a la compatibilidad de su uso con los ordenadores.

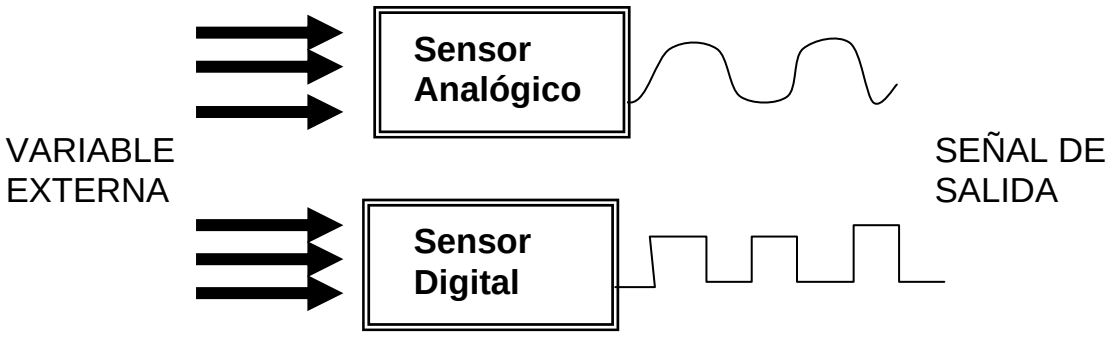

Figura 3 Clasificación de los sensores de acuerdo al tipo de señal.

1.3.1 Dispositivos de entrada (Sensores):

Capturan el estado y acciones del usuario que transmiten información hacia el entorno virtual.

En un sistema de realidad virtual no inmersiva, los dispositivos de entrada necesarios pueden reducirse a un simple mouse, pero cuando se trata de un sistema inmersivo, las cosas se empiezan a complicar. Hay que disponer de sensores de posición para averiguar la dirección en la que el usuario está mirando y su posición relativa en el mundo virtual. Simultáneamente, el sistema puede estar recibiendo información de un guante de datos y procesándola para mostrar una representación de la mano del usuario o cibernauta dentro del mundo virtual. Además, aun se debe resolver el problema del desplazamiento dentro de un entorno completamente tridimensional, utilizando dispositivos de entrada que fueron creados para aplicaciones bidimensionales (Mouse, Joysticks, etc.). Por otro lado, los dispositivos de salida permiten al usuario observar, oír, tocar, en resumen, “vivir” en el mundo creado. [10]

Cuando se trata de simulación los sensores se clasifican en dos grandes grupos generalizados

- Sensores de localización

Están ubicados en diferentes partes del cuerpo. Estos le dan sensación al usuario de ubicación en el mundo virtual. [3] En este grupo están algunos sensores como por ejemplo:

- Electromagnéticos
- Mecánicos.

- Sónicos. Ultrasonidos.
- Ópticos. Rotoscopia.

Figura 4 Sensores de localización

- Sensores de Control:

Los sensores de posición como también son conocidos permiten conocer los cambios de dirección que realiza el usuario y algunos permiten seguir el movimiento de los dedos Aquí se agrupan los sensores que posibilitan al usuario controlar el sistema virtual. [3] Por ejemplo:

- Ratones y joystick 3D
- Sensores de simulación
- Bioeléctrico
- Electro guantes.

Figura 5 Sensores de control (Electro guante).

1.3.2 Dispositivos de Salida (Efectores):

Estimulan los sentidos del usuario que transmite información del estado al entorno virtual. También se le conoce como actuadores porque son los que el usuario interactúa de una forma sencilla, ya que de ellos solo obtiene la información que necesita del mundo virtual.

Tipos:

- Dispositivos Visuales:

Una de las principales consideraciones en este tipo de dispositivos es el detalle de las imágenes contra la rapidez en la formación de las imágenes que forman las escenas, además de una visión monoscópica

contra una estereoscópica. La formación de escenas en tiempo real le da un sentido de realidad al usuario al eliminar la discontinuidad.[3]

- Visiocasco
- Sistemas Binoculares
- Monitor estereoscópico + gafas.
- Lentes LCD resplandecientes
- Despliegues montados en la Cabeza (*Head-Mounted Display* HMD)
- HMD proyectados.
- Monitor omni-direccional binocular (*Binocular Omni-Orientation Monitor* BOOM)

Figura 6 Dispositivos Visuales (HMD)

- Dispositivos auditivos:

El uso de sonido es muy importante dentro de los SRV puesto que el sistema auditivo es uno de los componentes perceptuales más importantes, proporciona información alternativa o suplementaria al usuario [3]:

- Sonido 3D
- Sonido Realista.
- Cascos auditivos.

Figura 7 Dispositivos auditivos

- Dispositivos hápticos:

El estimular el sentido del tacto, como permitir al usuario "tocar" objetos de manera que pueda sentir la forma, textura, temperatura, firmeza y fuerza de éstos, puede agregar un buen nivel de realismo al ambiente virtual. [3]

- Almohadillas inflables
- Plataformas móviles
- Cybertron.
- Guantes
- Mayordomos

Figura 8 Dispositivos hápticos

1.4 World ToolKit, como herramienta de desarrollo.

1.4.1 ¿Qué es WorldToolKit? Características

World ToolKit (WTK) es un entorno de desarrollo multiplataforma avanzado, para aplicaciones gráficas 3D en tiempo real con un alto rendimiento, integra aplicaciones 3D para usos científico y comercial. La biblioteca WTK tiene más de 900 funciones de alto nivel para configurar, interactuar y controlar las simulaciones en tiempo real. [11.3]

WTK cuenta con funciones incluidas para:

- Instanciar dispositivos
- Configurar el *display*
- Detección de colisiones
- Cargar geometrías desde archivos.
- Creación de geometrías dinámicas
- Comportamiento de los objetos
- Control de *render*
- Interfaces gráficas 2D.

Las aplicaciones de WTK permiten que objetos del mundo virtual puedan tener propiedades y comportamientos del mundo real. Los mundos son controlados por una variedad de dispositivos de

entrada sensores 2D hasta 6D; son representados por displays de computadoras o HMDS. Las simulaciones son construidas por nodos montados en jerarquía llamada escena gráfica.

1.4.2 Clases de WTK

WTK esta estructurado en una manera orientada a objeto, aunque no usa herencia o creación dinámicas. Las funciones WTK son orientadas a objetos, y están agrupadas en clases. El trabajo con los dispositivos de entrada y salida se centra fundamentalmente en las siguientes clases: [11.3]

- ***Sensor***: se pueden conectar sensores para transformar nodos, cámaras…
- ***Motion Links:*** conecta una fuente de información de posición y orientación con un objetivo, por ejemplo, un sensor con una cámara. El objetivo se mueve en correspondencia con el estado de la fuente.
- ***Serial Port:*** maneja un grupo de funciones que simplifican las tareas de comunicación con los puertos en serie

Cada clase tiene un *typedef* definiendo el tipo de objeto. Por ejemplo

- WTsensor es un objeto sensor

Los objetos se ocupan siempre con el uso de indicadores, son opacos forzando la abstracción de los datos. El estado de los objetos es accedido por funciones *set* y *get*. Todos los métodos de una clase tienen un nombramiento convencional que empiece como el de la clase

1.4.3 Los sensores en WTK.

En WTK el objeto sensor genera posición orientación y otros tipos de datos leyendo las entradas que originan en el mundo verdadero. Estas entradas se pueden utilizar para controlar el movimiento y otros aspectos del comportamiento de los objetos en la simulación. Usando los sensores y los motionlinks el usuario puede directamente acoplar los puntos de vistas, los objetos gráficos y las luces en el universo.

WorldToolKit apoya muchos de los sensores 3D y 6D (posición/orientación) que están disponibles ver tabla 2. Hay dos categorías principales para los sensores:

Sensores que generan entradas relativas, es decir, cambia en la posición y la orientación, dispositivos usados en el cuerpo genera típicamente los registros absolutos, es decir, los valores que corresponden a su localización espacial específica. En esta categoría están los dispositivos convencionales, tales como el ratón, la palanca de mando y las bolas isométricas tales como el Spaceball CIS de la tecnología del Jr. y de Spaceball de la bola de la geometría que responden a las fuerzas y a los esfuerzos de torsión aplicados por el usuario. Usando tales dispositivos, un objeto 3D se puede manipular, desplazar o rotar directamente, con la bola actuando como si esté conectado directamente con el objeto. Los sensores de la bola son también útiles para mover el punto de vista, aplicando dislocaciones y fuerzas rotatorias al movimiento y rotan el punto de vista.

La segunda categoría de sensor son los que generan registros absolutos, incluye a perseguidores electromagnéticos 6D tales como el Polhemus Fastrak y pájaro de la ascensión. Este tipo de sensor se puede utilizar para el punto de vista que sigue cuando está puesto a una exhibición cabeza-montada. Además de los dispositivos electromagnéticos, una variedad de dispositivos ultrasónicos y los dispositivos ópticos existen para seguir un valor absoluto de la posición y de la orientación. Un ejemplo es el perseguidor ultrasónico del ratón y de la cabeza de Logitech 3D.

Sin importar la tecnología de hardware subyacente por la cual funcionan, los objetos del sensor de WorldToolKit se tratan homogéneamente y se pueden utilizar alternativamente. Una vez que se cree un objeto del sensor, es mantenido automáticamente por el encargado de la simulación, al igual que los objetos a los cuales se une el sensor. [11.1]

Sensores soportados por WTK	
Standard Mouse	*Virtual i-o head tracker*
Spaceball	*Magellan*
Spaceball Space Controller	*Wayfinder*
Geoball	*Thrustmaster T2 steering wheel*
Isotrak, Isotrak II, Fastrak, and Insidetrak	*Thrustmaster serial FCS and WCS (serial joysticks)*
Bird, Extended range bird, and flock of	*CyberMaxx 2 head tracker*

birds	
Logitech 3d mouse (red baron)	*Pinch glove*
CrystalEyes and CrystalEyesVR	*5DT glove*
Fakespace BOOM	*Cyberglove (SGI only)*

Tabla 1 Listado de sensores soportados por WTK.

1.4.4 La construcción del sensor

WTK proporciona el uso fácil de macros para permitir que el usuario cree objetos de tipo sensor. Estos objetos son almacenados en estructuras WTsensor. Cada tipo de sensor soportado por WTK tiene su propio macro, el constructor del sensor

La siguiente tabla enumera la lista de macros de construcción del sensor y los sensores correspondientes que se construyan

MACRO	SENSOR
WTmouse_new	*Standard mouse*
WTspaceball_new	*Spaceball*
WTspaceballSC_new	*Spaceball Space Controller*
WTgeoball_new	*Geoball*
WTpolhemus_new	*Isotrak e IsotrakII* con un sensor
WTisotrak2_new	*IsotrakII* con 2 sensores
WTfastrak_new	*Fastrak*
WTinsidetrak_new	*Insidetrak*
WTbird_new	*Bird*
WTbirdERC_new	*Extended range bird*
WTredbaron_new	*Logitech 3d mouse*
WTboom_new	*Fakespace BOOM*
WTspacecontrol_new	*Magellan*
WTpercision_new	*Wayfinder*

WTiglasses_new	*virtual-o iglasses tracker*
WTjoyserial_new	*Thrustmaster FCS* y *WCS*
WTglove5dt_new	*5DT glove*

Tabla 2 macros de construcción de algunos sensores en WTK.

Los programadores pueden crear objetos del tipo sensor con la función genérica WTsensor_new del constructor del sensor o con una de las macros especificas del constructor del dispositivo de WTK.

1.4.5 Funcionamiento de los *drivers* para sensores.

Los manejadores (drivers) de sensores de WTK se escriben para permitir que el usuario tenga acceso a la información, tal como coordenadas absolutas, y presionado del botón, dependiendo del sensor.

WTK también proporciona al usuario la capacidad de cambiar como los datos leídos en el sensor son interpretados permitiendo que el usuario cree sus propias funciones de actualización.

Los manejadores de sensores escritos de WTK consisten en tres funciones importantes, una función abrir ("*open*"), una función cerrar ("*close*") y la función actualizar ("*update*").

