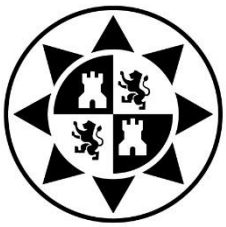
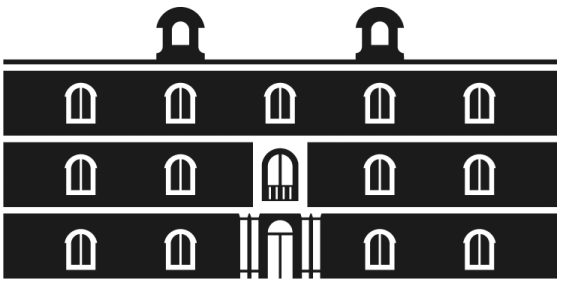


Universidad
Politécnica
de Cartagena



industriales
etsii UPCT

“Simulación a través de Matlab de los movimientos de un Cyberglove”

Titulación: Ingeniería Técnica Industrial
Especialidad: Electrónica Industrial
Alumno/a: John Stalin Briceño Díaz
Director/a/s: Jorge Juan Feliu Batlle
Carlos Alberto Díaz Hernández

Cartagena, 24 de febrero de 2015

Índice General

Capítulo 0
Introducción

El auge de la realidad virtual ha estado precedido de un largo tiempo de intensa investigación. En la actualidad, la realidad virtual se plasma en una multiplicidad de sistemas que permiten que el usuario experimente artificialmente múltiples áreas en el campo de la medicina, ingeniería, aeronavegación, diseño, psicología, etc.

Para el desarrollo de este proyecto presenta como resultado, la fusión de dos componentes como son: el hardware y el software, reunidos en el denominado "DT5 Data Glove" empleado como periférico de entrada para trabajar en un espacio tridimensional, de esta manera no nos limitamos a trabajar sobre una superficie plana, sino que logramos emular el movimiento de la mano en tres dimensiones, por medio de un conjunto de sensores de flexión de fibra óptica que permite localizar su posición.

Para la interacción del usuario con el ordenador se utilizarán 16 sensores de contacto colocados en partes estratégicas de los dedos del guante, esta interacción se logra mediante hardware: comunicación RS-232 a través del puerto serie, y mediante software: utilizando el protocolo de red TCP/IP. El desarrollo está basado en las herramientas Matlab-Simulink y Microsoft Visual C++.

0.1 Motivación

En la actualidad, la realidad virtual se plasma en una multiplicidad de sistemas. Profesionales en campos como la medicina, economía, exploración espacial, industria de los videojuegos, etc.; utilizan los sistemas virtuales para una gran variedad de funciones. Los cirujanos pueden realizar operaciones simuladas para ensayar las técnicas más complicadas, antes de una operación real. Los arquitectos pueden hacer que sus clientes, enfundados en cascos y guantes, visiten los pisos-piloto en un mundo de Realidad Virtual. Por otra parte, permite la anticipación de errores de diseño y la recreación de experiencias físicas más cercanas a la realidad.

Así, en este Proyecto se trata de estudiar cómo construir estos sistemas virtuales y analizar su funcionamiento, además de acercarnos más a los dispositivos periféricos de entrada que realizan el seguimiento del usuario e interactúan con el entorno virtual.

Además nos introduciremos en redes de comunicación TCP/IP, analizaremos como se provee conectividad de extremo a extremo, especificando la manera en que los datos deberán ser enviados y recibidos por el destinatario.

0.2 Problemática

Se desea desarrollar un entorno virtual que emule el movimiento de la mano en tiempo real y su procesamiento de datos.

Para ello, utilizaremos un guante de realidad virtual DT5 Data Glove haciendo una adquisición y filtrado de datos, utilizando los datos de las fuerzas ejercida por los dedos lo trasladaremos hacia nuestro entorno virtual creado en Simulink (VRML).

Se utilizará la comunicación del puerto serie RS-232 para recibir datos del guante y el protocolo TCP/IP para la comunicación entre Matlab y el software del guante.

0.3 Objetivos

- Desarrollar una aplicación para gestionar la comunicación TCP/IP y la comunicación con el guante

Para este apartado se utilizará la programación en Microsoft Visual C++, se estudiará la forma más simple de conectar los datos del guante a la interfaz de comunicación TCP/IP y se diseñará una aplicación-servidor con varios subprocesos (hilos de trabajo) que controlarán la lectura de los sensores del guante, las comunicaciones y el envío de datos hacia Matlab.

- Desarrollar un bloque (Función-S de nivel 2) en Simulink para recibir los datos del servidor y la creación de entorno virtual con su respectivo control.

Éste es el principal objetivo de este proyecto. Primero se desarrollará un bloque Función-S de nivel 2 que hará la función de recibir los datos y mostrarlos en un formato de salida. Segundo se creará el entorno virtual de la mano en formato 3D. Tercero se creará el control del entorno virtual y la simulación del mismo.

0.4 Definición de realidad virtual

"Realidad virtual: un sistema de computación usado para crear un mundo artificial en el cual el usuario tiene la impresión de estar y la habilidad de navegar y manipular objetos en él". Manetta C. y R. Blade (1995)

"La realidad virtual te permite explorar un mundo generado por computadoras a través de tu presencia en él". Hodder y Stoughton(s/a).

"La realidad virtual es un camino que tienen los humanos para visualizar, manipular e interactuar con computadoras y con información extremadamente compleja". Aukstakalnis (1992)

En el marco de este trabajo, nosotros definiremos a la Realidad Virtual como un sistema tecnológico que pretende simular las percepciones sensoriales de forma que el usuario las tome como reales. Para ello, se define lo virtual como algo que percibimos pero que no se corresponde con la realidad en ese espacio-tiempo (espejismo, grabación virtual, película, etc.). Si queremos que un usuario perciba algo virtual como real necesitaremos una interfaz que lo simule en tiempo real y le permita interaccionar con él a través de múltiples canales sensoriales (visión, audición, tacto, olor, gusto).

El objetivo último de la realidad virtual es crear, almacenar y simular un mundo alternativo, modelar objetos en él, definir relaciones entre ellos y la forma en la que interaccionan, para que el usuario pueda más tarde percibirlo.

Capítulo 1
Dispositivos de realidad virtual y aplicaciones

1.1 Estado del arte

Un guante electrónico de datos es un dispositivo interactivo que está equipado con sensores electrónicos, cuya finalidad es la de servir como periférico de entrada, principalmente en entornos de realidad virtual para controlar movimientos en un entorno tridimensional, los guantes son equipados con sensores de tacto o sensibilidad. Cabe mencionar que un guante de datos no permite "tocar' el mundo virtual, sino que provoca la sensación al usuario de que se encuentra en este entorno tridimensional (inmersión).

El empleo del guante permite al usuario una mejor interacción con el sistema ya que posee sensores que miden su posición y orientación.

El primer guante de datos fue desarrollado en los laboratorios Bell de AT&T por el Dr. G. Grimes, denominado "Digital Data Entry Glove"; equipado con sensores de flexión en los dedos, sensores táctiles en la parte de la yema de los dedos y con sensores de posicionamiento espacial.

1.2 Guantes de realidad virtual

Actualmente en el mercado hay otros dispositivos parecidos al 5DT Data Glove que permiten la lectura del movimiento de una o ambas manos.

- **DG5 VHAND 2.0**

Es un equipo completo, equipado con 5 sensores, miden el movimiento de los dedos, mientras que el acelerómetro detecta los movimientos y orientación de la mano. La transmisión se realiza mediante protocolo estándar RS-232 con 115200 6PS. Equipada con una batería de 20mA, necesita un voltaje de entrada de 3.3V a 5V. Sus usos están dirigidos hacia: robótica, realidad virtual, juegos. El costo en el mercado es de aproximadamente USD 992.

Figura 1. DG5 VHAND 2.0.

- **DATA GLOVE 5DT**

Fabricado en licra y diseñado para la animación en tiempo real y captura de movimiento, alta tasa de transmisión de datos e igual calidad de los mismos. Un sensor en cada dedo para medir su flexión, se comunica con la PC vía USB. Dispone de conexión con el puerto RS-232 independiente de la plataforma. Ofrece una resolución de 8-bits en la flexión. Su costo en el mercado es de alrededor de USD 1,425.

Figura 2. Data Glove 5DT.

- **DT DATA GLOVE 5 MRI**

La serie RMI de guante de datos 5DT está destinado para ambientes de proyección de imagen de resonancia magnética (MRI). El guante se comunica por medio de una caja de control vía fibra óptica; esta caja se conecta con la PC mediante un cable que está conectado al puerto serie RS-232 (independiente de la plataforma). Existen modelos de 5 y 14 sensores. Tiene un costo de USD 4,177.

Figura 3. 5DT Data Glove 5 MRL.

- **CYBER GLOVE II**

Cuenta con 18 sensores: plegado, captura, medición, arqueo de la mano, flexión y captura de la muñeca. El sistema de captura de movimiento es utilizado en aplicaciones reales como realidad virtual biomecánica y animación. Posee un software programable para el conmutador sobre la muñeca que agrega capacidad adicional de entrada y salida. Tiene un costo de USD 12,295.

Figura 4. Cyber Glove II.

- **CYBERGRASP EXOSKELETON**

Éste es un dispositivo tipo exoesqueleto, el cual se coloca sobre el CyberGlobe. Este sistema permite al usuario sentir el tamaño y la forma de los objetos 3D, es decir permite literalmente sentir el objeto manipulado por medio de una red de tendones conectados a las yemas de los dedos a través del exoesqueleto.

Figura 5. Cybergrasp Exoskeleton.

- **ACCELEGLOYE**

Éste es un guante de datos de código abierto, programable que registra los movimientos de la mano y de los dedos. Software programable en Java. Anthro Tronix desarrolló inicialmente el guante junto al Departamento de Defensa de los EEUU, para funciones relacionadas con el control robótico. Bajo la punta de cada dedo existe un acelerómetro, los cuales detectan la orientación tridimensional de los dedos y la palma con relación a la gravedad terrestre.

Figura 6. Acceleglove.

1.3 Otros dispositivos de captación de movimiento de las manos

- **Leap Motion Controller**

Leap Motion controller es un pequeño dispositivo periférico que está diseñado para ser colocado sobre un escritorio físico, idealmente mirando hacia arriba. Éste posee dos cámaras y tres LEDs infrarrojos y está básicamente orientado a permitir la detección de manos, dedos, herramientas de ciertas características y gestos realizados con éstas de forma muy precisa. El alcance espacial es de aproximadamente 0,01 mm, con un FOV ("field of view") de 115° centrado en el dispositivo, el cual forma una pirámide invertida.

Leap ha distribuido de forma gratuita múltiples unidades alrededor del mundo para los desarrolladores interesados en hacer uso del mismo. Las mismas son denominadas "developer units" y poseen algunas restricciones con respecto a las versiones comerciales del mismo. Las unidades comercializadas actualmente tienen un precio de USD 7,999.

En primer lugar, destacamos que el espacio de observación que abarca Leap es más pequeño pero a la vez de mucho mayor resolución, que otros dispositivos de mercado. El reconocimiento de los dedos de las manos proporciona precisión para los gestos efectuados con ellas.

Así, básicamente lo que se propone el dispositivo Leap es poder lograr precisión con los gestos de los usuarios realizados con las manos (interacción multitáctil, reconocimiento y tracking de usuarios, gestos que no requieran demasiada precisión, etc.). Es por ello que una buena ubicación dentro de una oficina del futuro sería empotrar el dispositivo Leap Motion Controller en la mesa, frente a donde cada participante estaría ubicado.

Por otro lado, Leap cuenta actualmente con una comunidad muy activa, donde varios desarrolladores se han mostrado muy interesados en desarrollar aplicaciones que saquen jugo de este dispositivo. El equipo de Leap Motion ha proporcionado SDKs para una gran variedad de plataformas y lenguajes de programación, entre ellos para C/C++, el cual cuenta con portabilidad para Mac así como también para Windows; aún resta portabilidad para Linux pero es un punto en el cual se encuentran trabajando. El desarrollo del SDK de Leap se viene expandiendo mediante la liberación de nuevas versiones de forma ágil. Por ejemplo, recientemente se liberó una nueva versión que incluye reconocimiento de gestos básicos, lo cual indica que el SDK va en crecimiento en cuanto a la funcionalidad brindada.

- Sistema de Coordenadas

Leap Motion Controller maneja un sistema de coordenadas cartesiano centrado en el dispositivo y los valores reportados por el mismo son en milímetros. Los ejes se pueden ver en la Figura 7, donde el eje X y Z conforman un plano horizontal siendo el eje X

paralelo al lado más largo del dispositivo. Por otro lado, el eje Y es perpendicular al dispositivo considerando los valores positivos como aquellos por encima del mismo, mientras que los valores en Z positivos son aquellos por delante del dispositivo.

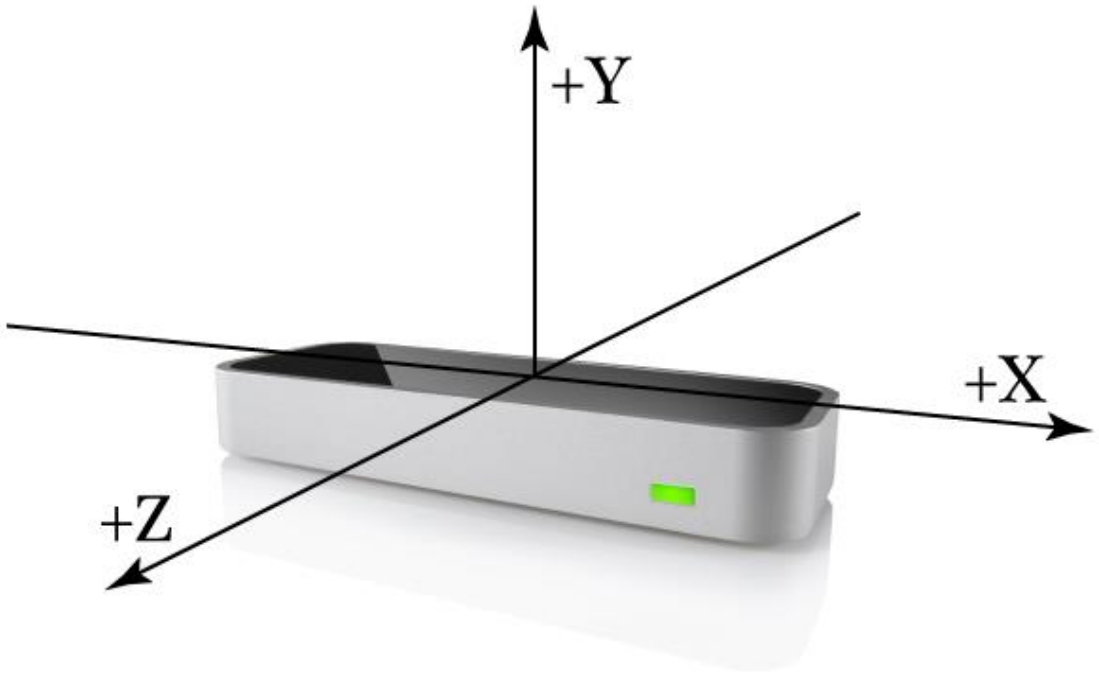


Figura 7. Sistema de coordenadas de Leap Motion Controller.

- Tracking Data

Leap sensa la información capturada detectando dedos, manos, herramientas con ciertas características y gestos a los cuales les asocia un identificador único mientras sigan siendo visibles en el FOV del dispositivo. Si alguno de estos objetos detectados deja de ser visible y luego vuelve a ser detectado, se le asignará un nuevo identificador, no permitiendo así asociar el mismo identificador que tenía antes. Si bien tener un reconocimiento único a lo largo del tiempo no es posible con la versión actual del SDK de Leap Motion Controller, es algo que los desarrolladores tienen pensando permitir.

1.4 Aplicaciones

Como ya se ha visto hay una gran variedad de dispositivos que interactúan con el usuario y le dan una experiencia muy cercana a la realidad. Por eso estos dispositivos se encuentran en multitud de Campos.

Medicina. Facilita la manipulación de órganos internos e intervenciones quirúrgicas, también la creación de pacientes virtuales para poner en práctica las habilidades de los estudiantes, un ejemplo de esto es: "Sistema de Entrenamiento virtual para medicina"[1].

Ciencia. Para el estudio de los sistemas de partículas y moléculas.

Diseño. Herramientas de CAD para diseñar prototipos y probarlos en ambientes virtuales.

Defensa. Simuladores de vuelo, entrenamiento militar.

Arquitectura. Navegación por futuros edificios.

Enseñanza. Facilita la atención de los estudiantes mediante su inmersión en mundos virtuales.

También hay proyectos y aplicaciones destinadas al control de robots o elementos parecidos con otro tipo de actuador.

Los controles en el ámbito industrial más comunes son los joysticks y los controladores numéricos, los joysticks permiten un movimiento continuo e intuitivo, en cambio los controladores numéricos son más precisos pero complicados de usar. Fuera del ámbito industrial hay varios modos de control muy innovadores, como por ejemplo el control de actuadores mediante impulsos eléctricos del cerebro.

Capítulo 2
Manual 5DT Data Glove

2.1 Introducción

El 5DT Data Glove 16, obtiene medidas de la flexión del dedo, así como la separación entre los dedos. La flexión de los dedos se mide en dos lugares: Primero, conjuntos nudillo y segundo, conjunto dedo.

Características:

- Calidad asequible.
- Calibración automática, resolución mínima flexión de 8 bits.
- Conexión a cualquier plataforma - interfaz de puerto serie (RS 232).
- Software incluido.
- Alta tasa de actualización.
- La versión inalámbrica disponible (5DT Data Glove 16-W).
- Baja diafonía entre los dedos.
- Procesador a bordo.
- Conexión rápida.
- Drivers para varias plataformas.

2.2 Instalación y Configuración

2.2.1 Contenido del paquete

El paquete estándar 5DT Data Glove 16 consta de los siguientes elementos:

- El 5DT Data Glove 16 cuenta con sensores de fibra óptica, cable de transmisión y unidad de procesamiento.
- Cable de interfaz.
- Una unidad de fuente de alimentación
- Un CD-ROM de instalación.
- Manual de uso.

El paquete para el 5DT Data Glove 16-W inalámbrica consta de los siguientes elementos:

- El 5DT Data Glove 16-W con sensores de fibra óptica, sensor de inclinación, cable de datos y unidad de procesamiento.
- Cable de interfaz
- Una unidad de fuente de alimentación
- Unidad de recepción inalámbrica
- Unidad de transmisor inalámbrico

- Cable de interfaz transmisor Corto
- Cable de interfaz transmisor Largo
- Manual de uso.
- Un conjunto de discos de instalación y/o CD-ROM

2.2.2 Conexión del guante estándar

El 5DT Data Glove 16 consiste en un guante de licra con sensores de fibra óptica incrustados.

Estos sensores están vinculados al ordenador a través de una unidad opto-eléctrica, a través de un cable de datos y una caja de interfaz.

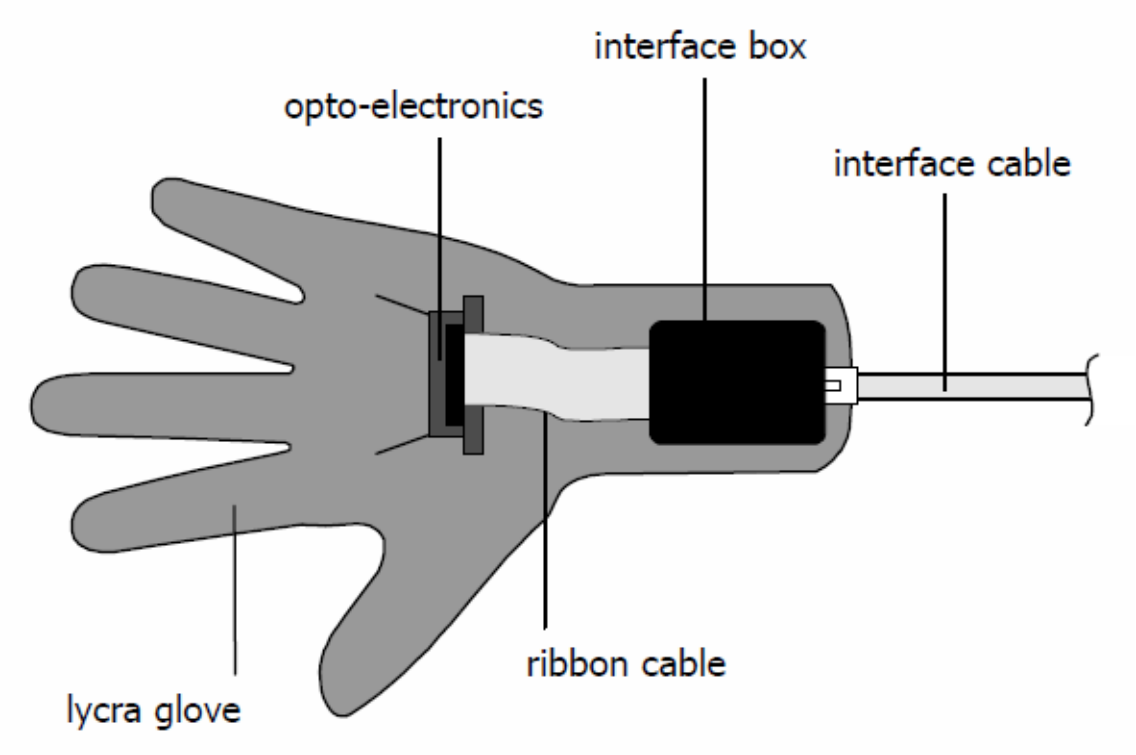


Figura 8. Componentes del 5DT Data Glove 16

El 5DT Data Glove 16 se conecta a un puerto serie RS 232 de 9 pines, (conector DB9) a través del cable de interfaz. Si el guante se va a utilizar con un conector de puerto serie de 25 pines, de 9 a 25 pines, se utilizará un adaptador.

El procedimiento recomendado para conectar el guante al equipo será de la siguiente manera (consulte la Figura 9.):

1. Conecte el cable serie al ordenador (conector DB9).

2. Conecte la fuente de alimentación al cable de interfaz/potencia de serie.

3. Conecte el cable serie al guante (conector RJ12).

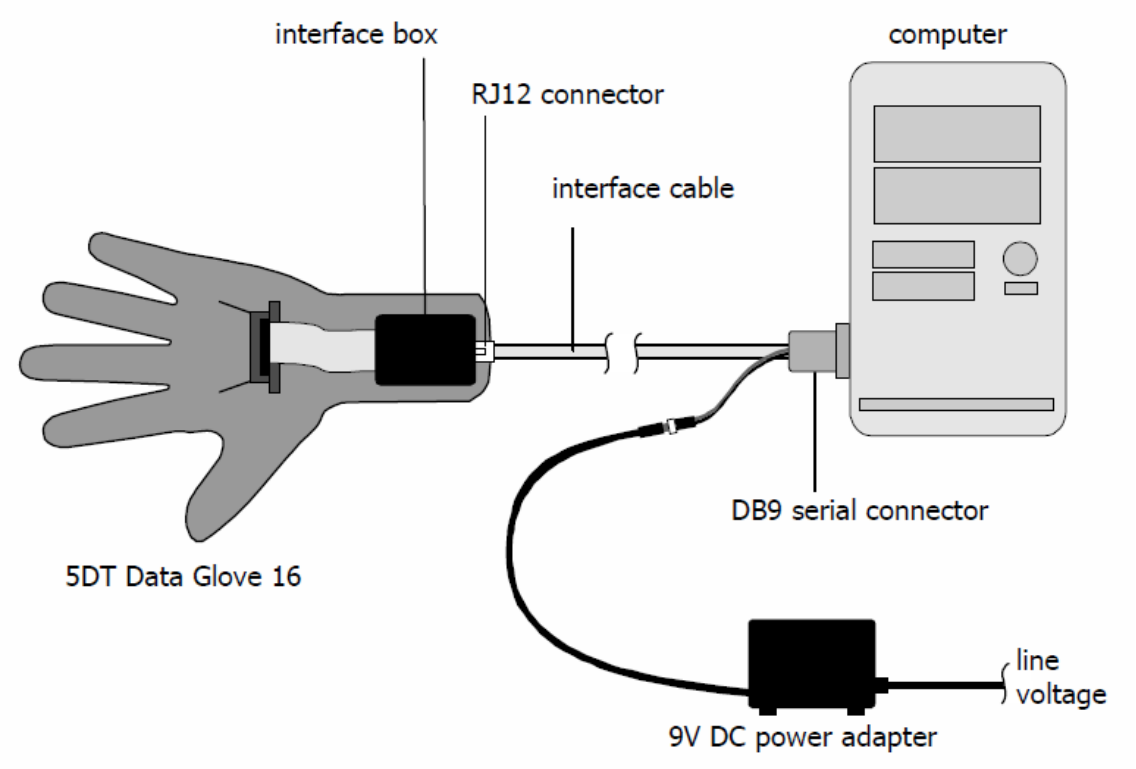


Figura 9. Conexión estándar al ordenador.

2.2.3 Instalación del Software.

El 5DT Data Glove 16 se suministra con un CD- ROM. Este disco contiene el software de calibración y pruebas, aplicaciones, drivers, demos, programación, ejemplos y el manual en formato electrónico.

El software suministrado está pensado para Windows de 32 bits, y requieren Windows 95/98/NT o compatibles.

Para instalar el software en el ordenador, inserte el CD en su lector de disco, ejecutar el programa setup.exe y siga las instrucciones que aparecen en pantalla.

La carpeta de instalación sugerida es:

C:\Archivos de programa\5DT\Data Glove 16\

El software puede no obstante ser instalado en cualquier carpeta de su elección.

El software instalado en su disco duro incluye:

- Gestor guante (más detalle en la sección 2.3).

...\GloveMan

- Drivers del guante (más detalle en el Apéndice A)

...\Driver

- Guante Demo (más detalle en la sección 2.4)

...\Demo

- Ejemplos de programación

...\Ejemplos de programación\VC Muestra

- Manuales

...\Manual

La carpeta de instalación predeterminada de los accesos directos es:

...\Windows\Start Menu\Programs\Data Glove 16

Los accesos directos que se crearán en esta carpeta son:

- Gestor del guante
- Demo
- Manual

2.3 Gestor del Guante.

El 5DT Glove Manager es un programa útil que se puede utilizar para lo siguiente:

- Conexión del 5DT Data Glove al ordenador.
- Pruebas del 5DT Data Glove.
- Mostrar una rutina de calibración de software para los datos del guante.
- Mostrar una rutina de registro de datos para los datos del guante.

2.3.1 Ejecución del Gestor

Inicie el programa ejecutando GloveMan.exe, que se instala en el directorio

…\Data Glove 16\GloveMan\ carpeta por defecto. La ventana principal del programa aparecerá:

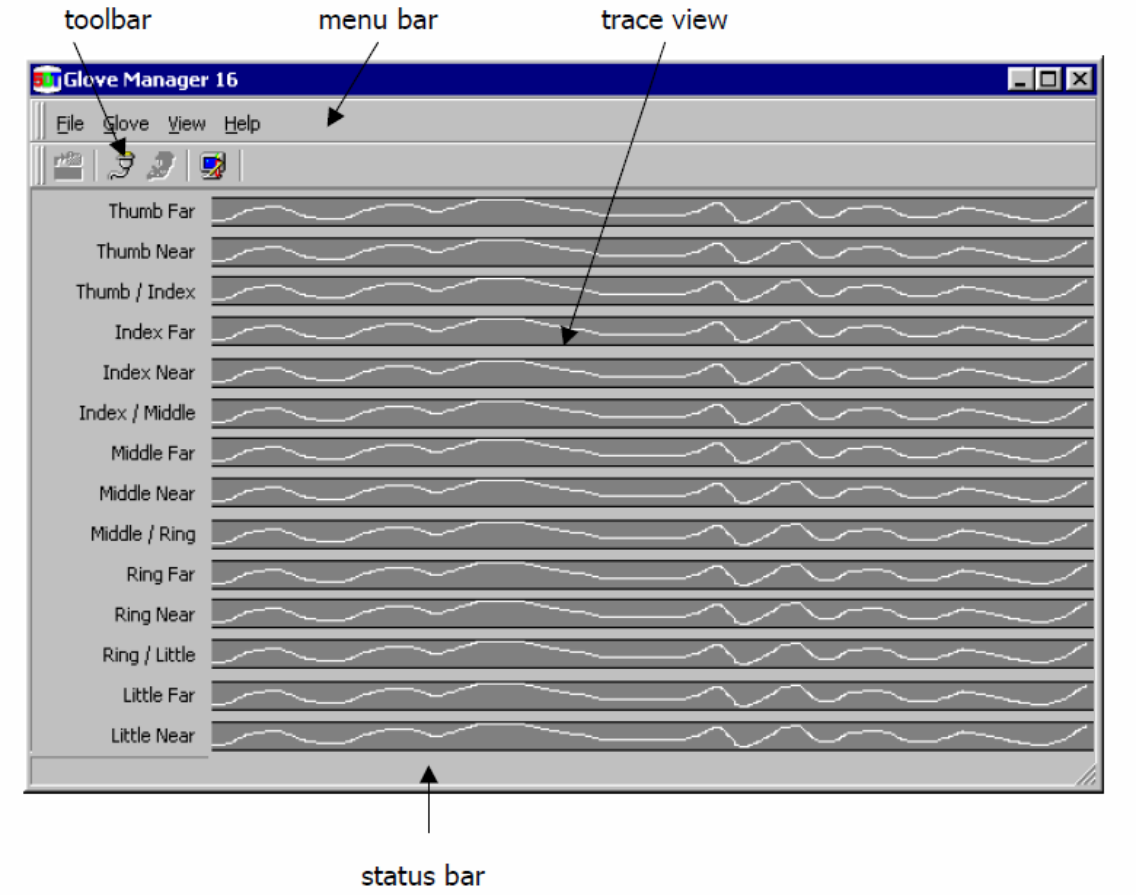


Figura 10. Ventana principal Gestor.

La parte superior de la ventana alberga un menú y una barra de herramientas, y hay una barra de estado en la parte inferior. La vista de seguimiento muestra gráficos de la flexión de los catorce sensores implementadas.