WTK incluye un sistema completo de funciones para escribir su propio *driver* del sensor. Estas proporcionan una interfaz de *hardware* independiente a los varios dispositivos de entrada y salida. [11.1]

Función	Descripción
WTsensor_setrecord	Se utiliza para almacenar el registro relativo actual de la posición y de la orientación con su sensor.
WTsensor_setmiscdata	Se utiliza para almacenar datos misceláneos del sensor con el objeto sensor.
WTsensor_setupdatefin	Se utiliza para cambiar la función de la actualización de un sensor. La función de la actualización de un objeto sensor se fija inicialmente en la función WTsensor_new del constructor del objeto sensor.

Funciones abrir del sensor

Las funciones abrir del sensor abren las comunicaciones con los sensores; y se llama una sola vez cuando el sensor es construido con una llamada a una de las macros del sensor. El sensor esta listo para la comunicación y se obtiene le primer valor del sensor. Si el sensor retorna datos como la posición absoluta y orientaciones entonces ese valor se almacena para usarlo mas tarde para crear un valor relativo.

Funciones cerrar del sensor

Las funciones cerrar, cierran las comunicaciones con el sensor y rescatan cualquier parámetro del sistema que pudo haber sido cambiado. Se llama solamente una vez cuando cualquier objeto sensor es destruido o cuando se sale del universo.

Funciones actualizar del sensor

Estas funciones son las de mayor fuerza en el trabajo de los *drivers* de sensores. Son llamadas una vez a través del lazo de la simulación para cada sensor que este en el universo. Se obtiene un registro del sensor. Puede ser registro de posición/ orientación el registro es relativo en esta función de modo que el encargado de la simulación puede utilizar los valores para actualizar los punto de vistas, objetos o luces que se relacionen a los motionlinks de los sensores. [11.1]

En caso de no usar los driver del sensor se puede hacer acceso a los sensores a través del puerto serie utilizando la clase *Serial Port*.

1.4.6 Acceso al puerto serie en WTK

WTK cuenta con una clase que posibilita la comunicación con el puerto serie. Esta clase se utiliza por lo general cuando no se trabaja con los driver de los sensores. Ya que en la programación de dichos driver se usa de forma implícita y no es necesario que el programador haga uso de las mismas

Sus principales funciones son:

WTserial_new:

*WTserial *WTserial_new (*

*char *port,*

int baud,

char parity,

int databits,

int stopbits,

int buffersize)

Esta función crea un objeto del puerto serial. Una vez que esté creado, el objeto del puerto serial se pueda pasar adentro a la función genérica WTsensor_new de la construcción del sensor cuando se desea crear un objeto del sensor que expedientes sean obtenidos por la comunicación sobre el puerto serial. Sin embargo, si se está utilizando la macro del dispositivo especifico para crear un objeto del sensor, es innecesario llamar WTserial_new porque la macro crea el objeto del puerto serial llamando WTserial_new. [11.2]

Argumentos que utiliza esta función:

Port:	Cadena de caracteres que identifican el puerto de la computadora
Baud:	Velocidad con la que ocurrirá la comunicación sobre el puerto serie (valores validos: 1200, 2400, 4800, 9600, 19200, 38400, 57600).
Parity:	Especifica la paridad “N” para la no paridad.
Databits:	Números de bits de datos (8 para 8 bits de datos).
Stopbits:	Números de bits de parada (1 para un bit de stop).
Buffersize:	especifica el numero de bytes en el buffer circular

WTserial_delete:

*void WTserial_delete (WTserial *serial)*

Esta función libera un objeto del puerto serial. Si se está utilizando el puerto serial funciona como parte del driver del sensor de WTK, es innecesario llamar WTserial_delete cuando se desea terminar la comunicación con el objeto del sensor porque WTsensor_delete llama a WTserial_delete. [11.2]

Wtserial_read:

short WTserial_read (

*WTserial *serial,*

*char *data,*

int length,

FLAG retry)

Esta función lee una secuencia de una longitud especificada (número de bytes) de un puerto serial dentro de un buffer llamado data. Vuelve el número de los caracteres que fueron leídos realmente. La longitud solicitada debe ser no más grande que el tamaño del buffer del puerto serial.

La bandera de la recomprobación se utiliza para especificar qué debe suceder si pocos bytes que el número solicitado están realmente disponibles. Esto puede suceder si las llamadas de WTserial_read se hacen en tal sucesión rápida que el número solicitado de caracteres no ha tenido tiempo para llegar. [11.2]

WTserial_ntoread:

*int WTserial_ntoread (WTserial *serial)*

Esta función determina cuantos caracteres esperan por ser leídos en le puerto serie

WTserial_write:

short WTserial_write (

*WTserial *serial,*

*char *buffer,*

int length)

Esta función escribe una secuencia de una longitud dada a un puerto serial y vuelve el número de los caracteres que fueron escritos con éxito. El argumento serial se refiere al objeto del puerto serial a el cual usted desea escribir. El argumento buffer contiene la secuencia que se desea escribir al puerto serial. El argumento longitud (*lenght*) es el número de los caracteres a escribir.

WTserial_setdata:

*void WTserial_setdata (WTserial *serial, void *data)*

Esta función fija una zona de informaciones definida por el usuario en un objeto del puerto serial. La aplicación privada de data se puede almacenar en cualquier estructura. Dentro de un objeto del puerto serial se guarda un puntero de tipo void a dicha estructura

WTserial_getdata:

*void *WTserial_getdata (WTserial *serial)*

Esta función recupera los datos definidos por el usuario, almacenados dentro de un objeto del puerto serial. Se debe echar el valor vuelto por esta función al mismo tipo usado para almacenar los datos con la función de WTserial_setdata.

1.5 Flash Lite.

La Flash Lite es una tarjeta que brinda la posibilidad de conectar varios dispositivos de entrada y salida la misma pertenece a la compañía *JK Microsystems* la misma tiene las siguientes características. La misma utiliza la conexión con el puerto serie de las computadoras para el flujo de información [9]

	Flash Lite 186
Procesador	RDC R8822
Velocidad Reloj	33MHz
Memoria RAM	512K
Memoria Flash	512K
Opciones de Expansión	DiskOnChip
E/S	44 líneas digitales
Puerto Serie	1-RS232, 1-RS232 ó RS485
Voltaje de Entrada	7-34 V
Consumo	1.8 Watts
Sistema Operativo	JK DOS 3.3
Watchdog	SI
Opciones de Expansión	Tarjetas Periféricas

Tabla 3 características físicas de la flash lite 186

Los puertos de Entrada ó Salida de este procesador compacto están agrupados de la siguiente forma:

Puertos	Cantidad de Bits
A	4
B	4
C	8
D	8

E	8
F	8
TOTAL	40

Tabla 4 Puertos de la Flash Lite 186

Según las necesidades del diseño de hardware los 6 puertos se programan independientemente como entrada o salida.

La firma *JK Microsystems* ofrece una tarjeta de conversión A/D que se acopla a la Flash Lite 186; la misma posee las siguientes características: 12 bits de conversión en base a al conversor MAX197, tiempo de conversión de 6 µs, posee 8 canales de entradas de conversión digital y los voltajes de conversión a la entrada pueden tener los siguientes rangos: ± 10V, ±5V, 0V a 10 V y 0V a 5V.

1.6 Control del puerto serie

El puerto serie es una interfaz de comunicación entre ordenadores y periféricos en donde la información es transmitida bit a bit enviando un solo bit a la vez. En Windows no es posible acceder a los dispositivos físicos directamente, a través de las direcciones de sus puertos. A menos que se este programando sobre un driver los acceso a los mismo deben hacerse a través de funciones de la Interfaz de Programación de Aplicaciones, cuyo acrónimo en inglés es API (*Application Programming Interface*). [8]

Los puertos serie, por tratarse de dispositivos incluidos como parte de las Computadoras Personales desde sus comienzos, están muy bien integrados en el API de Windows que contiene funciones para manejarlos.

Windows trata los puertos serie (y también el paralelo), como si se tratase de un fichero de entrada y salida. La única peculiaridad es que su comportamiento es asíncrono, y esta característica influye mucho en el modo en que de deben programar las aplicaciones cuando usan uno de estos puertos. El comportamiento asíncrono se debe a varias características de este tipo de comunicación, los datos se envían secuencialmente, a una velocidad relativamente baja. El sistema tiene que estar preparado para recibir los datos en el momento en que están disponibles, debido a que el tamaño del buffer del que dispone el sistema operativo es finito y si no se retiran con frecuencia los datos recibidos pude ocurrir la

pérdida de información. Los datos que se reciben por uno de estos canales hay que leerlos cuando llegan, casi nunca sabremos cuándo el dispositivo que se tiene conectado al otro extremo del cable decide enviar los datos. [8]

En cuanto a la escritura, no se pude prever con precisión si el dispositivo al que se envían los datos los va a procesar con la velocidad requerida, o si está o no preparado para recibirlos.

1.6.1 API de WINDOWS

Las API son un conjunto de funciones residentes en bibliotecas (generalmente dinámicas) que permiten que una aplicación corra bajo el sistema operativo Windows.

Debido a su estrecha relación con el desarrollo de software, los programas en sus especificaciones generalmente explicitan la versión de la API del sistema operativo, mediante diversas nomenclaturas tales como la versión específica del sistema operativo (para Windows 98, por ejemplo), o explicitando la versión del conjunto de librerías (Plataforma Win32, etc.). [12]

Las funciones API se dividen en varias categorías:

- Depuración y manejo de errores
- E/S de dispositivos
- DLLs, procesos e hilos
- Comunicación entre procesos
- Manejo de la memoria
- Monitoreo del desempeño
- Manejo de energía
- Almacenamiento
- Información del sistema
- GDI (interfaz gráfica) de Windows
- Interfaz de usuario de Windows

Las versiones modernas de Windows utilizan la API de 32 bits llamada Win32. Está compuesta por funciones en C almacenadas en librerías de enlace dinámico (DLL), especialmente en las del núcleo: [12]

- kernel32.dll
- user32.dll
- gdi32.dll

Aunque la implementación de Microsoft tiene derechos de autor, generalmente se acepta que otras empresas puedan emular Windows proporcionando APIs idénticas, sin que implique violación de derechos de autor.

Las funciones para el manejo de los dispositivos de E/S que se conectan al puerto serie radican en la biblioteca kernel32.dll de la cual podemos citar un conjunto de funciones que manejan las operaciones en el puerto serie. Las más utilizadas son: [13]

***CreateFile*:** Crea un manipulador que puede ser usado para acceder al objeto.

***CloseHandle*:** Invalida el manipulador de objeto especificado, decrementa el contador de manipuladores del objeto y realiza una comprobación del objeto. Una vez que el último manipulador del objeto ha sido cerrado, el objeto se elimina del sistema operativo. Se usa para cerrar manipuladores creados por la función CreateFile

***GetComState*:** Esta función rellena un bloque de control de dispositivo con los valores actuales de control para el dispositivo de comunicación

***SetCommState*:** Configura un dispositivo de comunicaciones de acuerdo con las especificaciones de un bloque de control de dispositivo. La función reinicia todo el hardware y valores de control, pero no vacía las colas de entrada o salida.

ReadFile: Lee datos desde un fichero, comenzando en la posición indicada por el puntero del fichero. Después de que la operación de lectura ha sido completada, el puntero del fichero se ajusta según el número de bytes leídos, a no ser que el manipulador de fichero haya sido creado con el atributo de

superposición (overlapped). Si el manipulador de fichero es creado para entradas y salidas superpuestas, la aplicación debe ajustar la posición del puntero del fichero después de la operación de lectura

***WriteFile*:** Escribe datos a un fichero, está diseñada tanto para operaciones síncronas como asíncronas. La función empieza escribiendo datos al fichero en la posición indicada pro el puntero del fichero. Después de que la operación de escritura ha sido completada, el puntero del fichero se ajusta en el número de bytes escritos actualmente, excepto cuando el fichero ha sido abierto con FILE_FLAG_OVERLAPPED. Si el manipulador de fichero fue creado para entrada y salida superpuesta (overlapped I/O), la aplicación debe ajustar la posición del puntero de fichero después de que la operación de escritura haya terminado.