2.3.2 Conectar el guante al ordenador

Antes de poder utilizar el guante, es necesario conectarse a él. Seleccione Conectar en el menú del guante, o haga clic en el icono de la barra de herramientas correspondiente. El cuadro de diálogo de conexión se aparecerá, mostrando una lista de puertos disponibles. Seleccione el puerto (por ejemplo, COM2) en el que el guante está conectado y luego haga clic en el botón Conectar.

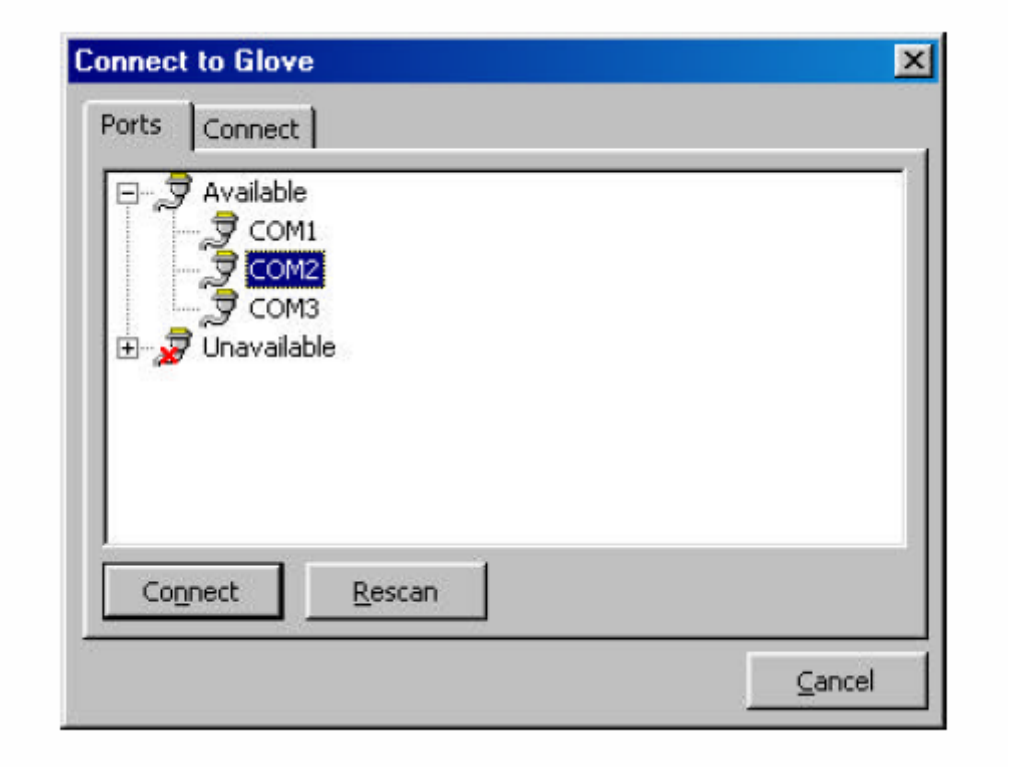


Figura 11. Ventana de conexión del Gestor.

El programa pasará de la pestaña Ports a la pestaña Connect y luego intente abrir el puerto y conectar con el guante. El siguiente mensaje será mostrado:

Opening port COM2...
COM2 opened. Detecting Data Glove 16...
Data Glove 16 found.
Connected to Glove.

Si un guante no se puede encontrar, se mostrará la razón exacta de esto.

Haga clic en el botón Cerrar para cerrar el cuadro de diálogo de conexión. En la ventana principal, los gráficos mostrarán la flexión actual de los sensores. Como se flexiona y deflexionan los dedos, las señal en los gráficos de flexión se moverá superior e inferior respectivamente.

2.3.3 Rutina de calibración del software

Cada persona tiene una mano diferente en tamaño. El Data Glove 5DT está diseñado para adaptarse a la mayoría de los usuarios cómodamente, pero necesita ser calibrado con el fin de lograr la máxima sensibilidad dentro del programa.

Es importante señalar que esta rutina de calibración calibra el software de control del guante para su uso dentro del programa Gestor. Sirve como una ilustración/ejemplo solamente y no calibrar el guante para otros programas/aplicaciones. Tendremos que desarrollar una rutina de calibración en el programa para nuestras

aplicaciones. Tenga en cuenta que el Data Glove suministrado cuenta con una calibración del software incorporado.

Para calibrar el guante para su uso dentro del gestor, seleccione Calibrar en el menú, o haga clic en el icono de la barra de herramientas correspondiente. El cuadro de diálogo de calibración aparecerá:

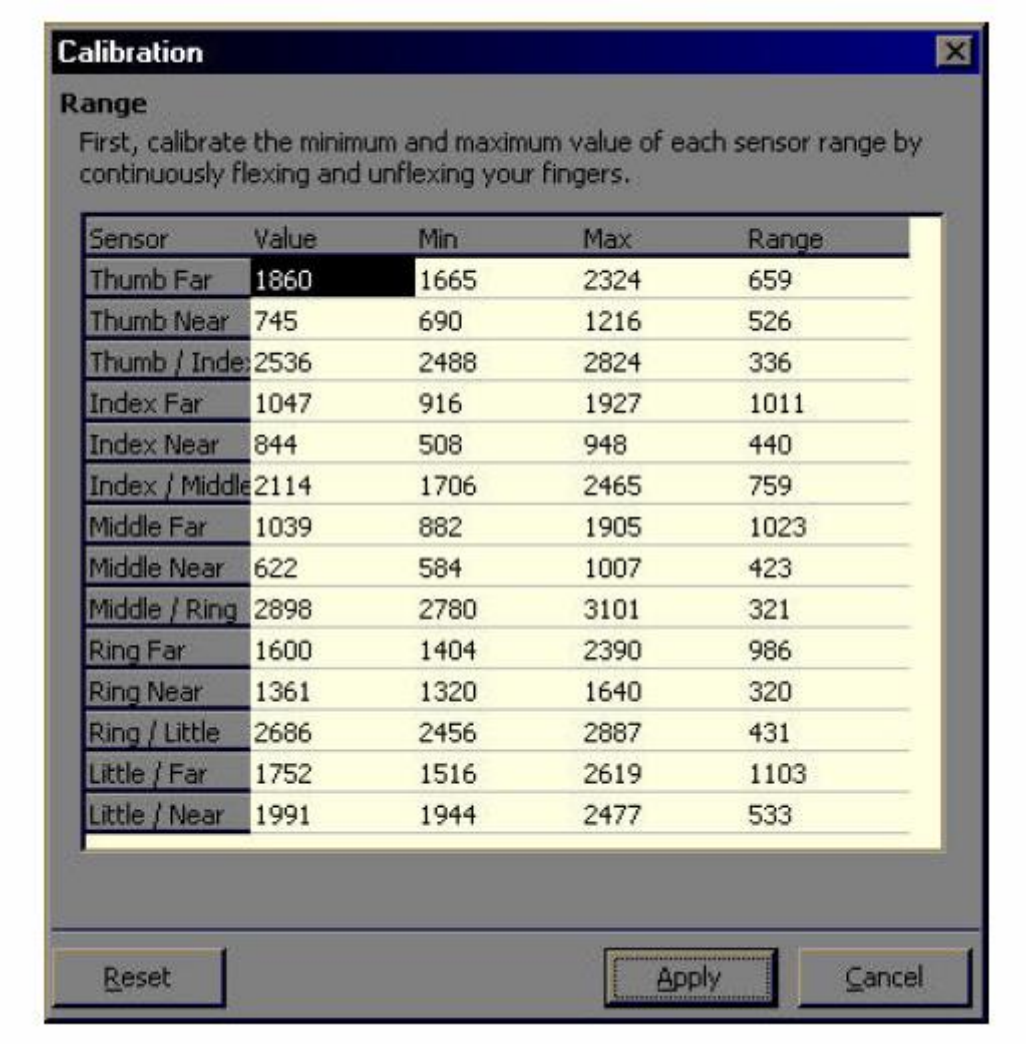


Figura 12. Calibración del guante: Valores de ajuste.

Para calibrar, realizar repetidamente la flexión de cada uno de los dedos. Es importante mantener la mano relajado en todo momento, manteniendo el movimiento natural. A medida que flexiona los dedos, los valores mínimos y máximos de cada uno de los sensores se muestran en la columna respectiva, y el rango dinámico en la última columna. El valor actual de cada sensor se muestra en la primera columna.

Una vez que esté satisfecho de que los valores mínimos y máximos se han creado, haga clic en el botón Aplicar para finalizar.

Para ahorrar tiempo en el futuro, es posible guardar la información de calibración en un archivo.

A continuación, puede cargar la calibración en cualquier momento con gran rapidez. Seleccionar Guardar calibración o Cargar Calibración en el menú Archivo de la ventana principal.

2.3.4 Guardado de los datos del guante a un archivo de registro.

Hay situaciones en las que se necesita "grabar" los datos recibidos del guante para guardarlo en un archivo de registro.

Data Glove 16 puede guardar los datos en formato CSV (valores separados por comas). La estructura CSV es muy simple, y los archivos en este formato se puede leer en programas como Microsoft Excel muy fácilmente.

Para guardar los datos en un archivo, abrir Glove Manager y selecciona Save datos de la Menú File. El siguiente cuadro de diálogo aparecerá:

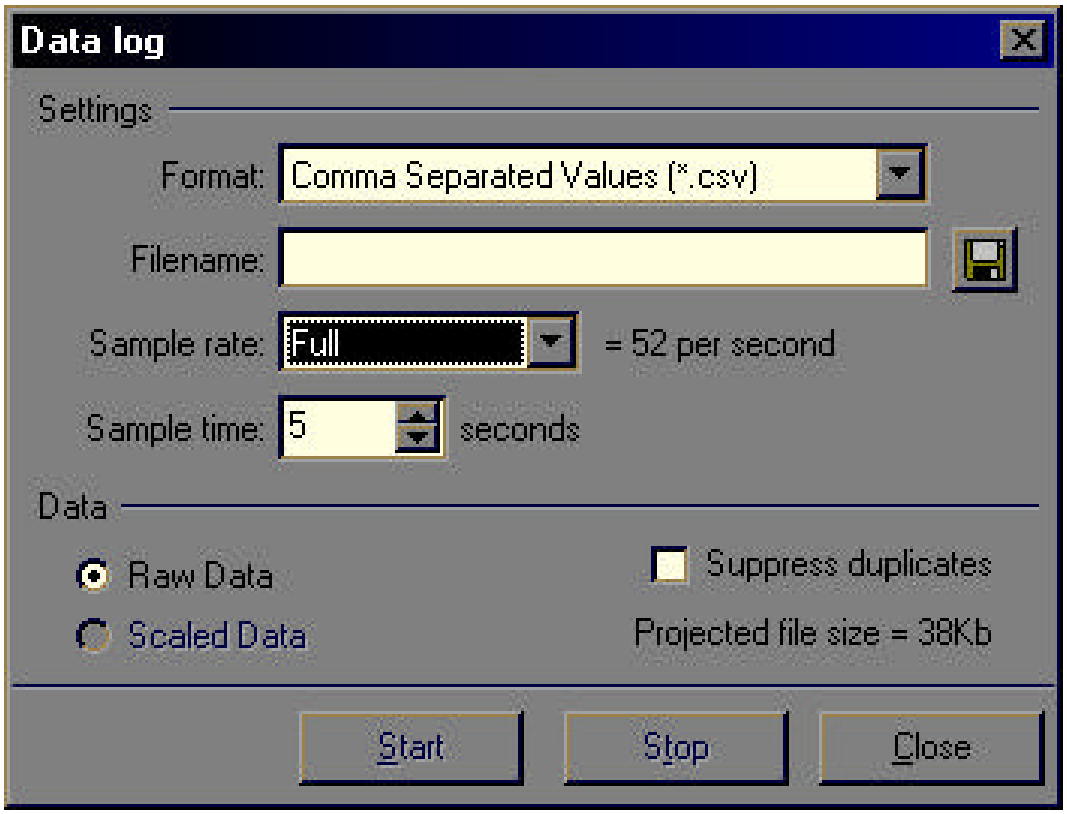


Figura 13. Ventana de registro de datos del Guante.

La última opción es Suppress Duplicates (Suprimir duplicados). Si seleccionamos esta opción, el programa sólo da salida a un valor de datos si éste cambia. Se seleccionará esta opción si usted ha desarrollado su propio software de procesamiento y desea aprovechar el ahorro de espacio en disco, para un procesamiento de datos más veloz.

2.3.5 Cambio de las opciones del programa.

El Data Glove tiene muchas opciones que se pueden configurar para cambiar la forma en que funciona. Para cambiar nos vamos a Glove Manager, seleccionar Ver, a continuación, en Opciones.

El cuadro de diálogo de opciones aparecerá Éstas son algunas de las opciones que se puede ajustar:

Cerrar el cuadro de diálogo de conexión en conexión con éxito. Marcar esta opción para forzar el cuadro de diálogo se cierre automáticamente cuando una conexión con éxito se hace con un guante.

Consejos para mostrar la barra de herramientas. Marque esta opción para activar la visualización de la barra de herramientas consejos. Estos consejos explican lo que representa cada icono, y la lista de la tecla de acceso directo para los comandos.

Puede cambiar la apariencia del programa mediante la selección de uno de los colores predefinidos. También es posible crear su propio esquema mediante el ajuste colores.

2.4 Otros Software

Por favor, consulte el archivo Readme.txt en la carpeta de instalación principal. Este archivo explica todo el software que se instala en el disco duro.

Gover Driver

...\Driver

El 5DT Data Glove se explica en detalle en el Apéndice D.

Guante Demos

...\Demo

Estos Demos actualmente le permiten hacer lo siguiente:

Ver la parte animada por su cuenta

Utilice la parte animada de navegar un paisaje

Animar el movimiento de un carácter

Animar el movimiento de un personaje en un paisaje

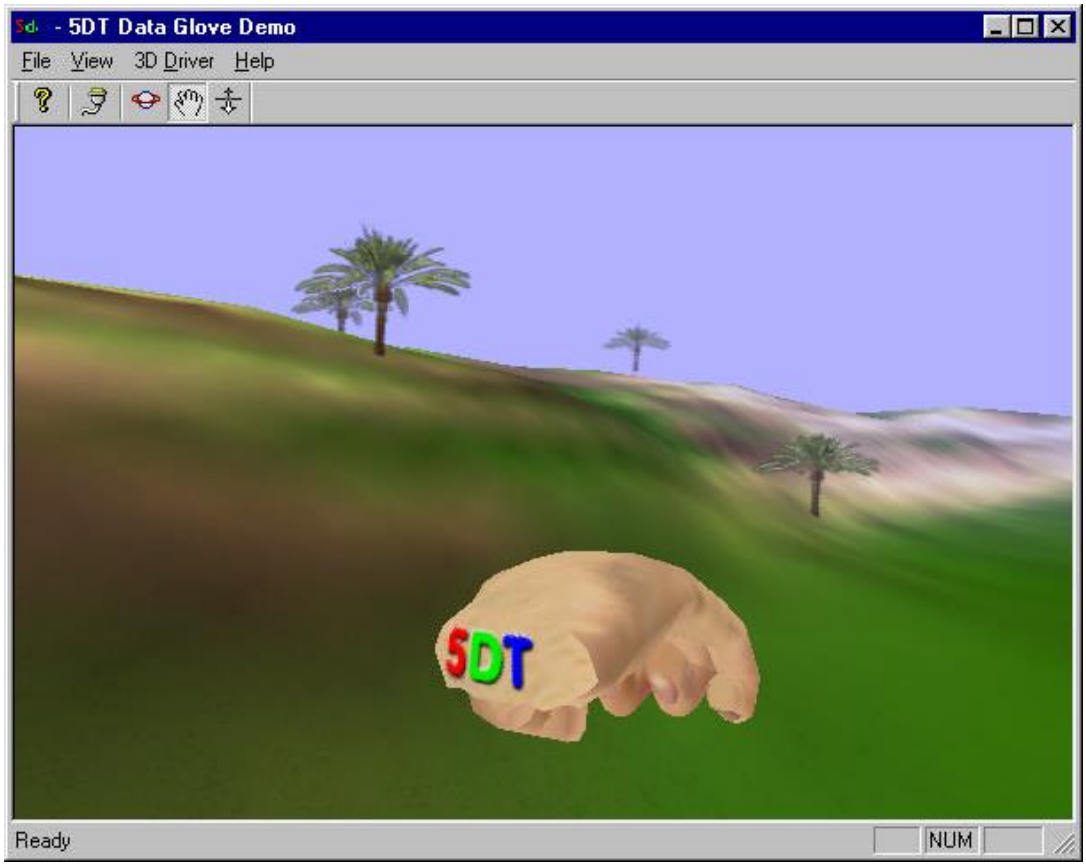


Figura 14. Captura de pantalla de la demo de 5DT Data Glove 16.

2.5 Especificación Hardware

MATERIAL:

Negro lycra tramo

FLEXIÓN RESOLUCIÓN:

12-bit A/D para cada sensor. Rango dinámico mínimo es de 8-bits.

SENSORES BEND:

Tecnología flexor basado fibra óptica patentada. 2 sensores por dedo, 1 sensor entre cada dedo.

(Sensor de posición del pulgar y el sensor de flexión de la muñeca todavía no implementado)

INTERFAZ DE ORDENADOR:

Guante estándar: RS-232 (3 hilos), GND, TX, 19200 bps (sólo transmisión), 8 bits de datos, 1 bit de parada, sin paridad.

Guante inalámbrico: RS-232 (2 hilos), GND, TX, 9600 bps (sólo transmisión), 8 bits de datos, 1 bit de parada, sin paridad.

ALIMENTACIÓN:

Máximo de 150 mA a 9 V DC, Conector de alimentación de CC centro positivo.

TASA DE MUESTREO:

La mano completa (14 sensores) puede ser muestreada a 100 muestras por segundo.

FRECUENCIA DE TRANSMICION:

Guante inalámbrico diestro: 418 MHz

Guante inalámbrico Zurdo: 433,92 MHz

ALCANCE INALAMBRICO TÍPICO: hasta 30m

2.5.1 Detalles Técnicos de la Interfaz de Cableado

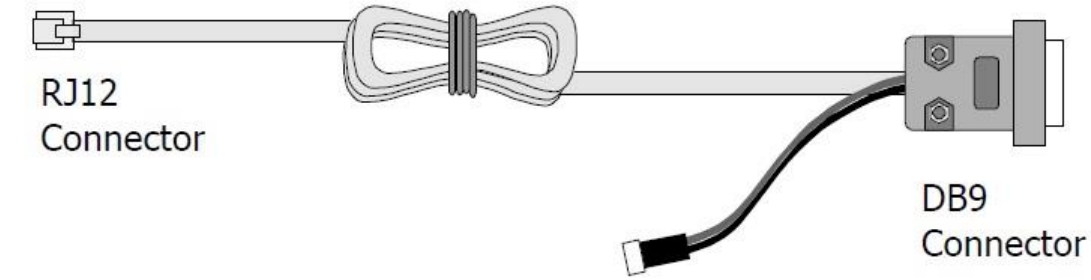


Figura 10. El cable interfaz

Figura 15. RJ12 Conector.

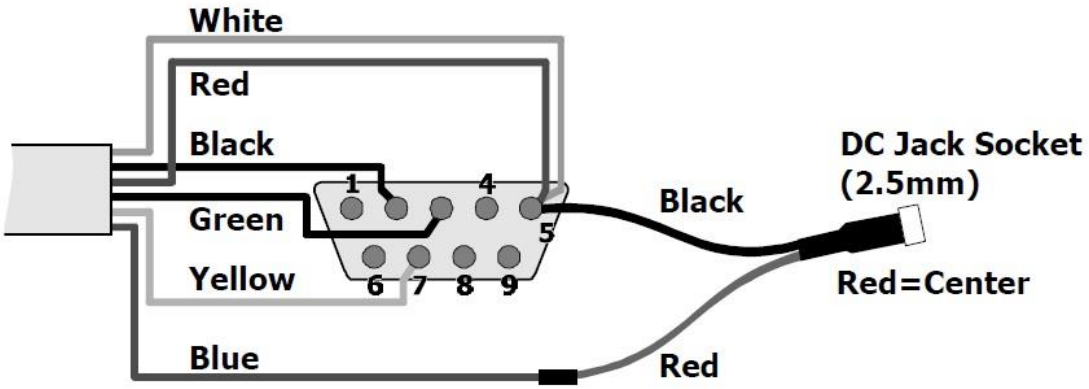


Figura 16. DB9 Estructura del conector.

DB9 Pines usados:

- 2 – RD
- 3 – TD*
- 5 – GND
- 7 – RTS* (modo de emulación de ratón)

* Pines 3 y 7 son usados por el guante.

2.6 Drivers 5DT Data Glove

El 5DT Data Glove proporciona acceso a la gama de guantes de datos en un nivel intermedio. Proporcionan para Windows 95/98/NT la versión de la librerías en un archivo C/C ++ denominado header (.h), para Microsoft Visual C++ se incluye la biblioteca (.lib) y una biblioteca de librería dinámica (.dll). Proporcionan para Linux en la forma de un C/C ++ el header (.h), y una biblioteca dinámica (.so). La funcionalidad de los driver incluye:

- Múltiples casos.
- Inicialización fácil y parada.
- Valor de los sensores básico.
- Valor de los sensores escalado (auto calibrado).
- Funciones de Calibración.
- El reconocimiento de gesto Básico.
- El Uso de la aplicación, Programa del Interfaz.

2. Utilización de los driver del guante.

El 5DT Data Glove Driver es fácil puesta en marcha. Use las siguientes directrices:

Windows 95/98/NT

i. Asegúrese que el archivo de header *fglove.h*, el archivo de biblioteca *fglove.lib* y el archivo de biblioteca de librerías dinámicas *fglove.dll* reside en el directorio actual

(de aplicación), o en algún sitio que ellos pueden ser llamados. El archivo *fglove.dll* puede ser copiado en su directorio de sistema de Windows.

ii. Incluya el archivo de header fglove.h en la aplicación donde sea necesario.

iii. Añada el archivo de biblioteca *fglove.lib* al proceso principal. Hay un también una versión de ajuste del driver (fgloved.lib, fgloved.dll) cuyas salidas eliminan fallos de mensajes al depurador.

Linux

El driver requiere el Linux Posix, la biblioteca de hilos *libpthread.so*, que por lo general es instalado con Linux. El driver está vinculado a libc6.

Instalación de la biblioteca

El driver es un archivo de biblioteca dinámica (*libfglove.so*) que debe estar instalado en algún lugar donde las aplicaciones pueden encontrar. Si tiene acceso root al sistema, el método más fácil es copiar el archivo en el directorio */usr/lib* , por ejemplo:

 # Cp libfglove.so /usr/lib

Si usted no tiene acceso root al sistema, debe colocar la biblioteca en algún lugar de su directorio personal, a continuación, establezca la variable de entorno *LD_LIBRARY_PATH* para incluir la ruta completa del directorio en el que se ha colocado la biblioteca. Esto indicará al cargador de bibliotecas dinámicas dónde encontrar el archivo. Por ejemplo:

$ mkdir /home/yourhomedir/libs

$ cp libfglove.so /home/yourhomedir/libs

$ export LD_LIBRARY_PATH="/home/yourhomedir/libs"

En el ejemplo anterior, reemplace "yourhomedir" con su propio nombre directorio principal.

Configuración del acceso al puerto serie

El Data Glove 5DT accede al puerto serie utilizando los archivos en dispositivos estándar de Unix/Linux en el directorio /dev. Las aplicaciones que utilizan el guante deben típicamente dar al usuario la opción de especificar qué dispositivo comprobar, por ejemplo: */dev/ttyS1*. Se recomienda crear un enlace simbólico /dev/fglove a su archivo de dispositivo, por ejemplo:

 # Cd /dev
 # Ln -s ttyS1 fglove

Esto puede hacer que la configuración de las aplicaciones sea más fácil si necesitas cambiar el puerto donde el guante está conectado.

También se requiere que el usuario del guante tenga acceso de lectura/escritura en el archivo de dispositivo de puerto serie de forma predeterminada. Normalmente sólo

el usuario root tiene estos derechos. Si los usuarios no root en el sistema van a utilizar el guante, el usuario root debe otorgar derechos de acceso al archivo de dispositivo. Por ejemplo, el siguiente comando dará a cada uno en el sistema de acceso completo al puerto serie */dev/ttyS1*:

 # Chmod 777 /dev/ttyS1

El archivo de cabecera

El archivo de cabecera (*fglove.h*) se debe copiar en algún lugar donde el compilador puede encontrarlo. Si usted tiene acceso a la raíz, el más fácil es copiar a */usr/include*. Si no, entonces sólo tiene que colocar el archivo de cabecera en el mismo directorio que la aplicación.

Ejemplo de código fuente

Código fuente de ejemplo que hace uso del controlador de guante se distribuye con los drivers.

2.7 Guantes compatibles

El driver del guante soporta todos los 5DT Data Glove. La versión actual implementa 18 posibles sensores, e incluye los sensores de balanceo y cabeceo del 5DT Data Glove 5. El driver intenta asignar valores a todas las salidas de los sensores. Si no es capaz de hacerlo el valor predeterminado sensor a un valor razonable. Este valor se puede ajustar al obligar a dar un valor específico.

Asignaciones de los sensores para el Guante 5DT Data Glove

Los sensores en el 5DT datos Guante 5 se colocan como en la Figura 17

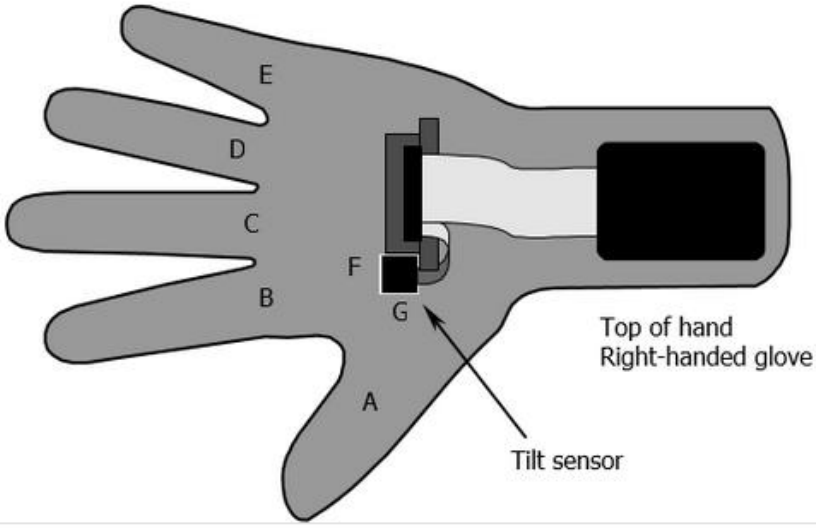


Figura 17. Asignaciones de sensores para la 5DT Data Glove 5.

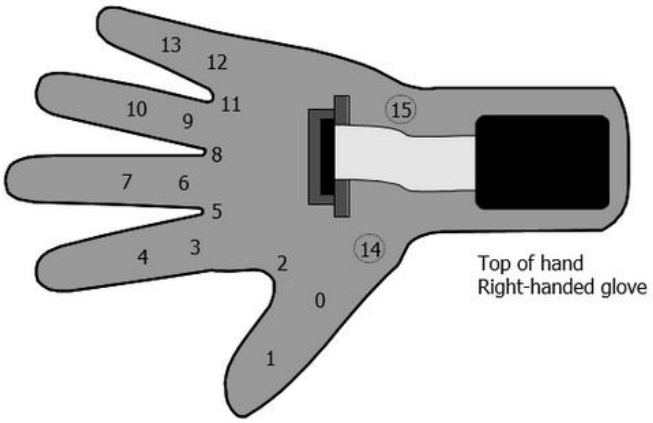


Figura 18. Asignaciones de sensores para la 5DT Data Glove 16.

2.8 Auto-calibración

El driver puede proporcionar salidas de los sensores de forma automática, calibrándose de forma lineal. Durante cada actualización, el valor nominal se leerá desde el sensor se compara con los valores máximos y mínimos primas actual y (raw min y raw max) tal como se establece por las funciones *fdSetCalibrationAll ()*, *fdSetCalibration ()* o *fdResetCalibration ()*. Si se exceden los valores máximos y mínimos actuales, que son sobrescritos. Por tanto, los valores de calibración superior e inferior son empujados continuamente "hacia fuera". La salida normalizada está dada por la ecuación de primer orden

$$out = \frac{raw_{val} - raw_{\min}}{raw_{\max} - raw_{\min}} . Max, \qquad \text{(F-1)}$$

Ésta está en [0, ... Max]. El valor de Max se establece por las funciones *fdSetSensorMaxAll ()* y *fdSetSensorMax ()*. Haciendo unos pocos movimientos de flexión con la mano pone rápidamente los valores de operación para raw min y raw max y calibra el guante.

El proceso de auto-calibración puede ser ignorado simplemente sólo respecto a las salidas de los sensores primarios. Siempre el desarrollador de aplicaciones puede proporcionar un proceso de calibración adecuado. Tenga en cuenta que la calibración es obligatorio, especialmente con el guante 16 sensores que no contiene posibilidades de calibración hardware (potenciómetros preestablecidos en miniatura).

2.9 Software de calibración

Cuando el 5DT Data Glove se calibra en la fábrica, tratamos de obtener el mayor rango dinámico posible. El rango dinámico es la diferencia entre el valor máximo de salida (lado flexionada) y el valor de salida mínimo (parte plana).

Dynamic Range = Valuemax - Valuemin

El rango dinámico puede variar cuando personas con diferentes tamaños de mano utilizan el guante. Por ello, recomendamos que una rutina de calibración del software se incluye con todos los guantes aplicaciones/programas.

El software de calibración normaliza el efecto de diferentes rangos dinámicos para diferentes tamaños de mano.

La rutina de calibración del software debe almacenar los valores mínimo y máximo y escalar la salida para el máximo rango dinámico. Por ejemplo, vamos a trabajar a través de un ejemplo para el pulgar.

El valor mínimo es: 40 = Value min
El valor máximo es: 206 = Value max
El rango dinámico es: 206-40 = 166

Para escalar los valores medidos a través de todo el rango dinámico (256 valores), por lo tanto, tiene que realizar la siguiente operación:

Valuescaled = (Valuemeasured - Valuemin) x (255/Dynamic Range)
Para esto se convierte en 40 (40-40) x (255/166) = 0
Para esto se convierte en 206 (206-40) x (255/166) = 255

Esta rutina también podría ser implementada de forma dinámica, en donde el guante se calibra mientras el usuario lo utiliza. La rutina es similar al régimen normal de calibración con la diferencia de que el valor mínimo y máximo se actualiza continuamente. Los valores iniciales para el mínimo y máximo pueden ser elegidos como 126 y 128 respectivamente.