***ClearCommError*:** Recupera información sobre errores de comunicación e informa sobre el estado actual del dispositivo de comunicaciones. La función es llamada cuando ocurren errores de comunicación, y elimina la vadera de error del dispositivo para permitir operaciones de entrada y salida adicionales. Si la función tiene éxito, el valor de retorno es TRUE. Si la función falla, el valor de retorno es FALSE. Para obtener mayor información sobre el error, se llama a GetLastError.

***GetCommMask*:** Recupera el valor de la máscara de eventos para el dispositivo de comunicaciones especificado. Usa una variable de máscara de 32 bits para indicar el conjunto de eventos a monitorizar para un recurso de comunicaciones en particular. Se puede especificar un manipulador de un recurso de comunicaciones en la llamada a la función WaitCommEvent, que esperará a que ocurra uno de los eventos. Para modificar la máscara de eventos para un recurso de comunicaciones, usar la función SetCommMask.

***SetCommMask*:** Especifica el conjunto de eventos que pueden monitorizarse para un recurso de comunicaciones en particular. Se puede especificar un manipulador de recurso de comunicaciones en una llamada a la función WaitCommEvent, que esperará a que ocurra uno de los eventos. Para obtener la máscara de eventos actual para un recurso de comunicaciones, usar la función GetCommMask.

***WaitCommEvent*:** Espera a que ocurra un evento para el dispositivo de comunicaciones especificado. El conjunto de eventos que son monitorizados por esta función está contenido en la máscara de eventos asociada con el manipulador del dispositivo.

Conclusiones

En el transcurso de este capítulo, como base para el entendimiento del tema en que se desenvolverá este proyecto, se mencionan aspectos de La Realidad Virtual que enmarcan el uso de los dispositivos de E/S las características de los mismos y el trabajo que hace la herramienta WTK con dichos dispositivos. Se estudio a la tarjeta inteligente Flash Lite, las principales características de dicha tarjeta y se estudio el acceso al puerto serie ya que es el puerto de comunicación establecido por ahora para dicho dispositivo.

2

Capítulo 2 Soluciones Técnicas

Introducción

En este capítulo se describe la solución al problema y los elementos fundamentales que son importantes tener en cuenta en el desarrollo de la solución ya que constituyen las bases técnicas del producto a crear. Se describirá como se hará el acceso al puerto serie como puerto de comunicación y la descripción del protocolo de comunicaciones que se utilizará para establecer la comunicación.

2.1 Propuesta del sistema

Se propone para dar respuesta al problema planteado crear un modulo que va actuar como una interfaz entre la herramienta de desarrollo de aplicaciones virtuales y los dispositivos de entrada y salida. La cual permitirá de manera flexible utilizar todo tipo de dispositivos tanto de entrada comos salida.

Para el desarrollo de esta biblioteca se va a utilizara la Flash lite 186 que tiene acoplada un conversor analógico-digital. Se seleccionó esta tarjeta inteligente porque la misma cuenta con un procesador que s capaz de obtener todos los datos emitidos por los sensores y enviarlos a la computadora personal por el puerto serie. Esta le ahorra tiempo de procesamiento a la computadora y de esta forma se gana en tiempo en la simulación que es unos de los aspectos fundamentales en todo SRV.

Esta biblioteca de clase debe dar la posibilidad de manipular y configurar el puerto de comunicación, que para esta primera versión es el puerto serie. Esta biblioteca se utilizará por lo general en aplicaciones que utilicen la simulación por encuesta por lo tanto se debe de crear una estructura con los valores reales emitidos por los sensores. Se envía en forma de estructura para que el sistema a su conveniencia haga el uso de la misma.

Con el uso de la esta biblioteca los programadores no necesitan conocer cuales son las características ni el rango de funcionamiento en el que emiten sus señales los sensores conectados las misma.

2.2 Acceso y Configuración del puerto serie

Para el acceso y la configuración del puerto serie se hará uso de las funciones que la API Win 32 almacenadas en la biblioteca kernel32 para el manejo y configuración de los puertos de las computadoras.

Las funciones API a utilizar son:

CreateFile: Con el objetivo de obtener un manejador del puerto a usar.

GetComState: rellena una estructura *DCB* para control del dispositivo.

SetCommState: Configurar el dispositivo a partir de la estructura DCB).

ReadFile: Leer los datos de entrada en el puerto.

WriteFile: Escribir los datos en el puerto para ser enviados a la FL.

ClearCommError: para el manejo y control de errores usando como base la función GetLastError.

Por tanto esta primera versión solamente tendrá implementada la comunicación en el Sistema operativo (SO) Windows

2.3 Protocolo de comunicación con la Flash Lite 186.

Para establecer una comunicación entre la FL 186 y la computadora existe un protocolo de comunicación que esta definido por comandos y bloques de datos. Es importante que tanto la FL como la aplicación a crear manejen bien estos conceptos ya que es el único modo de comunicación que ambas manejarían a lo unísono. Y es de vital importancia ya que la comunicación se realiza en tiempo real. Estos comandos pueden ser de solicitud o de respuestas. Se considera como comandos de solicitud aquellos que la aplicación envía a la FL dentro de un paquete de datos haciendo solicitud de una tarea que la FL sea capaz de generar. A continuación se describe el propósito de cada comando y con los fines a usar:

Comandos:

Bloque 2: Repetir Bloque.

1b	2	#buf	CRC

Repetir la transmisión del último bloque. La aplicación utiliza este comando cuando se ha recibido con anterioridad un bloque de datos y el bit de control de error (CRC) no es correcto o hubo algún problema con el paquete de datos y es necesario recibir la información.

El bit # buf indica cual de los tres buffer de transmisión es el que se vuelve a enviar.

#buf=0 Tx bloque 0

#buf=1 Tx bloque 1

#buf=2 TX bloque 2

#buf = 255 se debe enviar los tres últimos mensajes almacenados en los buffer

Si # buf es distinto 0, 1, 2, 3 ó 255 hay error en el comando recibido. Se recibe de la FL un Bloque 2 con #buf = 255

Bloque 3 Parar la Transmisión

1b	3	#buf	CRC

Quitar el permiso de Transmisión de la FL, no se transmitirán mas cambios de datos hasta que llegue el comando Comenzar Transmisión de Cambio. Este comando se utiliza cuando la aplicación no requiere de una continua recepción de información.

Bloque 4 Solicitud de los datos de entrada.

1b	4	#buf	CRC

Este comando se utiliza cuando la aplicación requiere conocer cuales son los valores que están emitiendo todos los dispositivos conectados a los puertos de entrada de la FL. Cuando se solicita este comando siempre ejecuta, aunque no este dado el comando de Parar la Transmisión.

Como respuesta a este comando la FL envía a la aplicación el siguiente bloque de datos. Conocido como el comando 0

Bloque 0 Enviar todos los datos de entrada.

1b	0	#buf	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4

1-4				5-8			
Byte 5	CH0 L	CH0 H	CH1 L	CH1 H	CH2 L	CH2 H	CH3 L
9-12				13-16			
CH3 H	CH4 L	CH4 H	CH5 L	CH5 H	CH6 L	CH6 H	CH7L
17-20				21-24			
CH7 H	S1 L	S1 H	S2 L	S2 H	S3 L	S3 H	S4 L
25-28				29-32			
S4H	CRC						
33-34							

CHn L- CHn H: dato correspondiente al canal A/D “n”.

SnL-SnH: dato correspondiente al sensor “n”.

Byte 0: Puerto A

Byte 1: Puerto B

Byte 2: Puerto C

Byte 3: Puerto D

Byte 4: Puerto E

Byte 5: Puerto F.

Bloque 5 Comenzar la transmisión de cambios

1b	5	#buf	CRC

Cuando se envía este comando a la FL se activa la bandera de transmisión de cambios en TRUE, lo cual habilita la transmisión de los cambios que ocurren en los puertos de entrada cada vez que se detecte. De

igual forma ocurre con las entradas del canal analógica Para inhabilitar este comando se debe enviar el comando Bloque 3 para parar la transmisión.

Si se cumple que:

- Hay cambios en algunas de las informaciones de entrada del sistema (canal analógico, puertos de entrada ó encoder)

- Hay permiso para la Transmisión de los Cambios

Se conformará el Bloque 1 en la cual se incorporan los datos que han variado. La estructura y codificación de los datos de información que se envía es la siguiente:

T Dato n	Dato n	Longitud
00, 01, … , 05	Pto A, Pto B, ... Pto F	1 byte
10, 11, … , 17	Canal 1, Canal 2, … , Canal 8	2 byte
20, 21, 22, 23	Encod 1, Encod 2, Encod 3, Encod 4	2 byte

Bloque 1 Cambios de los datos de entrada

1b	1	#buf	Cant Dat	TDato 1	Dato 1	TDato 2	Dato 2
1-4				5-8			
…	TDato n	Dato n	CRC	Longitud bloque 1= 4 + Cant Dat			
Longitud variable				Cant Dat = n*2			

Bloque 6 Comenzar transmisión de los datos de los sensores.

1b	6	#buf	CRC

Cuando se solicita este comando siempre ejecuta, aunque este dado el comando de Parar la Transmisión. Se envía con el objetivo de recibir los valores que estén emitiendo los sensores (encoders) en el momento en que se solicita. Por lo tanto se debe esperar una respuesta de la FL utilizando el comando 13 que es un bloque que contiene los valores de los 4 sensores que están conectados a la FL.

Bloque 13 Enviar los datos de los encoder

1b	13	#buf	S1 L	S1 H	S2 L	S2 H	S3 L
1-4				5-8			
S3 H	S4 L	S4 H	CRC				
9-12							

Bloque 7 Transmitir los datos de salida.

1b	7	#buf	CRC

Cuando se solicita este comando siempre se ejecuta en la FL, aunque no este dado el comando de Parar la Transmisión. Como respuesta se envía al procesador los datos que están aplicados a la salida utilizando el comando 10.

Bloque 10 Todos los datos de salida

1b	10	#buf	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4
1-4				5-8			
Byte 5	CRC						
9-10							

El bloque 10 también se puede comportar como un comando de solicitud. Cuando la FL recibe este comando se actualizan los valores de todos puertos que están programados como salida en la Flash Lite 186. Todo esto teniendo en cuenta el dato de configuración de los puertos de la Flash Lite 186 que se encuentra en el fichero Reg.Dat. No se genera mensaje de respuesta.

En el caso de que solo se quieran actualizar algunos datos de salida se hace uso del comando 11 que posibilita actualizar uno ó varios puertos de acuerdo a los requerimientos. Evitando así tener que mandar los datos de los 6 posibles puertos de salida. El programa tiene en cuenta cuales son los puertos programados como salida para evitar daños en los puertos de E/S de la Flash Lite 186. No se genera mensaje de respuesta

De acuerdo a los datos que llegan se hace una copia de ellos y son sacados por los puertos de salida de la FL.

Bloque 11 Actualizar los datos de salida.

1b	11	#buf	Cant Dat	TDato 1	Dato 1	TDato 2	Dato 2
1-4				5-8			
…	TDato n	Dato n	CRC				
Longitud variable							

Longitud del bloque 11 = 4 + Cant Dat

Cant Dat= n*2

Conclusiones

Con este capítulo queda explícita la propuesta de solución al problema de este trabajo y sentadas las bases técnicas por las que se regirá el sistema. De éstas se diseñará una estructura de clases en correspondencia con las técnicas citadas y para lograr los objetivos del proyecto.

3

Capítulo 3 Características del sistema

Introducción

En este capítulo se comienza a tener la visión del sistema a desarrollar. Aquí se inicia la concepción práctica del producto a elaborar, sobre la base de las dificultades, necesidades y características organizacionales del cliente, conociendo ya las técnicas a utilizar descritas en el capítulo anterior.

3.1 Reglas del negocio

Para que las aplicaciones que usen esta biblioteca funcionen correctamente deben tener en cuenta algunas condiciones para su buen funcionamiento. Las personas deben apagar las PC antes de conectar las FL, y esta ya debe haber tener conectadas los sensores necesarios para la simulación, de forma que cuando se enciendan la computadora ya estén conectadas. La conexión debe hacerse al puerto serie que es el puerto de enlace predefinido por la FL. La FL a utilizar debe tener programado le Firmware que es el protocolo a utilizar para la comunicación, de forma que la tarjeta y la biblioteca sean compatibles.

3.2 Modelo de dominio

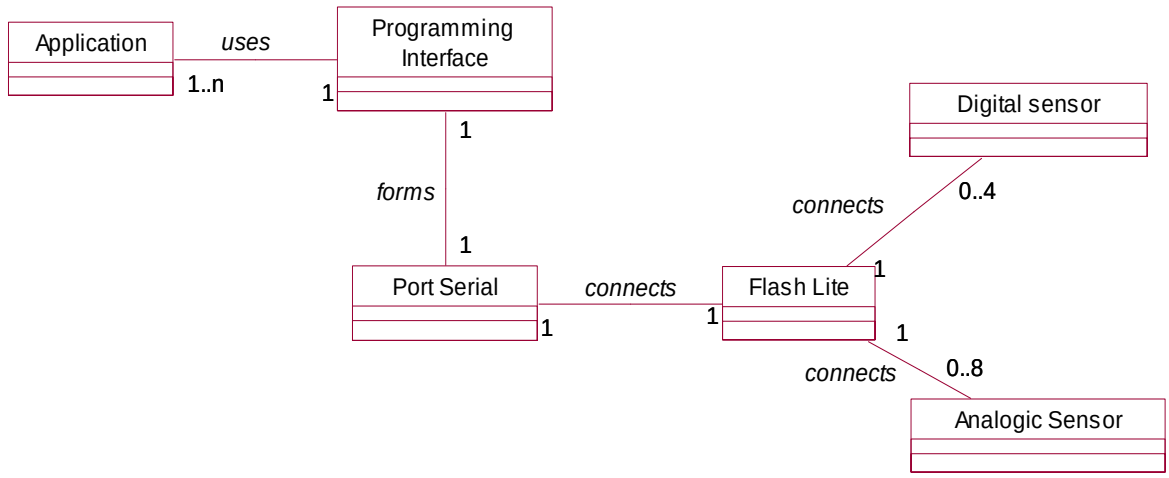

Figura 9 Modelo de Dominio.

En la Figura se ve la relación de distintos conceptos que muestran la interacción del sistema. Se tiene una Aplicación que utiliza la interfaz de programación con los sensores para el manejo de la información de los sensores que se utilicen en la simulación. Esta interfaz configura los valores que definen al puerto serie para establecer una comunicación exitosa entre la FL ya que este dispositivo inteligente esta programado par ser conectado por puerto serie. El mismo va a conectar a sus puertos sensores tanto analógicos como digitales. A través de estas clases se ve el flujo de información que iría desde los sensores hasta ala aplicación y viceversa.

3.3 Captura de Requisitos

El sistema a crear debe tener ciertas funcionalidades

3.3.1 Requisitos Funcionales

Iniciar Puerto serie
Iniciar buffers para el control del puerto.
Buscar puerto disponible en la computadora.
Verificar que un puerto serie es accesible.
Configurar puerto serie
Establecer velocidad del puerto serie
Establecer bits de parada por defecto 1
Establecer bits de datos por defecto 7
Establecer paridad por defecto no paridad.
Leer puerto
Escribir puerto
Finalizar Comunicación
Liberar el puerto.
Liberar buffer de comunicación con el puerto serie
Reiniciar todos los valores de entrada a la aplicación.
Guardar los datos en una estructura
Reiniciar todos los valores de salida.
Iniciar Comunicación
Seleccionar puerto automáticamente
Especificar velocidad
Especificar puerto
Devolver estado de comunicación
Obtener los datos de la FL
Obtener los datos de los sensores
Obtener los datos de los canales analógicos

Obtener los datos de los puertos de la FL
Recibir mensajes de la FL
Leer mensaje usando el protocolo definido.
Enviar mensaje a la FL
Crear mensaje usando el protocolo definido
Verificar autenticidad del bloque de datos
Parar la transmisión de datos
Reiniciar la transmisión.
Solicitar datos de los sensores

3.3.2 Requisitos no Funcionales del Sistema

Usabilidad: Los futuros usuarios del sistema serán programadores con conocimientos básicos de animación y de la terminología afín. El producto debe estar concebido para que el usuario piense en qué desea hacer y no cómo hacerlo, por lo que éste requerimiento debe estar presente en alto grado en el producto final.

Rendimiento: Como aplicación de tiempo real, debe tener alto grado de velocidad de procesamiento o cálculo, tiempo de respuesta y de recuperación, y disponibilidad.

Soporte: En una versión inicial deberá ser compatible con la plataforma Windows.

Legales: Se regirá por las normas ISO 9000.

Software: Sistema operativo Windows que necesiten del use de las API para hacer el control de los dispositivos.

Hardware: Compatible para usar la tarjeta de computadora Flash Lite 186, dispositivo Flash Lite, puerto serie en la primera versión, después USB.

Diseño e implementación: Se implementará en Lenguaje C++, se regirá por la filosofía de Programación Orientada a Objetos, se harán llamadas a las API.

Interfaz: Las funciones de trabajo con la Flash Lite serán sencillas, el usuario no necesita saber las características de la tarjeta ni de los sensores que a ella se conectan

3.4 Modelación de Casos de Uso del sistema

En esta sección se reconocen los posibles actores del sistema a desarrollar y se conciben, a través de la agrupación de los requisitos funcionales anteriormente hallados, los posibles resultados de valor que le pueda brindar a sus actores, o lo que es lo mismo, los casos de uso del sistema.

Además, se seleccionan los casos de uso correspondientes al primer ciclo de desarrollo para hacerles sus especificaciones textuales en formato expandido.

Actor del sistema

Actores	Justificación
El sistema que va a hacer uso de la biblioteca. (sistema usuario).	Es el que se beneficiará con las funcionalidades que brinda la herramienta esta interfaz ya que le facilitara el manejo de la información con los dispositivos de entrada y salida, haciendo uso de la FL.

Tabla 5 Descripción de los actores del sistema.

Casos de uso del sistema

Inicializar Comunicación.

Finalizar Comunicación

Parar Transmisión

Reiniciar Transmisión.

Obtener valores.

3.4.1 Diagrama de Casos de uso del sistema

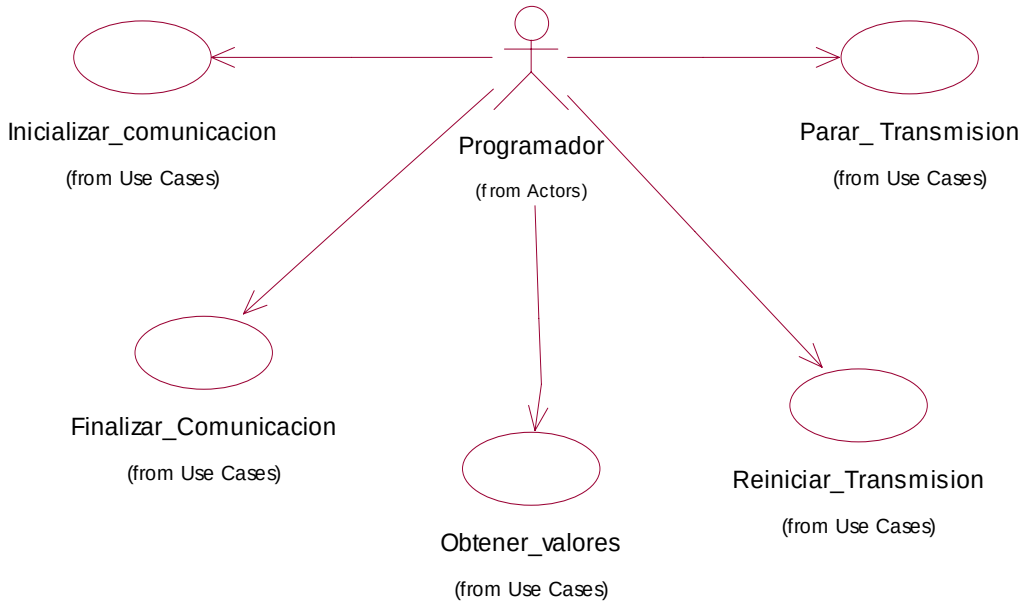

Figura 10 Diagrama de Casos de Uso del sistema

3.4.2 Especificación de los casos de uso en formato expandido

Nombre del caso de uso	**InicializarComunicacion**
Actores	Sistema usuario (inicia)
Propósito	Inicializar la comunicación entre la aplicación de realidad virtual y la FL
Resumen Se inicializa cuando el programador necesita hacer una aplicación de realidad virtual usando sensores conectados a la FL. El sistema realiza las acciones necesarias y concluye el CU.	
Precondiciones:	El programador debe reiniciar la PC una vez conectada la FL al Puerto Serie.
Referencias	R1, R5,R6,R7,R8

Curso normal de los eventos:	
Acción del actor	Respuesta del sistema
1. El sistema usuario hace uso de la función inicializar comunicación en la aplicación en la que este programando, especificando la velocidad de transmisión, y especificando o no el puerto.	
	1.1. Si se especifica el puerto, el sistema busca el puerto indicado para establecer la comunicación y verifica si está ocupado.
	1.2. Si existe el puerto y no está ocupado, abre el puerto de comunicación con la FL.
	1.3 Se modifica la estructura (DCB) para el manejo del puerto.
	1.3.1 Se configuran la velocidad de transmisión del puerto.
	1.3.2 Se configura los bits de datos con 7
	1.3.3 Se configura los bits de parada con 1
	1.3.4 Se configura la paridad
	14. Se verifica si la FL está conectada.
	1.4.1 Se solicita los datos de entrada
	1.4.2 Se solicita actualizar los dispositivos de salida en apagado.
	1.5 Retorna estado LISTO
Curso alterno de los eventos "Puerto no especificado"	
	1.1. Si no se especifica el puerto, se busca uno que no esté ocupado. Ir a 1.2 del curso normal de

	los eventos.
Curso alterno de los eventos "Puerto no disponible"	
	1.2. Si el puerto no existe, o existe y está ocupado, retorna estado NO PUERTO…
Curso alterno de los eventos "Flash Lite no conectada"	
	1.4 Si la FL no está conectada, retorna estado NO FLASH.
Prioridad	Crítico

Tabla 6 Descripción textual del CU "InicializarComunicación"

Nombre del caso de uso	**FinalizarComunicacion**
Actores	Sistema usuario (inicia)
Propósito	Una vez de que la aplicación quiera ser terminada es necesaria finalizar la comunicación para que la FL ni el puerto queden dañados.
Precondición	Debe existir una comunicación válida.
Resumen: El CU se inicia cuando el programador quiere finalizar una aplicación que utiliza la FL para el manejo de los dispositivos de entrada y salida. El sistema realiza las acciones pertinentes y termina el CU.	
Referencias	R4
Curso normal de los eventos:	
Acción del actor	Respuesta del sistema
1. El Sistema usuario hace uso de la función finalizar comunicación.	
	1.1 El sistema libera la máscara de eventos del puerto serie con la función SetcommMask de la WinApi.
	1.2 Cierra el puerto serie
	1.3 Actualiza el parámetro comunicación en falso

	1.4 Libera los buffers
	1.5 Se establece la variable estado de la comunicación en finalizada.
Prioridad	crítico

Tabla 7 Descripción textual del CU "FinalizarComunicación"