Tenga en cuenta que el controlador 5DT Data Glove incluye una rutina de calibración del software dinámico.

Capítulo 3
Protocolo TCP/IP y Protocolo RS-232

3.1 Protocolo TCP/IP

3.1.1 Definición

Se han desarrollado diferentes familias de protocolos para comunicación por red de datos para los sistemas UNIX. El más ampliamente utilizado es el Internet Protocol Suite, comúnmente conocido como TCP/IP.

Es un protocolo DARPA que proporciona transmisión fiable de paquetes de datos sobre redes. El nombre TCP/IP Proviene de dos protocolos importantes de la familia, el Transmission Control Protocol (TCP) y el Internet Protocol (IP). Todos juntos llegan a ser más de 100 protocolos diferentes definidos en este conjunto.

El TCP/IP es la base del Internet que sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local y área extensa. TCP/IP fue desarrollado y mostrado por primera vez en 1972 por el Departamento de Defensa de los Estados Unidos, ejecutándolo en el ARPANET una red de área extensa del Departamento de Defensa.

3.1.2 Capas Conceptuales del Software de Protocolos

Conceptualmente, enviar un mensaje desde un programa de aplicación en una máquina hacia un programa de aplicaciones en otra, significa transferir el mensaje hacia abajo, por las capas sucesivas del software de protocolo en la máquina emisora, transferir un mensaje a través de la red y luego, transferir el mensaje hacia arriba, a través de las capas sucesivas del software de protocolo en la máquina receptora.

En términos generales, el software TCP/IP está organizado en cuatro capas conceptuales que se construyen sobre una quinta capa de hardware. El siguiente esquema muestra las capas conceptuales así como la forma en que los datos pasan entre ellas.

- **Capa de aplicación**. Es el nivel más alto, los usuarios llaman a una aplicación que acceda servicios disponibles a través de la red de redes TCP/IP. Una aplicación interactúa con uno de los protocolos de nivel de transporte para enviar o recibir datos. Cada programa de aplicación selecciona el tipo de transporte necesario, el cual puede ser una secuencia de mensajes individuales o un flujo continuo de octetos. El programa de aplicación pasa los datos en la forma requerida hacia el nivel de transporte para su entrega.

- **Capa de transporte.** La principal tarea de la capa de transporte es proporcionar la comunicación entre un programa de aplicación y otro. Este tipo de comunicación se conoce frecuentemente como comunicación punto a punto. La capa de transporte regula el flujo de información. Puede también proporcionar un transporte confiable, asegurando que los datos lleguen sin errores y en secuencia. Para hacer esto, el software de protocolo de transporte tiene el lado de recepción enviando acuses de recibo de retorno y la parte de envío retransmitiendo los paquetes perdidos. El software de transporte divide el flujo de datos que se está enviando en pequeños fragmentos (por lo general conocidos como paquetes) y pasa cada paquete, con una dirección de destino, hacia la siguiente capa de transmisión. La capa de transporte debe aceptar datos desde varios programas de usuario y enviarlos a la capa del siguiente nivel. Para hacer esto, se añade información adicional a cada paquete, incluyendo códigos que identifican qué programa de aplicación envía y qué programa debe recibir, así como una suma de verificación para verificar que el paquete ha llegado intacto y utiliza el código de destino para identificar el programa de aplicación en el que se debe entregar.

- **Capa Internet**. La capa Internet maneja la comunicación de una máquina a otra. Ésta acepta una solicitud para enviar un paquete desde la capa de transporte, junto con una identificación de la máquina, hacia la que se debe enviar el paquete. La capa Internet también maneja la entrada de datagramas, verifica su validez y utiliza un algoritmo de ruteo para decidir si el datagrama debe procesarse de manera local o debe ser transmitido. Para el caso de los datagramas direccionados hacia la máquina local, el software de la capa de red de redes borra el encabezado del datagrama y selecciona, de entre varios protocolos de transporte, un protocolo con el que manejará el paquete.

- **Capa de interfaz de red.** El software TCP/IP de nivel inferior consta de una capa de interfaz de red responsable de aceptar los datagramas IP y transmitirlos hacia una red específica. Una interfaz de red puede consistir en un dispositivo controlador (por ejemplo, cuando la red es una red de área local a la que las máquinas están conectadas directamente) o un complejo subsistema que utiliza un protocolo de enlace de datos propios (por ejemplo, cuando la red consiste de conmutadores de paquetes que se comunican con anfitriones utilizando HDLC).

3.1.3 Funcionamiento TCP/IP

Una red TCP/IP transfiere datos mediante el ensamblaje de bloques de datos en paquetes, cada paquete comienza con una cabecera que contiene información de control;

tal como la dirección del destino, seguido de los datos. Cuando se envía un archivo por la red TCP/IP, su contenido se envía utilizando una serie de paquetes diferentes. El Internet Protocol (IP), un protocolo de la capa de red, permite a las aplicaciones ejecutarse transparentemente sobre redes interconectadas. Cuando se utiliza IP, no es necesario conocer que hardware se utiliza, por tanto ésta se ejecuta en una red de área local.

El Transmissión Control Protocol (TCP); un protocolo de la capa de transporte, asegura que los datos sean entregados, que lo que se recibe, sea lo que se pretendía enviar y que los paquetes que sean recibidos estén en el orden en que fueron enviados. TCP terminará una conexión si ocurre un error que haga imposible la transmisión fiable.

3.1.4 Administración TCP/IP

TCP/IP es una de las redes más comunes utilizadas para conectar computadoras con sistema UNIX. Las utilidades de red TCP/IP forman parte de la versión 4, muchas facilidades de red como un sistema UUCP, el sistema de correo, RFS y NFS, pueden utilizar una red TCP/CP para comunicarse con otras máquinas.

Para que la red TCP/IP esté activa y funcionado será necesario:

- Obtener una dirección Internet.
- Instalar las utilidades Internet en el sistema
- Configurar la red para TCP/IP
- Configurar los guiones de arranque TCP/IP
- Identificar otras máquinas ante el sistema
- Configurar la base de datos del oyente de STREAMS
- Comenzar a ejecutar TCP/IP.

3.2 Protocolo RS-232

3.2.1 Definición

El protocolo RS-232 es un protocolo de comunicación serial asíncrono, esto es, que no tiene un orden de envío de datos entre los dispositivos, por lo que se vuelve necesario el cuidar la sincronización del envío para evitar pérdidas de información o fallos en la comunicación.

Otra de sus características principales es ser un protocolo punto a punto, esto es, que solamente permite la comunicación de un dispositivo con respecto a otro empleando una terminal de comunicación determinada. No permite la creación de redes.

El protocolo puede trabajarse de manera asíncrona o síncrona y tipos de canal simplex, half duplex o full duplex. En un canal simplex los datos siempre viajarán en una dirección, por ejemplo desde DCE a DTE. En un canal half duplex, los datos pueden viajar en una u otra dirección, pero sólo durante un determinado periodo de tiempo; luego la línea debe ser conmutada antes que los datos puedan viajar en la otra dirección. En un canal full duplex, los datos pueden viajar en ambos sentidos simultáneamente. Las líneas de handshaking de la RS-232 se usan para resolver los problemas asociados con este modo de operación, tal como en qué dirección los datos deben viajar en un instante determinado.

3.2.1 Características básicas

Es un protocolo desarrollado para la comunicación serial de dispositivos sencillos, ampliamente utilizado debido a la facilidad de comunicación y las ventajas en coste que representa la comunicación serial. Hace uso de conectores de tipo DB-25, sin embargo, es común observar dispositivos con conectores de tipo DB- 9, de 9 pines debido a su menor costo. El interfaz eléctrico utiliza una conexión eléctrica asimétrica con circuitos no equilibrados, todos referenciados a tierra. Los estados lógicos son definidos por los siguientes niveles de voltaje.

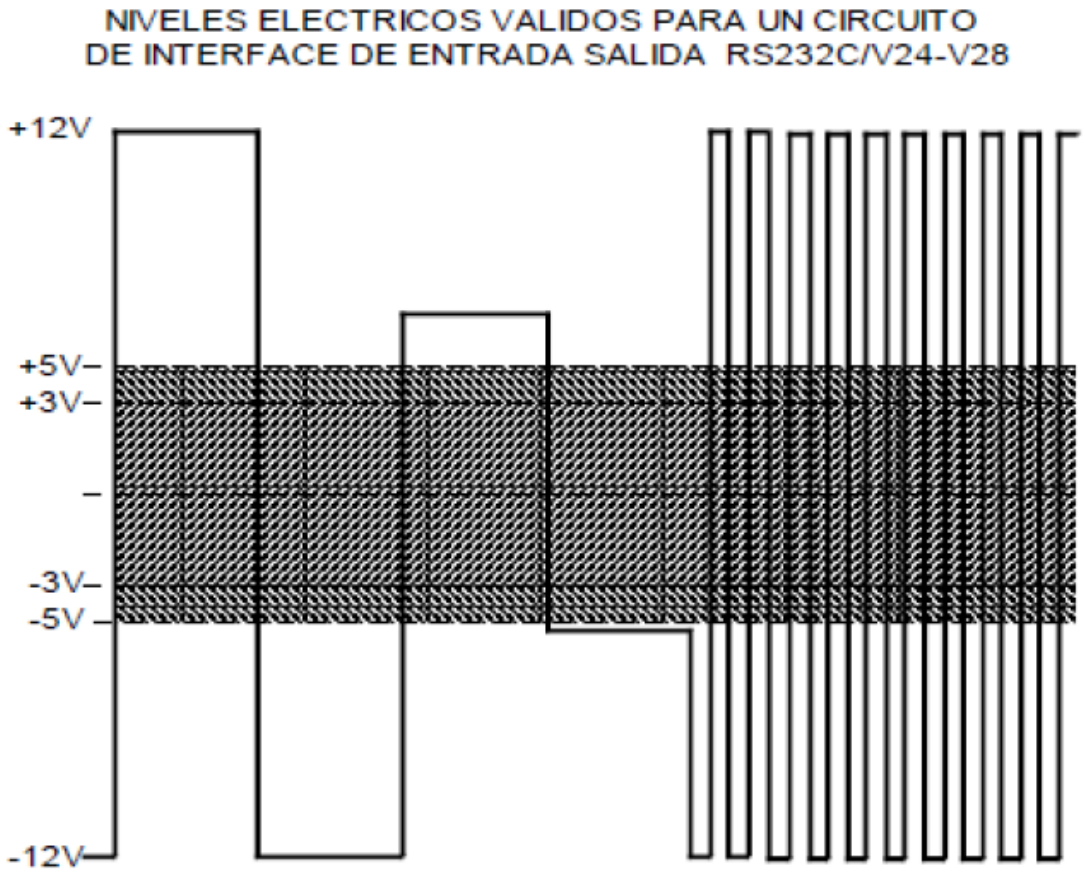


Figura 20. Especificaciones de voltaje del circuito RS-232.

Como puede observarse, el estándar considera uno lógico a todo valor de voltaje entre -5 y -12 volts, y un cero lógico a aquellos valores entre 5 y 12 volts positivos, con una zona de transición (zona de operación prohibida) de 5 a -5 volts.

La interfaz se utiliza a una razón de menos de 20Kbps para una distancia menor de 15m. En la práctica se pueden exceder estos límites utilizando cables de baja capacidad en entornos eléctricamente poco ruidosos. El protocolo RS-232 normal en el conector tipo DB-9 utilizado en esta práctica es el siguiente.

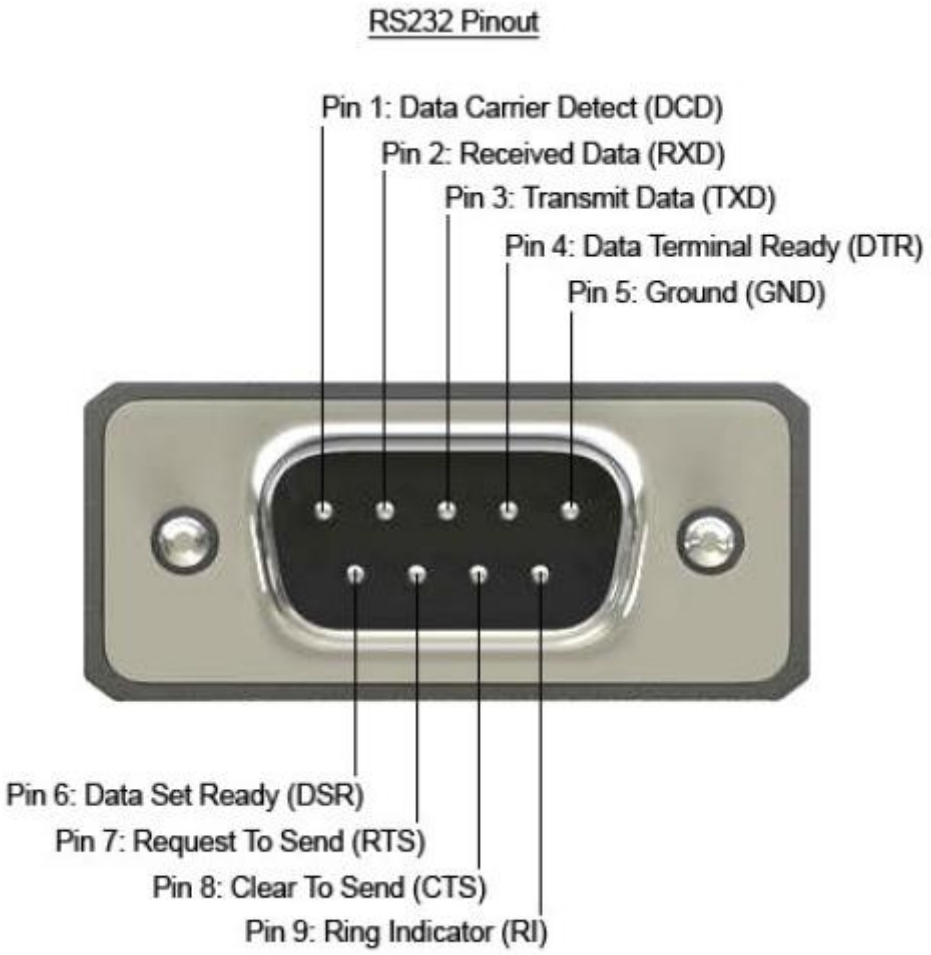


Figura 21. Conector del circuito RS-232 con indicación de cada uno de sus significados.

Capítulo 4
Desarrollo del software de Conexión

4.1 Introducción

En este capítulo se hará una profunda descripción del marco de trabajo utilizado para el desarrollo de la aplicación que controla la conexión con el guante y la conexión con TCP/IP hacia Matlab, para ello utilizaremos programación en C++ y librerías del guante. Luego se diseñará el entorno gráfico y el control de la mano virtual en el entorno de programación de Matlab-Simulink.

Se abordarán diferentes cuestiones desde el punto de vista de justificar las decisiones tomadas frente a otras alternativas, implementando los detalles de código fuente y de los objetivos del proyecto.