Nombre del caso de uso		**ParарTransmisión**
Actores	Sistema usuario (inicia)	
Propósito	Parar la transmisión en un momento dado en la simulación.	
Precondición	Debe existir una comunicación válida.	
Resumen: Se inicia cuando el programador considera que no le es necesario estar leyendo los datos sensados en un momento determino y decide detener la transmisión para en un momento posterior poder reiniciarla. El sistema realiza las acciones necesarias y termina el CU.		
Referencias	R14, R12,R8.R3	
Curso normal de los eventos:		
Acción del actor		Respuesta del sistema
1. El Sistema usuario hace uso de la función Parar la transmisión.		
		1.1 El sistema verifica que exista transmisión de datos
		1.1.1 El sistema envía la señal de la petición de enviar datos con la función *EscapeCommFunction*.
		1.1.2 El sistema envía la señal de los datos con la función *EscapeCommFunction,* de la *API.*
		1.1.3 Se actualiza la bandera de error del dispositivo con la función *ClearCommError* de la *API.*
		1.1.3 El sistema verifica que se pueda escribir en el

	puerto serie el comando de Parar la Transmisión
	1.1.4 Se limpia el error.
	1.2 El sistema establece la variable estado de la comunicación con el valor "parada"
	1.3 retorna estado LISTO indicando que no hay errores.
Curso alterno de los eventos "No transmisión "	
	1.1.3 si no se pueden escribir correctamente se hace la variable bcomunication en falso, se retorna el estado de NO TRANSMISION
Prioridad	crítico

Tabla 8 Descripción textual del CU "ParарTransmisión"

Nombre del caso de uso	**ReiniciarTransmisión**
Actores	Sistema usuario (inicia)
Propósito	Reiniciar la transmisión una vez de que la misma haya sido parada.
Precondición	Debe existir una comunicación válida. Y se debe haber parado el sistema con anterioridad
Resumen: Este CU se inicia una vez que el sistema usuario detiene la transmisión y desea restaurarla; el sistema hace las acciones pertinentes y finaliza el CU.	
Referencias	R8, R15, R12, R3
Curso normal de los eventos:	
Acción del actor	Respuesta del sistema
1. El Sistema usuario hace uso de la función Reiniciar la transmisión	
	1.1 El sistema transmite a la FL el comando de Solicitar Todos los Datos.

	1.1.1 El sistema escribe en el puerto serie dicho comando.
	1.2 se establece el estado de la comunicación como "reiniciado"
	1.3 El sistema retorna LISTO
Curso alterno de los eventos	
	1.1 Si no se puede transmitir el sistema retorna NO TRANSMISIÓN
Prioridad	crítico

Tabla 9 Descripción textual del CU "ReiniciarTransmisión"

Nombre del caso de uso	**Obtener valores**
Actores	Sistema usuario (inicia)
Propósito	Guardar en una estructura los valores sensados emitidos por la FL.
Resumen: Este CU se inicia cuando el sistema usuario necesita obtener los valores almacenadas en una estructura que se corresponde con los datos sensados por la FL en el momento de hacer la encuesta. El sistema realiza las operaciones pertinentes, devuelve la estructura y finaliza el CU.	
Referencias	R16, R9, R2, R11, R10, R13.
Curso normal de los eventos:	
Acción del actor	Respuesta del sistema
1. El Sistema usuario puede hacer uso hace uso de la funciones para obtener todos los datos.	
	1.1 El sistema invoca la función RxFlash
	1.1. El sistema despeja la señal de la petición de enviar

	datos con la función *EscapeCommFunction*.
	1.2 El sistema despeja la señal de los datos listos con la función *EscapeCommFunction,* de la *API.*
	1.3 Se actualiza la bandera de error del dispositivo con la función *ClearCommError* de la *API.*
	1.4 Se obtiene la cantidad de datos a leer
	1.5 Se leen los datos en el puerto serie
	1.6 Se crea un paquete con los datos leídos.
	1.7 Se verifican que este correcto el paquete de datos.
	1.8 Se actualiza la estructura SFlashdata en la clase CFlashLite
	1.9 Se actualiza un arreglo Buff con los valores emitidos
	1.9.1 Se obtiene y se guarda el valor del puerto 0 de la FL.
	1.9.2 Se obtiene y se guarda el valor del puerto 1 de la FL.
	1.9.3 Se obtiene y se guarda el valor del puerto 2 de la FL.
	1.9.4 Se obtiene y se guarda el valor del puerto 3 de la FL.
	1.9.5 Se obtiene y se guarda el valor del puerto 4 de la FL.
	1.9.6 Se obtiene y se guarda el valor del puerto 5 de la flash
	1.9.7 Se obtiene y se guarda el valor del canal analógico 0 de la FL.
	1.9.8 Se obtiene y se guarda el valor del canal analógico 1 de la FL.

	1.9.9 Se obtiene y se guarda el valor del canal analógico 1 de la FL.
	1.9.10 Se obtiene y se guarda el valor del canal analógico 2 de la FL.
	1.9.11 Se obtiene y se guarda el valor del canal analógico 3 de la FL.
	1.9.12 Se obtiene y se guarda el valor del canal analógico 4 de la flash
	1.9.13 Se obtiene y se guarda el valor del canal analógico 5 de la FL.
	1.9.14 Se obtiene y se guarda el valor del canal analógico 6 de la FL.
	1.9.15 Se obtiene y se guarda el valor del canal analógico 7 de la FL.
	1.9.16 Se obtiene y se guarda el valor del sensor 1 de la FL.
	1.9.17 Se obtiene y se guarda el valor del sensor 2 de la FL
	1.9.18 Se obtiene y se guarda el valor del sensor 3 de la FL.
	1.9.19 Se obtiene y se guarda el valor del sensor 4 de la FL.
	1.10 Se retorna el arreglo de enteros (Buff).
Curso alterno de los eventos	
	1.4 Si la cantidad de datos es 0 se retorna la estructura vacía y con el parámetro hay datos en falso se pasa al paso 1.9 para retornar los valores que hasta ese momento tenía la estructura.
Prioridad	crítico

Tabla 10 Descripción textual del CU "ObtenerValores"

Conclusiones

En el presente capítulo se definió qué es exactamente lo que espera el usuario con este sistema. Para ello quedaron establecidos los requisitos funcionales de éste, y descritos los casos de uso que le permitirán a su futuro usuario obtener los resultados esperados por el cliente.

4

Capítulo 4 Diseño e Implementación del Sistema

Introducción

En este capítulo se detallan los estándares de codificación que sirven de base para tener un buen manejo de los elementos de programación. Se describen las funcionalidades y atributos de las clases propuestas en el diseño y las relaciones entre ellas en un diagrama de clases del sistema propuesto como resultado del refinamiento de las etapas anteriores. Posteriormente se muestran los diagramas de secuencia, elaborados a partir de los casos de uso que intervienen en el desarrollo del módulo.

Se presentan además los componentes físicos, que se traducen en los ficheros .h y .cpp correspondientes a la implementación en C++. Además se elabora el diagrama de despliegue del sistema.

4.1 Estándares de codificación

El código de la herramienta sigue algunos estándares propuestos por el grupo de desarrollo (respetando los estándares de codificación para C++). Se programará en inglés, debido que las palabras son simples, no se acentúan y es un idioma muy difundido en el mundo informático.

El conocimiento de los estándares seguidos para el desarrollo de la misma permitirá un mayor entendimiento del código, y es una exigencia de los autores de la misma que cualquier módulo que se añada debe estar codificado siguiendo estos estándares.

Nombre de los ficheros:

Se nombrarán los ficheros .h y .cpp de la siguiente manera:

STKNameOfUnits.cpp

Constantes:

Las constantes se nombrarán con mayúsculas, utilizándose el "_" para separar las palabras: MY_CONST_ZERO = 0;

Tipos de datos:

Los tipos se nombrarán siguiendo el siguiente patrón:

Enumerados: enum **E**MyEnum {**ME**_VALUE, **ME**_OTHER_VALUE};

Indicando con "**E**" que es de tipo enumerado. Nótese que las primeras letras de las constantes de enumerados son las iniciales del nombre del enumerado. Véase otro ejemplo:

enum **EN**ode**T**ype {**NT**_GEOMETRYNODE,…};

Estructuras: struct **S**MyStruct {…};

Indicando con "**S**" que es una estructura. Las variables miembros de la estructura se nombrarán igual que en las clases, leer más adelante.

Clases: class **C**ClassName;

Indicando con "**C**" que es una clase

Interfaces: **I**MyInterface

Indicando con "**I**" que es una interfaz.

Listas e iteradores STD:

vector<> **T**Name**List**;

TNameList::iterator **T**Name**ListIter**;

map<> **T**Name**Map**;

TNameMap::iterator **T**Name**MapIter**;

multimap<> **T**Name**MultiMap**;

TNameMultiMap::iterator **T**Name**MultiMapIter**;

Declaración de variables:

Los nombres de las variables comenzarán con un identificado del tipo de dato al que correspondan, como se muestra a continuación. En el caso de que sean variables miembros de una clase, se le antepondrá el identificador "m_" (en minúscula), si son globales se les antepondrá la letra "g", y en caso de ser argumentos de algún método, se les antepondrá el prefijo "arg_".

Tipos simples:

bool **b**VarName;

int **i**Name;

unsigned int **ui**Name;

float **f**Name;

char **c**Name;

char* **ac**Name; // arreglo de caracteres

char* **pc**Name; // puntero a un char

char** **aac**Name; // bidimensional

char** **apc**Name; // arreglo de punteros

bool **m_b**MemberVarName; //variable miembro

char **g*C***GlobalVarName; //variable global, no se le antepone “”

short **s**Name;

Instancias de tipos creados:

EMyEnumerated **e**Name;

SMyStructure **k**Name;

CClassName **k**ObjectName;

CClassName* **pk**Name; //puntero a objeto

CClassName* **ak**Name; //arreglo de objetos

CClassName* **ak**Name; // variable miembro de clase

IMyInterface* **pI**Name; //puntero interfaces

Métodos

En el caso de los métodos, se les antepondrá el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepondrá nada.

En el caso de los argumentos se les antepone el prefijo “arg_”

Constructor y destructor:

CClassName (bool **arg**_bVarName, float& **arg**_fVarName);

~**C**ClassName ();

Funciones:

bool **b**Function1 (…);

int* **pi**Function2 (…);

CClassName* **pk**Function3 (…);

Procedimientos:

void Procedure4 (…);

Métodos de acceso a miembros

Los métodos de acceso a los miembros de las clases no se nombrarán “Gets” y “Sets”, sino como los demás métodos, pero con el nombre de la variable a la que se accede y sin “m_”:

```
int iMyVar;  //variable
```

Obtención del valor:

```
int iMyVar();
{
   return iMyVar;
}
```

Establecimiento del valor:

```
void MyVar(char* arg_iMyVar)
{
  iMyVar = arg_iMyVar;
}
```

Obtención y establecimiento del valor:

```
int& iMyVar();
{
   return iMyVar;
}
```

4.2 Descripción de las clases del diseño

Nombre: CSerial	
Tipo de clase controladora	
Atributo	Tipo
dcb	DCB
accadena	Char *
acsalida	Char *
idcomdev	handle
bcomunicacion	Bool
hsalida	HGlobal
hcadena	HGlobal
Para cada responsabilidad:	
Nombre:	*InitBuffers():void*
Descripción:	Crea espacio de memoria para los buffers que se utilizan para el control del puerto serie.
Nombre:	*ReleaseBuffers():void*
Descripción:	Libera los buffers
Nombre:	SerialStarComunication(DWord arg_velocity, int arg_iport):bool
Descripción:	Se encarga de iniciar la comunicación creando el handle de comunicación. Utiliza las opciones del puerto serie en la clase StipoOpciones para definir la estructura DCB que se corresponde con una estructura de puerto serie. Retorna la validez de la operación.
Nombre:	SerialCloseComunication ():void
Descripción:	Cierra la comunicación con el puerto serie. Se liberan los buffers. Se cierra el handle de comunicación. Retorna la validez de la operación.
Nombre:	SerialRead():DWORD

Descripción:	Lee los bytes que están en el buffer de entrada del puerto serie. Retorna los datos leídos.
Nombre:	SerialWrite(char buff):bool
Descripción:	Copia los datos del buff en el bufer de salida *acsalida*. Estos datos son escritos en el puerto serie. Retorna la validez de la operación.

Tabla 11 Descripción de la clase "CSerial"

Nombre: CFlashLite	
Tipo de clase controladora	
Atributo	Tipo
iCounterRequest	Int
bDataReceive	Bool
idesplazamiento	Int
kFlashData	SFlashData
kPortSerial	CSerial
Para cada responsabilidad:	
Nombre:	*FlashStart(DWORD arg_velocity, int arg_iport):bool*
Descripción:	Inicializa los datos de la FL almacenadas en la estructuras SFlashData. Es la encargada de llamar a la función *SerialStarComunication* de la clase *CSerial* pasándose los argumentos necesario que esta función necesita que son la velocidad de transmisión y el puerto al que se quiere conectar; retorna si la acción fue válida o no.
Nombre:	*Analogic(byte Dato, byte Dato1):int*
Descripción:	Una función auxiliar que se encarga de obtener la parte alta y la parte baja de un dato y complementarlas en un dato único.
Nombre:	*RequestFlashF(): SFlashData*
Descripción:	Verifica si se están transmitiendo datos y devuelve un objeto de

	tipo SFlashData
Nombre:	*RxFlash():SFlashData*
Descripción:	Utiliza la función SerialRead para leer los datos emitidos por la FL, los almacena en el objeto kFlashdata y retorna dicha estructura con los datos leídos.
Nombre	*TxFlash(bool comando, byte dato, Dword largeb): bool*
Descripción	Transmite un comando a la FL haciendo uso de la función SerialWrite
Nombre:	FlashEnd():void
Descripción:	Finaliza la comunicación con la FL y cierra el puerto serie llamando a la función SerialCloseComunication.
Nombre:	ParарTX():
Descripción:	Para la transmisión sin cerrar el puerto serie. Envía a la FL un comando usando la función TXFlash (0, 0,4).

Tabla 12 Descripción de la clase "CFlashLite"

Nombre: CFlashComunication	
Tipo de clase: interfaz de programación	
Atributo	Tipo
aiBuff[19]	int *
aiEncd[4]	Int *
aiPorts[6]	Int *
aiChan[8]	Int *
state	Ecomunication
pkFlashLite	CFlashLite
Para cada responsabilidad:	
Nombre:	*BeginComunication*():bool
Descripción:	Inicia la comunicación en la aplicación
Nombre:	*EndComunicacion*()

Descripción:	Termina la comunicación en la aplicación
Nombre:	StopTX():bool
Descripción:	Para la comunicación
Nombre:	*RestartTX():bool*
Descripción:	Reinicia la comunicación
Nombre	*AllData():int* *
Descripción	Devuelve todo los datos que se están emitiendo.
Nombre:	*EncoderData():int**
Descripción:	Devuelve los datos de los sensores.
Nombre:	*PortData():int* *
Descripción:	Devuelve los puertos configurados en la FL.
Nombre:	*AnalogicChanelData(): char**
Descripción:	Devuelve los datos de los sensores analógicos conectados a la FL.
Nombre:	*GetCounterRequest(): int*
Descripción:	Devuelve el valor del atributo *CounterRequest* perteneciente a la clase CFlashLite.
Nombre:	*SetCounterRequest(int arg_value):void*
Descripción:	Sobrescribe el valor del atributo CounterRequest con el del argumento de la función.
Nombre:	*RequestFlashData():bool*
Descripción:	Devuelve el valor del parámetro bRequestFlashdata del objeto kFlashdata de la clase CFlashLite.

Tabla 13 Descripción de la clase "CFlashComunication"

Nombre: SCommOptions	
Tipo de clase entidad (estructura)	
Atributo	Tipo
cPorts[5]	Char

ibaudios	Int
ibitsdata	Int
ibitsstop	Int
cParity[25]	Char

Tabla 14 Descripción de la estructura "StipoOpciones"

Nombre: SFlashData	
Tipo de clase entidad (estructura)	
Atributo	Tipo
bRequestFlashData	Bool
bError	Bool
coderror	Byte
In0	Byte
In1	Byte
In2	Byte
In3	Byte
In4	Byte
In5	Byte
In6	Byte
In7	Byte
Ch0	Int
Ch1	Int
Ch2	Int
Ch3	Int
Ch4	Int
Ch5	Int
Ch6	Int
Ch7	Int
S1	Int

S2	Int
S3	Int
S4	Int

Tabla 15 Descripción de la estructura "SFlashData"

4.2.1 Diagrama de clases

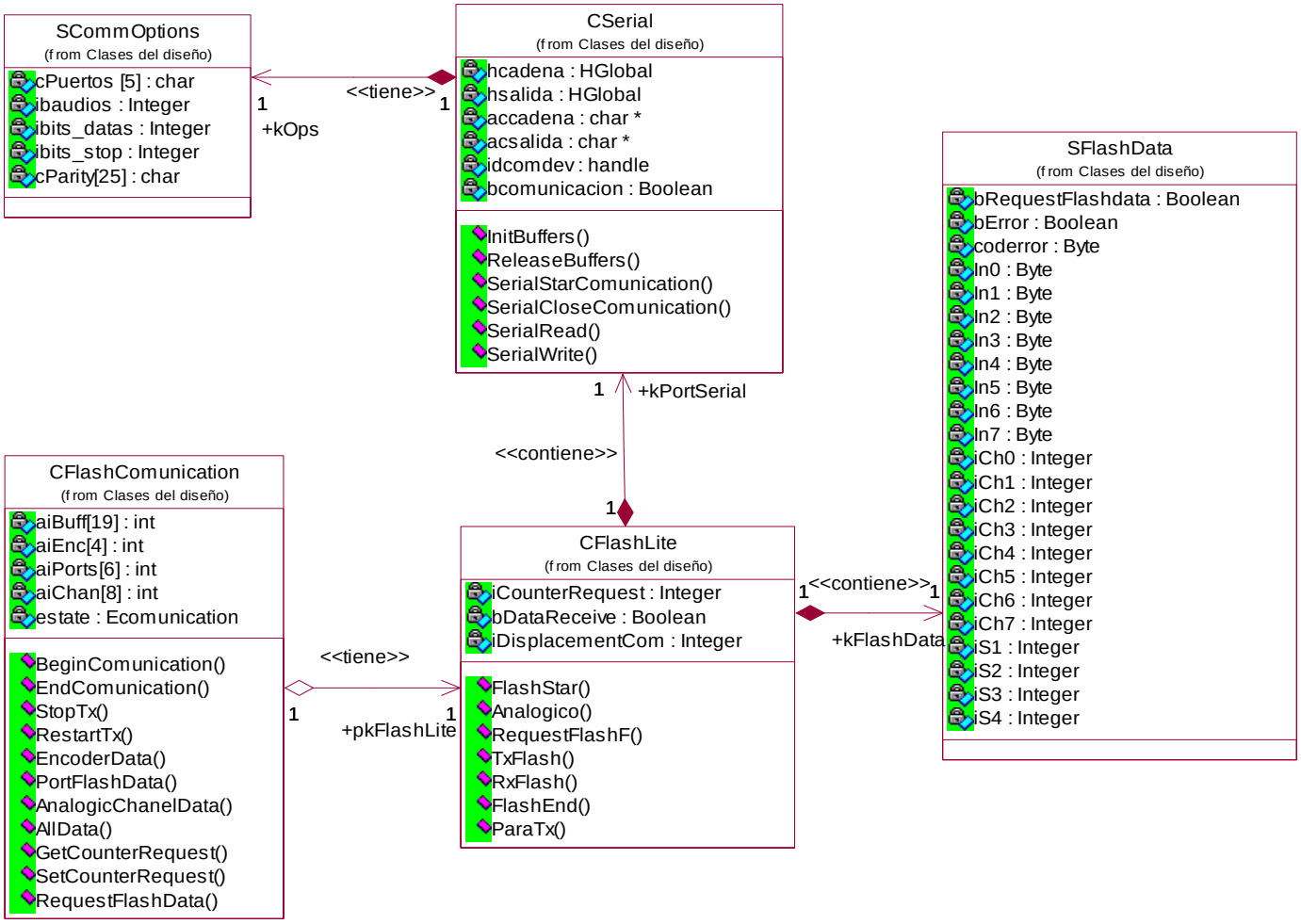

Figura 11 Diagrama de clases del diseño

4.2.2 Diagramas de interacción

Los diagramas de interacción muestran como interactúan las clases para dar cumplimiento a los CU en tiempo de ejecución y organización. En el caso de los diagramas de secuencias representan la interacción en el tiempo.

CU Inicializar Comunicación

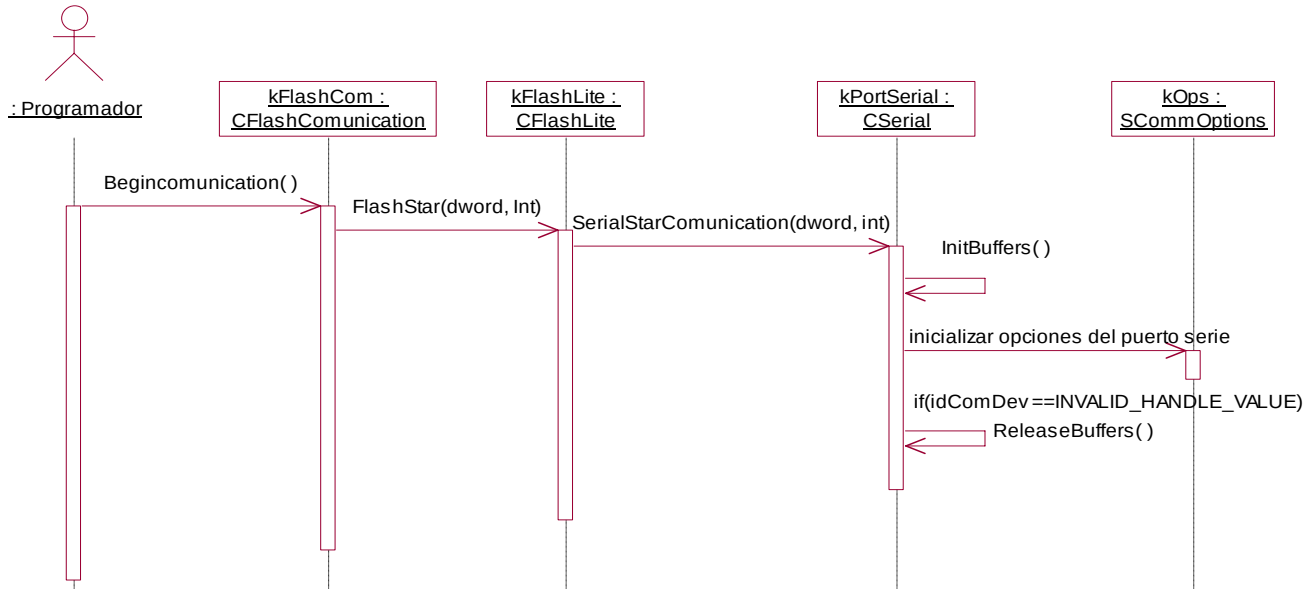

Figura 12 Diagrama de secuencia CU "Inicializar_Comunicación"