4.2 Desarrollo de la Aplicación en el software de programación Microsoft Visual C ++

El desarrollo de la aplicación no estaba demasiado claro cuando se establecieron los objetivos iniciales del proyecto. Esto se debía al desconocimiento del funcionamiento de las librerías y la compatibilidad en el software de programación de Matlab-Simulink. Por este motivo, se valoró como objetivo desarrollar una aplicación en lenguaje de programación en C++, donde se utilizaría protocolo TCP/IP para conectar las librerías del Guante por un lado y enviar datos al entorno Matlab-Simulink por otro lado.

4.3 Herramientas Utilizadas en Microsoft Visual C++

4.3.1 Programación Orientada a Objetos (POO)

La POO es una filosofía de programación que se basa en la utilización de objetos. El objetivo de la POO no es sino la utilización de un modelo de programación estructurada convencional: "imponer" una serie de normas de desarrollo que aseguren y faciliten la mantenibilidad y reusabilidad del código. Los mecanismos básicos de la POO son: objetos, mensajes, métodos y clases.

- **Objetos.** Un objeto es una entidad que tiene unos atributos particulares (datos) y unas formas de operar sobre ellos (los métodos o funciones miembro). Es decir, un objeto incluye, por una parte una serie de operaciones que definen su comportamiento, y una serie de variables manipuladas por esas funciones que definen su estado. Por ejemplo, una ventana Windows contendrá operaciones como "maximizar" y variables como "ancho" y "alto" de la ventana.

- **Mensajes.** En C++, un mensaje se corresponde con el nombre de uno de los métodos de un objeto. Cuando se pasa un mensaje a un objeto, éste responde ejecutando el código de la función asociada.
- **Método.** Un método (función miembro) se implementa dentro de un objeto y determina como tiene que actuar el objeto cuando se produce el mensaje asociado. En C++ un método se corresponde con la definición de la función miembro del objeto. La estructura más interna de un objeto está oculta, de tal manera que la única conexión con el exterior son los mensajes
- **Clases.** Una clase es la definición de un tipo de objetos. De esta manera, una clase "Empleado" representaría todos los empleados de una empresa, mientras que un objeto de esa clase (también denominado instancia) representaría a uno de esos empleados en particular.

Las principales características de la POO son: abstracción, encapsulamiento, herencia y polimorfismo:

- **Abstracción.** Es el mecanismo de diseño en la POO. Nos permite extraer de un conjunto de entidades datos y comportamientos comunes para almacenarlos en clases.

  Encapsulamiento. Mediante esta técnica conseguiremos que cada clase sea una caja negra, de tal manera que los objetos de esa clase se puedan manipular como unidades básicas. Los detalles de la implementación se encuentran dentro de la clase, mientras que desde el exterior, un objeto será simplemente una entidad que responde a una serie de mensajes públicos (también denominados interfaz de la clase). Existen tres tipos de interfaz de clases:
  - *Private:* es opcional y se toma por defecto desde el principio de la definición de la clase hasta que aparece otro especificador de acceso. Los atributos y los métodos sólo son accesibles desde la propia clase. Es el especificador de acceso más restrictivo. Los atributos son siempre private ya que sólo tendrá acceso a ellos la propia clase puesto que no forman parte de la interfaz del objeto.
  - *Public:* los atributos y los métodos son accesibles por cualquier clase. Es el especificador de acceso menos restrictivo. Los métodos son siempre public puesto que forman parte de la interfaz del objeto.
  - *Protected:* los atributos y métodos son accesibles sólo por la clase que los contiene y todas aquellas que hereden de ésta. Se encuentra entre private y public, en un nivel restrictivo intermedio.
- **Herencia**. Es el mecanismo que nos permite crear clases derivadas (especialización) a partir de clases bases (generalización). Es decir, podríamos tener la clase "Empleado" (clase base) y la clase "Vendedor" derivando de la anterior. Una librería de clases no es más que un conjunto de definiciones de clases interconectadas por múltiples relaciones de herencia.

- **Polimorfismo**. Esta característica nos permite disponer de múltiples implementaciones de un mismo método de clase, dependiendo de la clase en la que se realice. Es decir, podemos acceder a una variedad de métodos distintos (con el mismo nombre) mediante el mismo mecanismo de acceso. En C++ el polimorfismo se consigue mediante la definición de clases derivadas, funciones virtuales y el uso de punteros a objetos.

Otros dos conceptos muy importantes en la POO son relativos a la creación y destrucción de objetos. En lenguajes estructurados convencionales, cuando se define una variable se le reserva espacio en memoria y, si no se inicializa expresamente, se hace por defecto (por ejemplo, en C una variable global siempre se inicializa a 0, pero una automática no, por lo que si no se inicializa expresamente su contenido inicial será basura); por otra parte, cuando se destruye una variable (por que se abandona el ámbito de su definición - scope -) se libera la memoria que estaba ocupando. Si ahora hacemos el paralelismo obligado entre variables y objetos para los lenguajes POO nos daremos cuenta de que deben existir procedimientos especiales de construcción y destrucción de objetos. En concreto, cada clase tiene dos funciones miembro especiales denominadas constructor y destructor.

- **Constructor.** Función miembro que es automáticamente invocada cada vez que se define un objeto, su objetivo es la inicialización del mismo. Toma el mismo nombre que la clase, puede recibir parámetros y podemos tener varios constructores definidos.
- **Destructor.** Función miembro invocada automáticamente cada vez que se destruye un objeto. Su objetivo es realizar operaciones como liberación de memoria, cerrar ficheros abiertos, etc. Toma el mismo nombre de la clase comenzado primero por el carácter "~", no toma parámetros y no admite la sobrecarga (sólo puede existir uno en cada clase).

En muchos casos, para las clases más sencillas, podemos encontrar clases que no tienen constructor o destructor, ó ninguno de los dos. En C++, siempre existen constructores y destructores por defecto que realizan una inicialización/liberación estándar. Un esquema básico de creación de una clase se define en el siguiente esquema.

INTERFAZ CLASE NombreClase

METODOS

...

FIN NombreClase

IMPLEMENTACION CLASE NombreClase

ATRIBUTOS

...

METODOS

...

FIN NombreClase

4.3.2 Librerías Dinámicas

Las DLLs son trozos de código capaz de realizar determinadas operaciones, cuya funcionalidad puede ser utilizada desde otros ejecutables, ocupan una posición intermedia (diríamos que una solución de compromiso) entre otras posiciones extremas. Con las librerías estáticas comparten la característica de que es un trozo de código que acaba siendo incluido en el espacio del ejecutable que las utiliza. A su vez, comparten con los programas externos la característica de que constituyen ficheros distintos y físicamente independientes del ejecutable que los utiliza.

En realidad, la utilización de recursos pre-construidos por parte de un ejecutable puede realizarse de tres formas que podríamos resumir del siguiente modo:

- **Utilización de librerías estáticas**. Es el método tradicional. Son ficheros que en el momento de la construcción de la aplicación, son incluidos por el "Linker" en el propio ejecutable.
- **Utilización de librerías dinámicas.** En esta modalidad, los recursos ocupan un fichero independiente del ejecutable, que puede ser utilizado por cualquier aplicación que lo necesite. En algún momento, durante la carga del ejecutable, o posteriormente, en run-time, el ejecutable deberá integrar este bloque de código en su propio espacio, de forma que pueda acceder a los recursos contenidos en él.
- **Utilización de programas externos.** Es también un recurso utilizado desde siempre en programación. Un ejecutable puede llamar a ejecución a otro mediante mecanismos de varios tipos. El ejecutable llamado proporciona alguna funcionalidad antes de su terminación, y dispone de su propio espacio de ejecución independiente del programa que lo invocó.

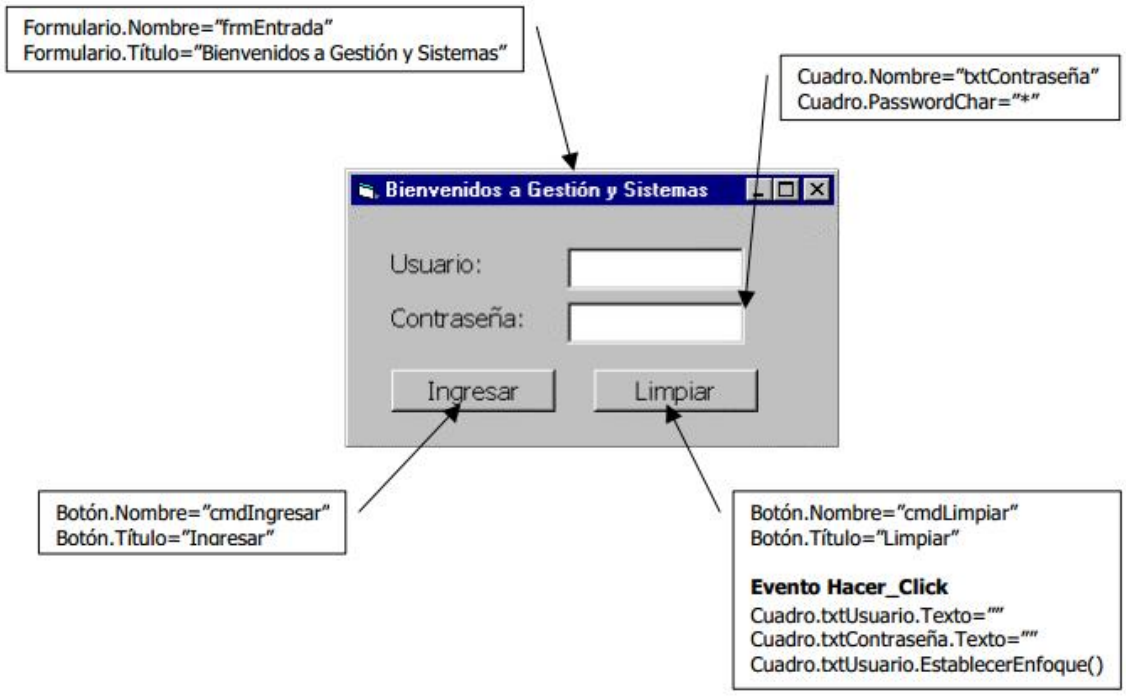
4.3.3 Aplicaciones con Visual Basic

Visual Basic es un ambiente gráfico de desarrollo de aplicaciones para el sistema operativo Microsoft Windows. Las aplicaciones creadas con Visual Basic están basadas en objetos y son manejadas por eventos. Visual Basic se deriva del lenguaje Basic, el cual es un lenguaje de programación estructurado. Sin embargo, Visual Basic emplea un modelo de programación manejada por eventos.

En las aplicaciones tradicionales o procedurales, es la aplicación quien controla que porciones de código se ejecuta, y la secuencia en que éste se ejecuta. La ejecución de la aplicación se inicia con la primera línea de código, y sigue una ruta predefinida a través de la aplicación, llamando procedimientos según sea necesario.

En las aplicaciones manejadas por eventos, la ejecución no sigue una ruta predefinida. En vez de esto, se ejecutan diferentes secciones de código en respuesta a eventos. Los eventos se desencadenan por acciones del usuario, por mensajes del sistema o de otras aplicaciones. La secuencia de eventos determina la secuencia en que el código se ejecuta. Es por esto que la ruta que sigue el código de la aplicación es diferente cada vez que se ejecuta el programa. Una parte esencial de la programación manejada por eventos es el escribir código que responda a los posibles eventos que pueden ocurrir en una aplicación. Visual Basic facilita la implementación del modelo de programación manejada por eventos.

Toda aplicación necesita una interfaz de usuario, la parte visual a través de la cual el usuario interactúa con la aplicación. Los bloques básicos de construcción de una interfaz de usuario son los formularios y los controles. Visual Basic utiliza técnicas de programación visual para diseñar las aplicaciones. Un ejemplo básico se describe en la siguiente ventana.



4.4 Desarrollo de la aplicación conexión TCP/IP

El programa creado para llevar a cabo la aplicación consta de tres partes, la clase guante, la clase TCP/IP, el programa principal. Para ello se utilizará la librería dinámica fgolved.dll, ésta se encarga de hacer de intermediario para poder acceder a los datos del guante y a sus drivers. El fabricante 5DT (Fifth Dimension Technologies) nos provee de los archivos para la programación en lenguaje Linux y C++, en nuestro caso solo utilizaremos C++. Para utilizar la librería en la aplicación es necesario ubicar el archivo en el mismo directorio en el que se encuentra el programa.

4.4.1 Clase Guante

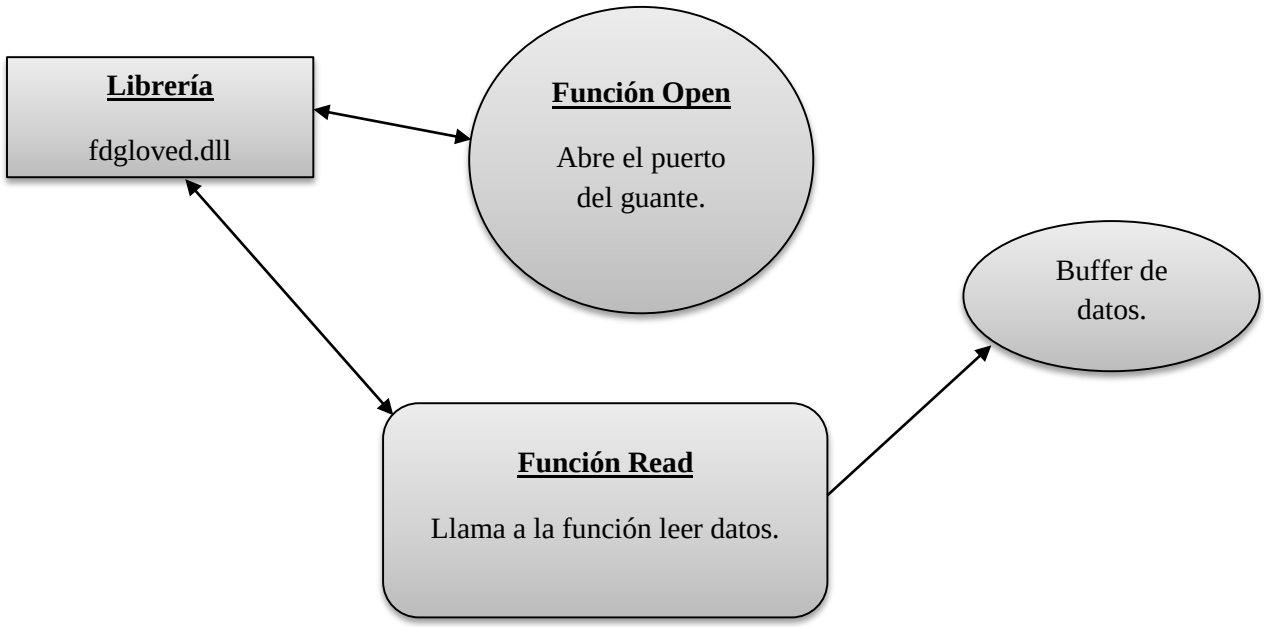
La clase Guante, como se ha dicho mencionado previamente, fue diseñada para gestionar todo lo relacionado con el guante (a decir configuración de parámetros, apertura del puerto y lectura de datos del guante). Esto fue realizado a través de una interfaz software de métodos que se mencionan a continuación:

- **Función Open**

Gestiona la correcta apertura del puerto de comunicación entre el guante y el ordenador, esta funcionalidad fue implementada utilizando las funciones de base de la librería del guante. De manera general, se ejecutan las funciones de la librería y se realiza el correspondiente control de errores, si el resultado es correcto entonces la aplicación puede continuar. En todo caso, se procede a notificar el estado de la conexión en la interfaz de usuario (ventana de dialogo).