CU Finalizar Comunicación

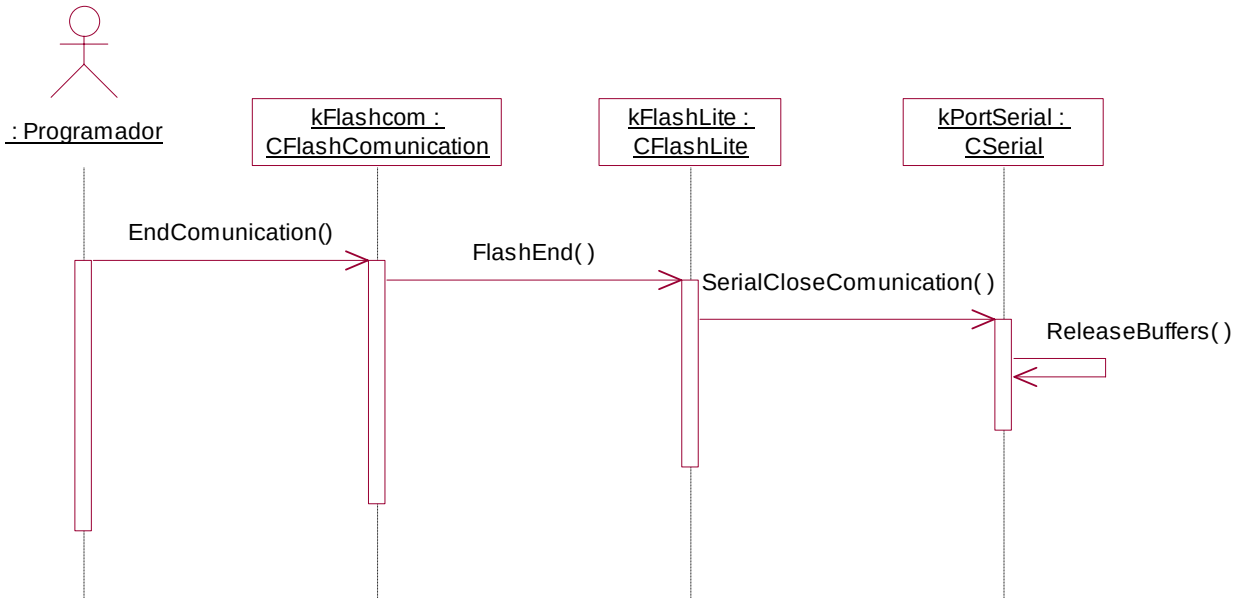

Figura 13 Diagrama de secuencia CU "Finalizar_Comunicación"

CU Detener Transmisión

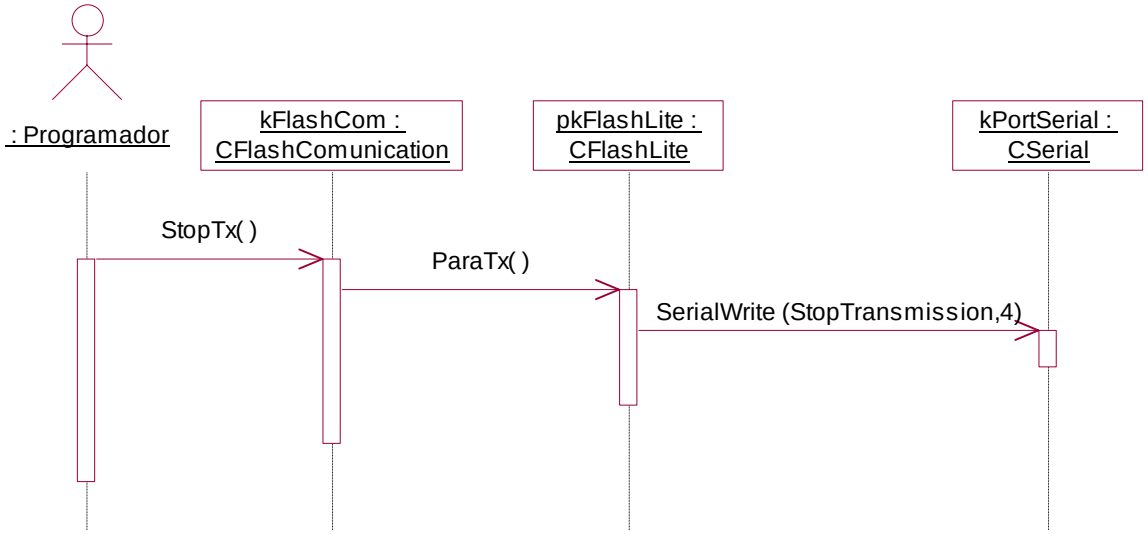

Figura 14 Diagrama de secuencia de CU "Parar Transmisión"

CU Reiniciar Transmisión

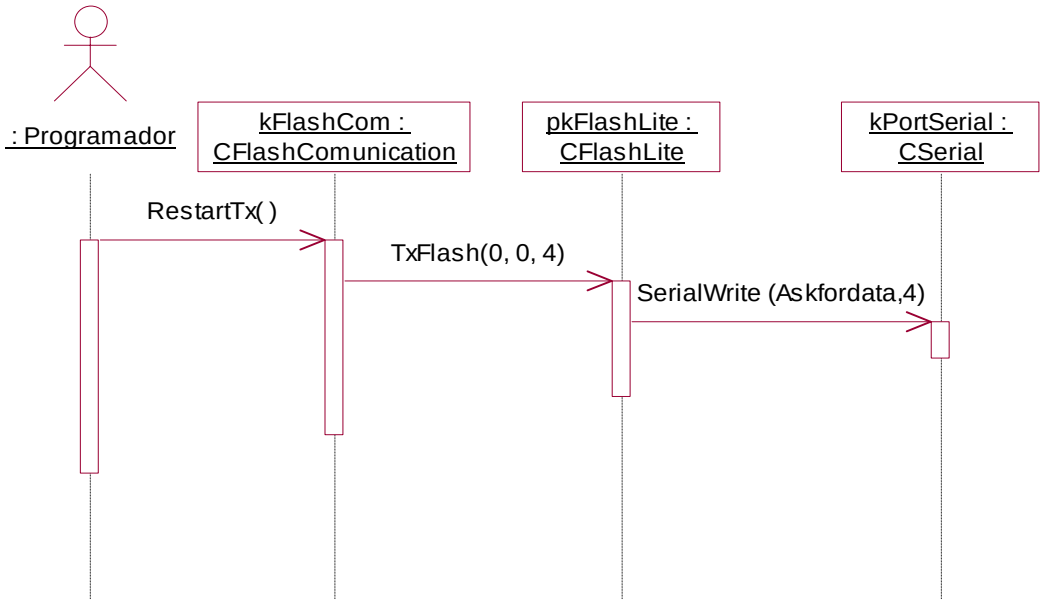

Figura 15 Diagrama de secuencia de CU "Reiniciar_Transmision"

CU Obtener estructura

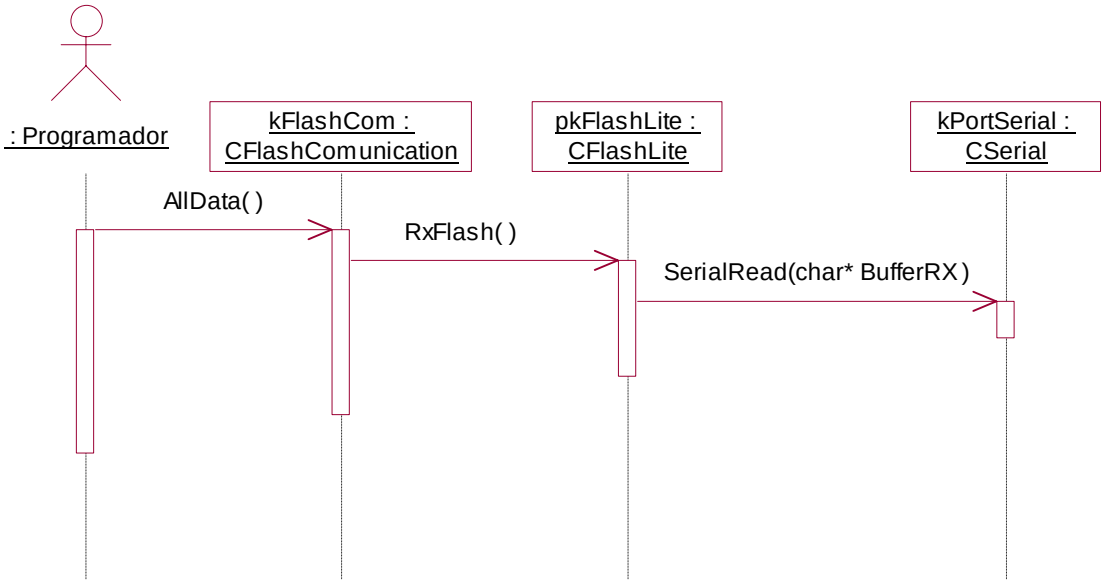

Figura 16 Diagrama de secuencia de CU "Obtener_valores".

4.3 Diagrama de despliegue

El producto final estará distribuido de la siguiente forma en una PC *Aplicación* donde estará corriendo el programa principal y donde se hará uso de esta biblioteca que necesita que se use el recurso FL a la cual estarán conectados todo tipo de sensores.

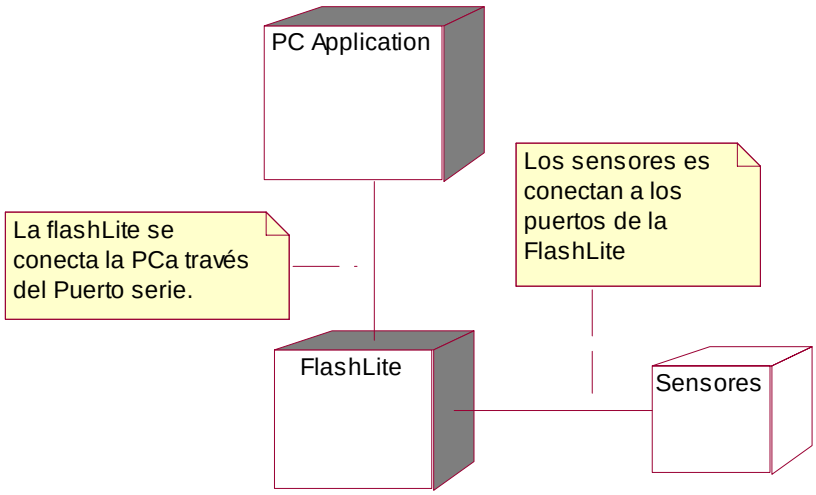

Figura 17 Diagrama de despliegue

4.4 Diagrama de componentes

Como se programó en el lenguaje VC++ las clases van a estar definidas en un fichero .h y su implementación en un fichero .cpp

Para la programación de esta interfaz se hace el uso de tres ficheros principales que contienen la implementación de sus clases correspondientes.

Se tiene un fichero “STKFlashCom.h” que definen a la clase CFlashComunication. Este fichero incluye al “STKFlash.h” que define a la clase CFlashLite y este a su vez hace uso de la clase CSerial definida en el fichero “STKSerial”.

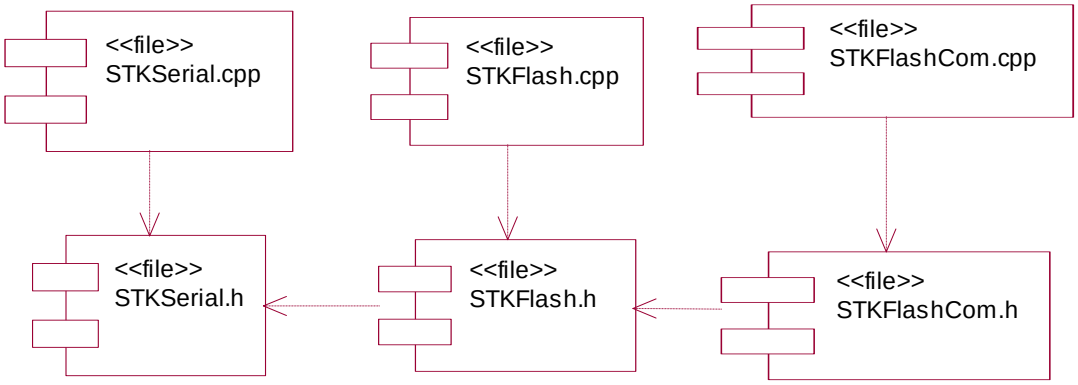

Figura 18 Diagrama de componentes

Conclusiones

Al concluir este capítulo se tiene concebido detalladamente el diseño completo del sistema y la secuenciación de pasos traducida a mensajes entre clases de los casos de usos a desarrollar.

Se tiene concluida la etapa de implementación, por lo que en este momento se encuentra todo preparado para pasar a la etapa de programación de los casos de uso desarrollados en el primer ciclo. Como posibilidad adicional que brinda la herramienta *Rational Rose*, ya es posible generar el código fuente de los componentes relacionados con los casos de uso a desarrollar.