- **Función Read**

Se encarga de leer los datos del guante y almacenarlos en un buffer, para ello se accederá a una función de las librerías que nos devolverá una cadena con 16 valores en formato "unsigned short" (2 byte por cada dato), y serán almacenadas en un búfer para su posterior utilización.



4.4.2 Clase TCP/IP

La clase TCP/IP fue diseñada para gestionar la comunicación TCP/IP entre el servidor desarrollado en MS-Visual C++ y Matlab/Simulink, en este caso se implementó una interfaz con 4 funciones que pasaremos a explicar. Cabe recalcar que

en este caso se utilizó las librería dinámica Winsock para implementar la conexión TCP/IP. Esta librería incluye soporte para envío y recepción de paquetes de datos a través de sockets.

- **Función Connect**

Esta función se encarga de crear el socket servidor, a través de parámetros como tipo de familia, dirección IP y puerto de enlace. Si la conexión resulto exitosa se devolverá un mensaje de éxito.

- **Función Listen**

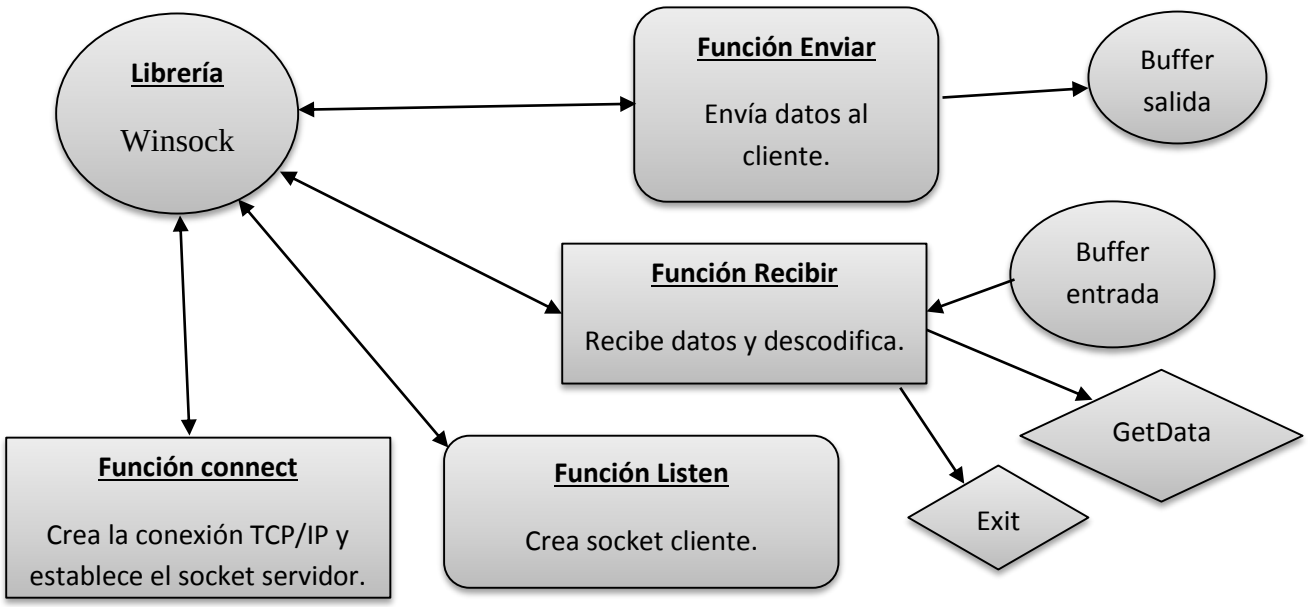
Esta función se encarga de poner en modo escucha al servidor, esperando a que el cliente envíe una petición de establecimiento de conexión. La conexión se almacena en el socket cliente y se devuelve un mensaje de éxito.

- **Función Recibir**

Esta función crea un buffer de entrada de datos donde el cliente, en este caso Matlab, envía los comandos para su posterior descodificación y almacenamiento en una variable. Los comandos son "GetData" (petición de nuevos datos) y "Exit" (No enviar más datos y salir del sistema).

- **Función Enviar**

Esta función tiene como propósito enviar los datos que recibió del guante a través del socket cliente. Se enviarán en formato "char" y se almacenaran en el buffer de Matlab.



4.4.3 Programa Principal

El programa principal se creará en el entorno de aplicaciones de Visual C++, procederemos a detallar cada uno de los pasos de la creación de la aplicación para este proyecto.

- **Interfaz de usuario**

El área de trabajo de la interfaz de usuario se caracteriza por contener los elementos de interacción con el usuario. Para esta ocasión hemos añadido tres botones, dos de pulsación habilitada y una de que se habilitará cuando las conexiones se realicen con éxito. Por otra parte añadimos un cuadro de texto donde se informara al usuario de los estados de los eventos. En la siguiente imagen se muestra el entorno de nuestra aplicación.

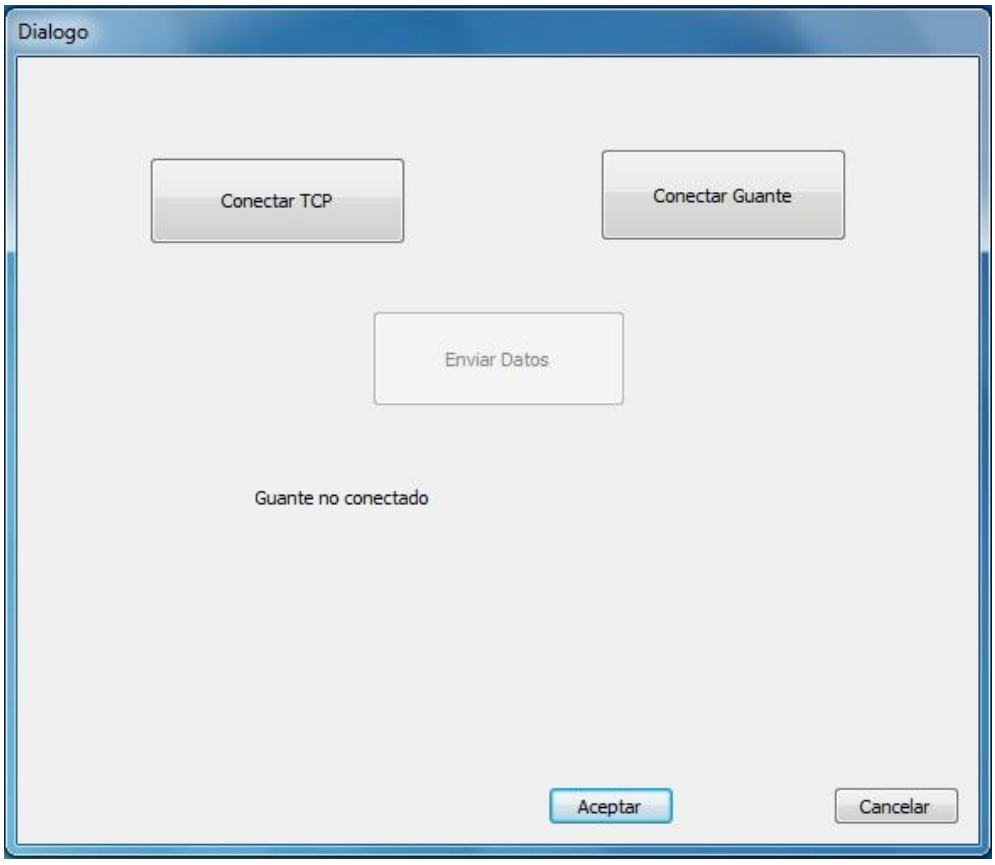


Figura 21. Ventana de la aplicación servidor del guante.

- **Código para los eventos**

En este caso se crearon 3 eventos para dar acciones a los diferentes botones de la aplicación. El primer evento se caracteriza por inicializar el guante y verificar si el guante se conectó con éxito y devolver al usuario información del mismo.

El segundo evento inicializa la conexión TCP/IP y genera el socket servidor y se pone en espera hasta que el cliente haya realizado la petición de conectarse, después si

la conexión es exitosa se informará al usuario y se activará el tercer botón para el siguiente evento.

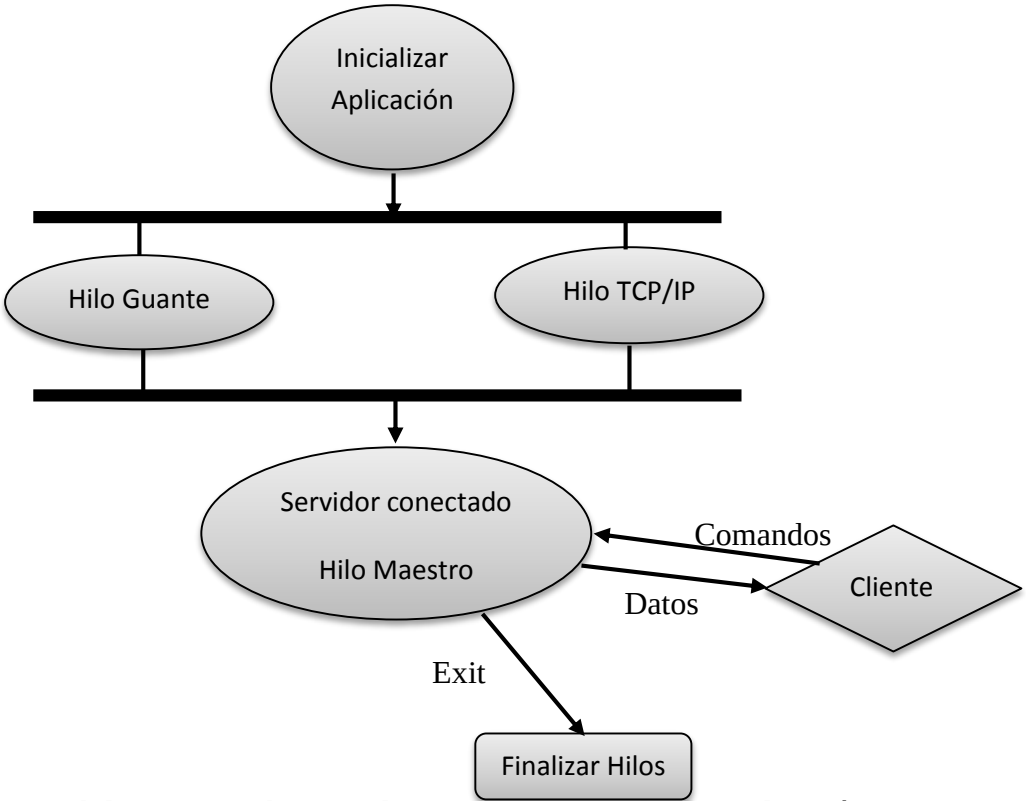
El tercer evento activa el hilo principal del programa que se caracteriza por enviar los datos hacia Matlab.

- **Hilos de trabajo**

Para esta aplicación utilizaremos hilos de ejecución que se irán activando mediante una serie de parámetros programados previamente. Para entenderlo mejor procedemos a explicar que es un hilo de ejecución.

Un hilo de ejecución, hebra o subproceso es la unidad de procesamiento más pequeña que puede ser planificada por un sistema operativo.

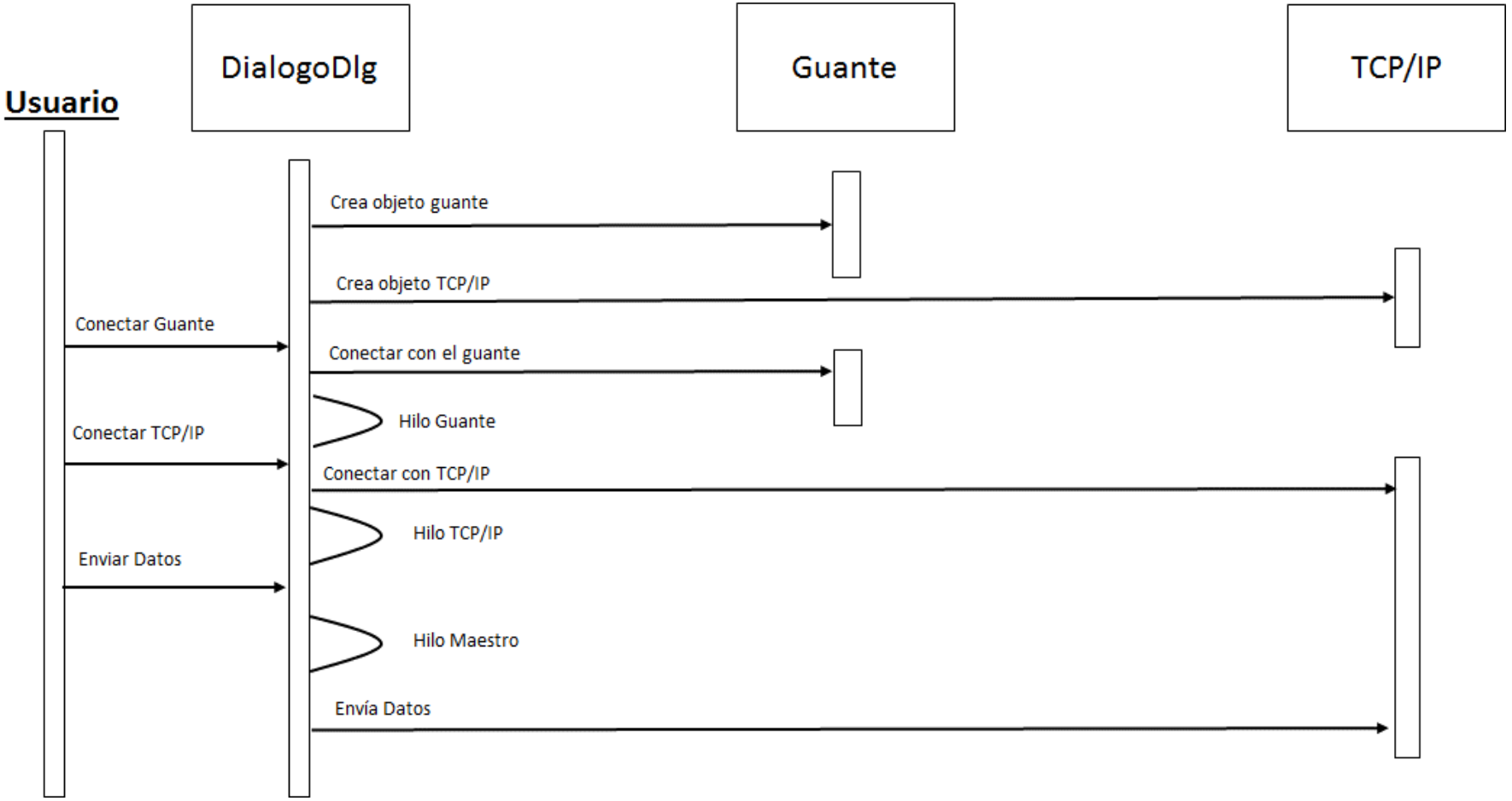
La creación de un nuevo hilo es una característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente. En nuestro caso estos hilos de ejecución nos ayudarán a interconectar los diferentes procesos que tiene el programa para enviar correctamente los datos.