Conclusiones

Para el cumplimiento de los objetivos de este proyecto, se realizó primeramente un estudio de las técnicas, tecnologías y tendencias actuales en relación con el tema. En el estudio se analizaron las características para desarrollar este tipo de aplicaciones. Además, a partir de esa investigación se proponen las características técnicas de la solución.

Posteriormente se realizó la captura de los requisitos funcionales y no funcionales, de la agrupación de dichos requisitos surgieron los primeros casos de uso del sistema, donde se realizó una descripción de cada uno.

A continuación se transitó por las etapas de análisis, diseño e implementación utilizando los artefactos de RUP, donde surgieron y maduraron las clases, se realizaron los casos de uso a desarrollar en la primera fase y se creó el diagrama de componentes final que contendrá a las clases de la aplicación.

Finalmente se implementó una interfaz que les facilita a los programadores de SRV interactuar con la tarjeta inteligente “Flash Lite”.

Recomendaciones

Para que la Interfaz para el manejo de los dispositivos de E/S en Sistemas de Realidad Virtual brinde mayores facilidades de uso y servicios se recomienda:

- Implementar la biblioteca de forma que pueda correr en otros Sistemas Operativos.
- Implementar comunicación para puerto USB.
- Implementar el resto de los comandos que permitan el envío automático de los datos.

Referencias bibliográficas

1. CHOVER, Miguel.
   Introducción a la Informática Gráfica.
   Capítulo 9. Aplicaciones de la Informática Gráfica.

2. GALEANO, Javier.
   La Realidad Virtual.
   Universidad de los Teques "Cecilio Acosta".
   Dpto. de Arquitectura del computador.
   Abril 2002.

3. EJARQUE, Cristina IBAÑEZ, Laura.
   Realidad Virtual.

4. Glosario.
   Disponible en: http://www.control-systems.net/recursos/glosario/s.htm

5. *Just what is "simulation" anyway?*
   University of Central Florida. Institute for simulation &training. Disponible en
   http://www.ist.ucf.edu/background.htm

6. CARBONELLAS, José María, MERA, José Manuel. Tecnología de simuladores.
   6.1. Capítulo 6. Arquitectura de simuladores.
   6.2. Capítulo 9. Tecnología para tiempo real.

7. Página Web disponible en: http://ciberhabit.gob.mx/parque/para-jugar/simulacion.htm

8. N. GOMEZ, M CAÑIZARES, O. MORA, M de la PARTE, A. GUIA, JA. RDGUEZ.
   Utilización de la API Win 32 para el control de la comunicación serie en sistemas para prueba de esfuerzo.
   II Congreso Latinoamericano de Ingeniería Biomédica,
   Mayo 2001, La Habana, Cuba.

9. JK microsystems, Inc.
Disponible en: http://www.jkmicro.com

10. CORRADO, Ericka, DELGADO, Julio, CASTAÑEDA, Salvador.
Tecnologías de la Realidad Virtual. Modelo Edificio Inteligente. Parte 3. Tecnología.
Centro de Investigación Científica y de Educación Superior de Ensenada (cicese).
Dpto. de Cómputo Dirección de Telemática.
Ensenada, B. C., México. Febrero 2001.

11. Sense8 CORPORATION TEAM.
World Toolkit Release 9, Reference Manual.
Sense8 Corporation (www.sense8.com). USA. 1999.
11.1. Capítulo 13 Sensors.
11.2. Capítulo 23 Serial Ports.
11.3. Capítulo 1 Introduction to World Toolkit

12. Api de Windows.
Disponible en: http://es.wikipedia.org/wiki/API_de_Windows

13. WinApi Conclase.
Disponible en: http://winapi.conclase.net

Bibliografía consultada

- Iván Santelice Malfante.
  La Realidad virtual y sus impactos en la industria moderna.
  Dpto. de Ingeniería Industrial, Universidad del Bio-Bio, Chile.

- Guillermo Perez Trabado.
  Sistemas Informáticos de Tiempo Real.
  Dpto. de Arquitectura del computador. Universidad de Málaga.

- Jesús Alain Fernández.
  Adquisición de datos a través del puerto serie RS232 del computador.
  Universidad Central de las Villas, Ingeniería Automática. 2002.

- MSDN 2003

Sitios Web

http://gsyc.escet.urjc.es/docencia/asignaturas/robotica/transpas/node3.html

http://www.telepolis.com/cgi-bin/web/DISTRITODOCVIEW?url=/mundopc/doc/Juegos/Simulacion.htm

http://www.telepolis.com/cgi-bin/web/DISTRITODOCVIEW?url=/mundopc/doc/Juegos/joystick.htm

http://www.superrobotica.com/sensors.htm

http://www.cmkrnl.com/faq01.html

Redes de sensores sin cables
Disponible en: http://www.euroresidentes.com/Blogs/avances_tecnologicos/2004/06/redes-de-sensores-sin-cable.htm

Glosario de abreviaturas

API: *Application Programmer's Interface,* (Interfaz de Programación de Aplicaciones).

CRC: Bit de control de error

CU: Caso de uso.

E/S: Entrada salida

FL: Flash Lite. Tarjeta de computadora con un procesador que permite realizar el procesamiento de los datos emitidos por los sensores.

HW: *Hardware*.

MM: Modelo Matemático

RX: Recepcionar

SRV: Sistemas de Realidad Virtual.

SW: Software.

TX: Transmitir

WTK: *World ToolKit*. Entorno para el desarrollo de aplicaciones de Realidad Virtual.

Glosario de términos

A:

Analógicos: Magnitudes o valores que varían con el tiempo en forma continua, pueden variar muy lento o muy rápido.

Analógica: Información presentada de manera secuencial y continua.

Aplicación: Cualquier programa que corra en un sistema operativo y que haga una función específica para un usuario.

*****

B:

Bits*:* Dígito Binario. Unidad mínima de almacenamiento de la información cuyo valor puede ser 0 ó 1 (falso o verdadero respectivamente).

Buffer*:* En informática, un buffer de datos es una ubicación de la memoria en una computadora o en un instrumento digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada.

Bytes: Conjunto de 8 bits. Puesto que 8 bits es la mínima cantidad requerida para representar los símbolos alfanuméricos

*****

C*:*

C++: Lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos (POO). C++ está considerado por muchos como uno de los lenguajes más potentes, debido a que permite trabajar tanto a alto como a bajo nivel

Cibernautas: Este nombre se aplica a los intrépidos exploradores que viajan por el ciberespacio en busca de información.

D:

Dispositivos: Son estructuras sólidas, electrónicas y mecánicas las cuales son diseñadas para un uso especifico, estos se conectan entre sí para crear una conexión en común y obtener los resultados esperados siempre y cuando cumplan con las reglas de configuración.

Driver: Controlador que permite gestionar los periféricos que están conectados al ordenador.

*****

E:

Estereoscopía*:* apreciación de las diferentes distancias y volúmenes en el entorno dando sensación de profundidad, lejanía o cercanía de los objetos. Lograr que los dos ojos vean lo que según su ángulo le corresponda.

Entorno*:* Mundo

*****

F:

FlashLite: Dispositivo inteligente**.**

*****

H:

Hardware: Conjunto de componentes materiales de un sistema informático. Cada una de las partes físicas que forman un ordenador, incluidos sus periféricos.

Handler: Manipulador. Rutina de software que realiza una determinada tarea. Por ejemplo, cuando se detecta un error, se llama a un manipulador de error para recuperarse de esa condición.

*****

I:

Inmersión: Un sistema es inmersivo cuando el usuario logra sentirse parte de el.

Interactivo: Un sistema es interactivo cuando permite un diálogo continuo entre el usuario y la aplicación, respondiendo ésta a las órdenes de aquel.

Interfaz: Zona de contacto o conexión entre dos componentes de "hardware"; entre dos aplicaciones; o entre un usuario y una aplicación. Apariencia externa de una aplicación informática.

*****

J:

Joystick: Dispositivos de entrada utilizada principalmente en juegos virtuales.

*****

L:

Librerías: En Inglés library. Cuando se habla de ordenadores, se refiere al conjunto de rutinas que realizan las operaciones usualmente requeridas por los programas. Las librerías pueden ser compartidas, lo que quiere decir que las rutinas de la librería residen en un fichero distinto de los programas que las utilizan. Los programas enlazados con bibliotecas compartidas no funcionarán a menos que se instalen las bibliotecas o librerías necesarias.

*****

M:

Modelo matemático: Conjunto de leyes físicas y matemáticas que rigen el funcionamiento de la simulación.

Multiplataforma*:* Multiplataforma es un término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

Mundo* virtual*: Puede considerarse como el paradigma que actualmente rige la construcción de modelos basados en Realidad Virtual. Proponiendo un cambio fundamental en la naturaleza de la denominada interfaz usuario (que rige la orientación de la interacción hombre-máquina), desplazándola hacia un diseño centrado en lo humano, según el cual el espacio alrededor del usuario se constituye en el ambiente de computación y una entera gama de percepciones sensoriales se conjuga en torno a la nueva interfaz. Todo esto se traduce en un esfuerzo por hacer que la tecnología de computación se haga más amistosa y accesible al usuario enfocándose hacia un planteamiento básico: si algo puede ser representado sensorialmente, es posible incorporarlo al medio computarizado.

*****

N:

Nodo: Es el elemento mas pequeño en una escena gráfica, es usado para construir una escena gráfica por arreglos en una jerarquía

*****

O:

Overlapped: Solapada

*****

P:

Paquete*:* Un paquete es un pedazo de información enviada a través del puerto serie La unidad de datos que se envía a través del puerto serie la cual se compone de un conjunto de bits que viajan juntos.

Programación *Orientada a Objetos:* POO es una filosofía de programación que se basa en la utilización de objetos. El objetivo de la POO es "imponer" una serie de normas de desarrollo que aseguren y faciliten el mantenimiento y reusabilidad del código.

Puerto: Circuito electrónico que gobierna la conexión entre dos dispositivos de hardware y los ayuda a intercambiar información de manera confiable.

Puerto *Serie:* Elemento hardware que permite el flujo de información en una sola línea de comunicación. El puerto serie es un medio sencillo de conectar entre sí dos aparatos electrónicos mediante un cable. En los ordenadores convencionales, podemos encontraras habitualmente dos de estos puertos. A través de este puerto, podemos conectar distintos dispositivos, un módem o un ratón.

*****

R:

Realidad Virtual: Realidad imaginaria.

*****

S:

Sensores: Dispositivo que detecta, o sensa manifestaciones de cualidades o fenómenos físicos

Sistema *de Realidad virtual:* sistema informático interactivo que ofrece una percepción sensorial al usuario de un mundo tridimensional sintético que suplanta al real

Simulador: Aparato que permite la simulación de un sistema, reproduciendo su comportamiento. Los simuladores reproducen sensaciones que en realidad no están sucediendo

Simulación: Recreación de procesos que se dan en la realidad mediante la construcción de modelos que resultan del desarrollo de ciertas aplicaciones específicas. Los programas de simulación están muy extendidos y tienen capacidades variadas, desde sencillos juegos de ordenador hasta potentes aplicaciones que permiten la experimentación industrial sin necesidad de grandes y onerosas estructuras;

Software: blando-duro, en referencia a la intangibilidad de los programas y corporeidad de la máquina. Software es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador.

*****

T:

Tiempo real: Se dice que un ordenador trabaja en tiempo real cuando realiza una transacción que le ha sido ordenada desde una terminal en ese mismo momento, sin espera alguna

Transductor: Dispositivo que recibe la potencia de un sistema mecánico, electromagnético o acústico y lo transmite a otro, generalmente en forma distinta. El micrófono y el altavoz son ejemplos de transductores. En comunicaciones (informática) es un transmisor/receptor de señales de radio frecuencia (RF), sirve para conectar aparatos por vía inalámbrica.

*****

U:

Usuario: Es el que recibe los estímulos de parte del sistema y a su vez se encarga de retroalimentarlo y definir su comportamiento.

*****

W:

WorldToolkit: Es un API 3D comercial que consiste de un conjunto de rutinas en C que permite a los desarrolladores construir simulaciones 3D y aplicaciones de RV.

Índice de figuras y tablas