- El primer hilo es creado en el primer evento de la aplicación y su funcionamiento es llamar a la función lectura de datos del guante y activar una bandera que especifica cuando hay datos nuevos en el buffer.

- El segundo hilo se activa en el segundo evento de la aplicación y su función es activar la recepción de datos y descodificar los datos recibidos para realizar las diferentes acciones.
- El tercer hilo se activa con la activación del tercer evento y su función se radica en poner en espera los demás hilos hasta haber trasladado los datos del buffer de entrada al buffer de salida y su posterior envío hacia Matlab. Luego se activaran de nuevo los demás hilos a espera de una nueva orden.

Todo este procedimiento se puede ver en el siguiente diagrama de secuencia:


Usuario
DialogoDlg
Guante
TCP/IP
Crea objeto guante
Crea objeto TCP/IP
Conectar Guante
Conectar con el guante
Hilo Guante
Conectar TCP/IP
Conectar con TCP/IP
Hilo TCP/IP
Enviar Datos
Hilo Maestro
Envía Datos

Capítulo 5
Simulación y entorno visual

5.1 Introducción

En este apartado se pretende realizar, por un lado la recepción de datos y reacondicionarlo en un bloque Simulink, por otro lado se diseñará un entorno visual en el programa V-Realm Builder para la estructura de la mano. Todo ello se ensamblará y se creará un entorno de control para la simulación en tiempo real de los movimientos del guante.

5.2 Herramientas Utilizadas en Matlab

5.2.1 Simulink

Simulink es una toolbox de Matlab que se utiliza para simular, modelar y analizar sistemas dinámicos. Se pueden simular modelos lineales o no lineales, en tiempo continuo o no continuo. Además permite un interfaz gráfico sencillo y una librería de objetos muy completa con fuentes de datos, operadores, sistemas, y sumideros de datos que permitan por ejemplo visualizar resultados. También permite el desarrollo de nuevas funciones, por ejemplo mediante las S-function. Se permite también el manejo de parámetros de cada modelo desde el espacio de trabajo de Matlab.

5.2.2 S-function

La S-function es un bloque que permite extender funciones a Simulink. Permite una adaptación del funcionamiento de archivos “dll” a Simulink. Una S-function se puede escribir en C o Matlab. Para utilizar la S-función en Simulink tiene que ser compilada como MEX-file, se puede hacer mediante la orden “mex archivo.c”. Las S-function escritas en C se dividen en tres partes importantes, una para inicializar, una que se ejecutará de manera periódica durante la simulación y una que se ejecutará para finalizar correctamente la simulación. En la inicialización se declara el funcionamiento del comienzo, número de entradas y las salidas y el tiempo de muestreo del bloque.

Cada una de las partes es implementada sobre algunas funciones cuyo funcionamiento describimos a continuación.

Función	Objetivo
mdlInitializeSizes	Define el número de puertos de entrada y salida del bloque, así como la dimensión de estos.
mInitializeSamleTime	Define el tiempo de actualización de los valores de entrada y salida al bloque

mdlStart	Solo se ejecuta al iniciar la simulación
mdlOutputs	Calcula las salidas en cada actualización
mdlUpdate	Actualiza valores intermedios en cada actualización
mdlTerminate	Solo se ejecuta al terminar la simulación

Tabla 2. Funciones ejecutadas en una S-function

En nuestro caso no necesitamos implementar todas las funciones para desarrollar las S-function.

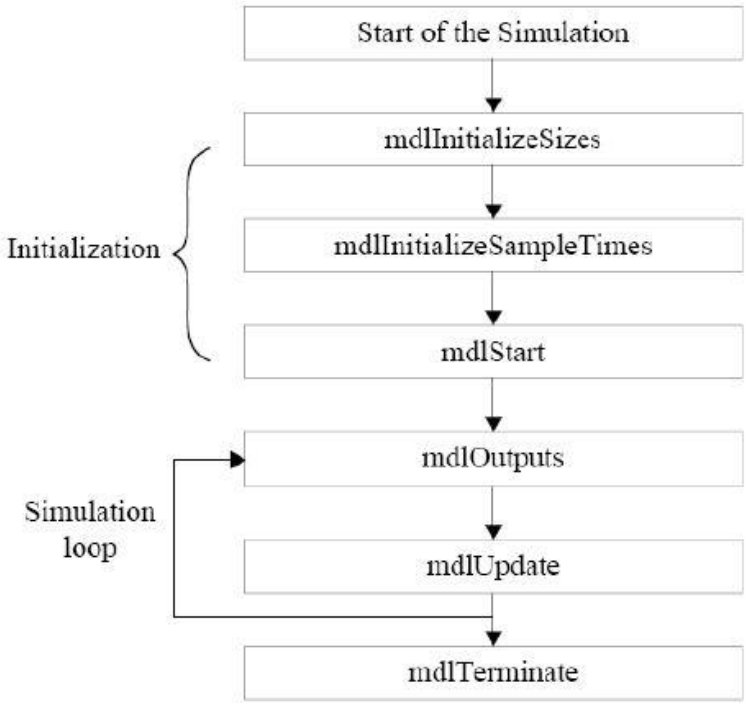


Figura 23. Funcionamiento de una S-function

5.2.3 Virtual Reality Toolbox

El Toolbox de Realidad Virtual de Matlab es una solución para visualizar e interactuar con sistemas dinámicos en un ambiente tridimensional de realidad virtual. Éste extiende las capacidades de Matlab y Simulink mediante el uso de mundos con gráficos virtuales. Hay dos formas de utilizar el Toolbox de realidad virtual; mediante la interface de Matlab (líneas de comandos y funciones) o mediante la interface de Simulink (mediante bloques y así observar el comportamiento de sistemas dinámicos) o bien se pueden utilizar los dos a la vez en caso de que sea absolutamente necesario. Generalmente la interface de Simulink suele ser más rápida ya que el procesamiento de los bloques se hace "paralelamente" mientras que la interface de Matlab se hace en serie (línea por línea), el uso de cualquiera de las dos interfaces depende de la aplicación que se desee hacer.

Conceptos claves que definen mejor la herramienta utilizada.

- **VRML (Virtual Reality Modeling Language):** Es un formato estándar para la creación o modelación de mundos virtuales.
- **VRML editor:** Editor de mundos virtuales estándar.
- **VRML viewer:** Visor de mundos virtuales estándar.

Creación de un mundo virtual.

Una vez se haya instalado el visor y el editor de VRML de Matlab, se accederá a un espacio de creación donde iremos añadiendo diferentes módulos enlazados que contendrán diferentes atributos como rotación, escala, traslación, etc. Se definirá cada atributo minuciosamente hasta alcanzar la construcción de todo el modelo.

En la siguiente imagen se muestra el entorno de creación del mundo virtual.

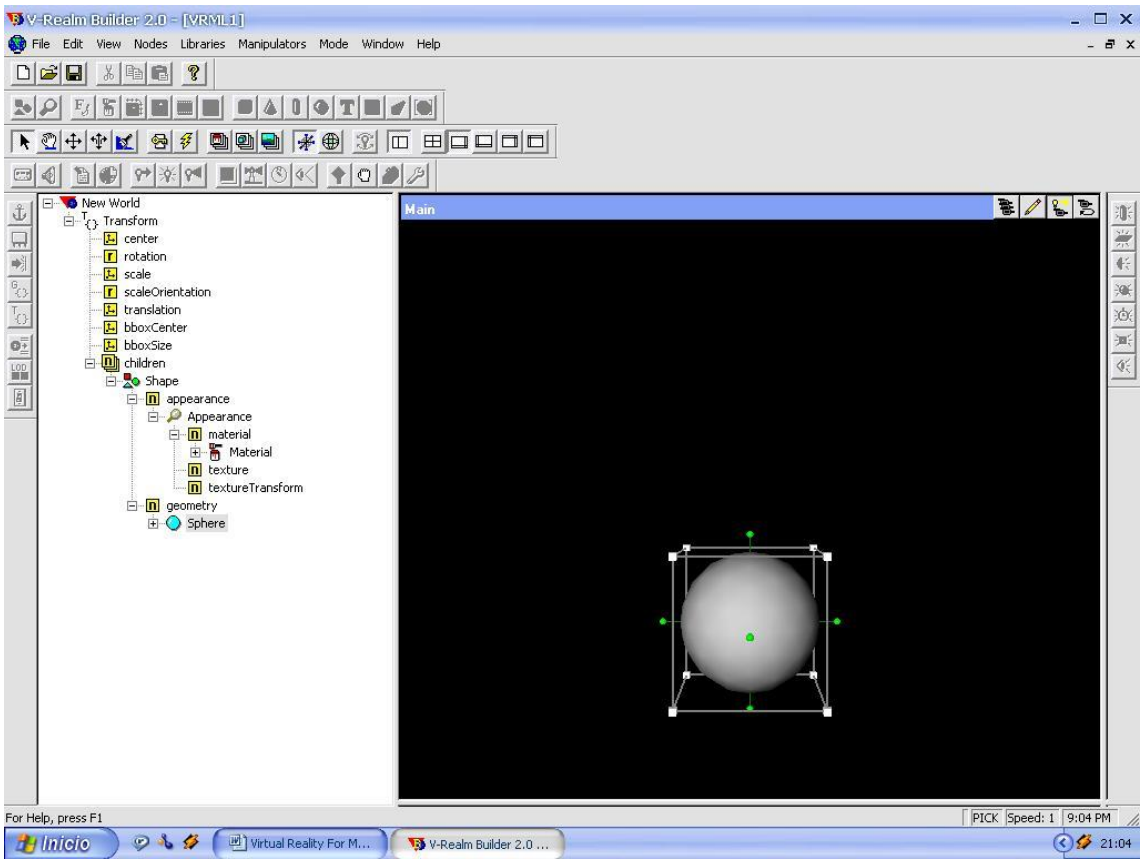


Figura 24. Entorno de creación de un mundo virtual

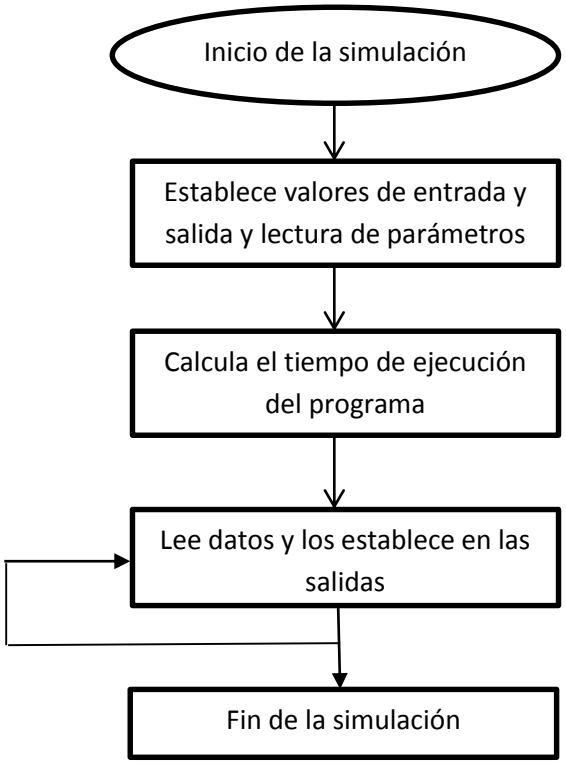
5.3 Desarrollo de la Simulación

Para el desarrollo de la simulación se procederá a crear un bloque Simulink donde se recibirán los datos, además se acondicionarán estos datos para trasladarlos a nuestro espacio virtual. El espacio virtual se creará en formato VRML y se creará un bloque VR link con la diferentes entradas para el movimiento de cada dedo. Pasaremos a explicar los pasos de la creación de nuestro entorno de simulación.

5.3.1 Creación bloque S-Function

Este bloque se desarrolló para interactuar con los funciones de nuestra programa en C++, ya que en el entorno de Simulink no existe alguna herramienta que satisficiera las necesidades que requeríamos. Esto se debe a que necesitábamos mandar comandos específicos para la petición de datos a nuestro programa en C++. Se optó por construir el bloque en una S-Function Matlab ya que reunía las capacidades de Matlab y Simulink juntas.

Para la creación de este bloque se partió de una plantilla S-function Matlab level 2 y se estructuró en función de lo que necesitábamos, al ser un bloque simple sólo necesitamos de las funciones principales y de inicio, además de la actualización de estados de la salida para la visualización de los datos entregados. Este bloque recibirá como parámetro el puerto de enlace y una entrada de marcha o parada. Cabe destacar que la recepción será asíncrona así que diseñamos nuestra función para actualizar los datos cuando éste lo reciba y mientras tanto permanezcan en salida los datos antiguos. En el siguiente diagrama se muestra la estructura de nuestra s-function.



5.3.2 Creación del entorno de simulación

Para la creación del entorno de simulación utilizaremos el toolbox Virtual Reality, que es una extensión de Simulink para la creación de espacios tridimensionales, que luego podrán ser simulados en el espacio de Simulink. Este sistema consta de variables de entrada para el movimiento de cada articulación de la mano. El sistema virtual quedará representado como en la siguiente imagen.

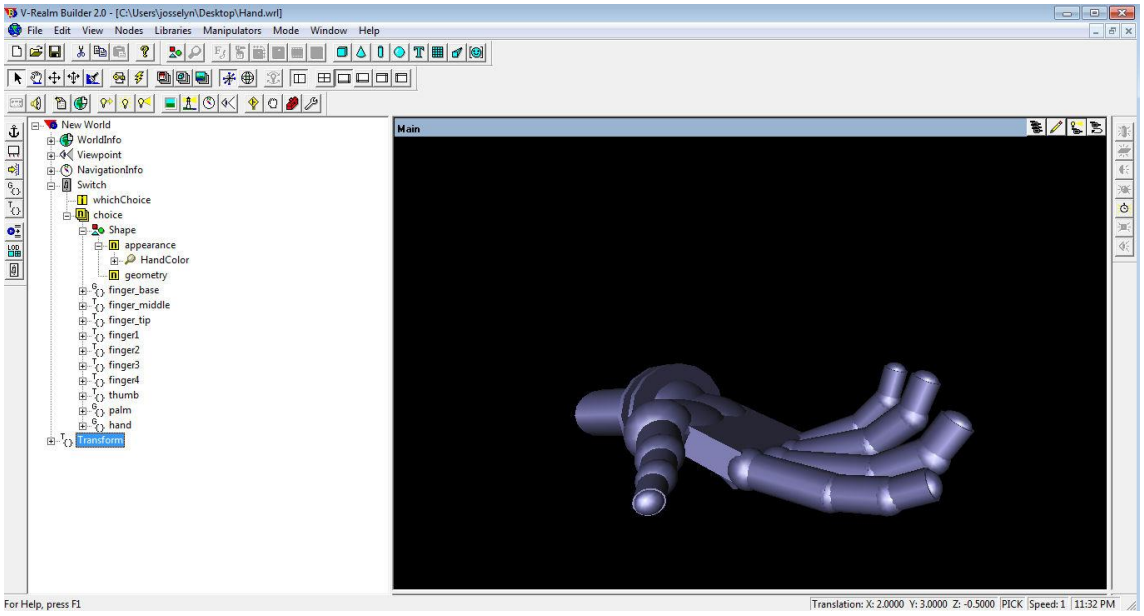
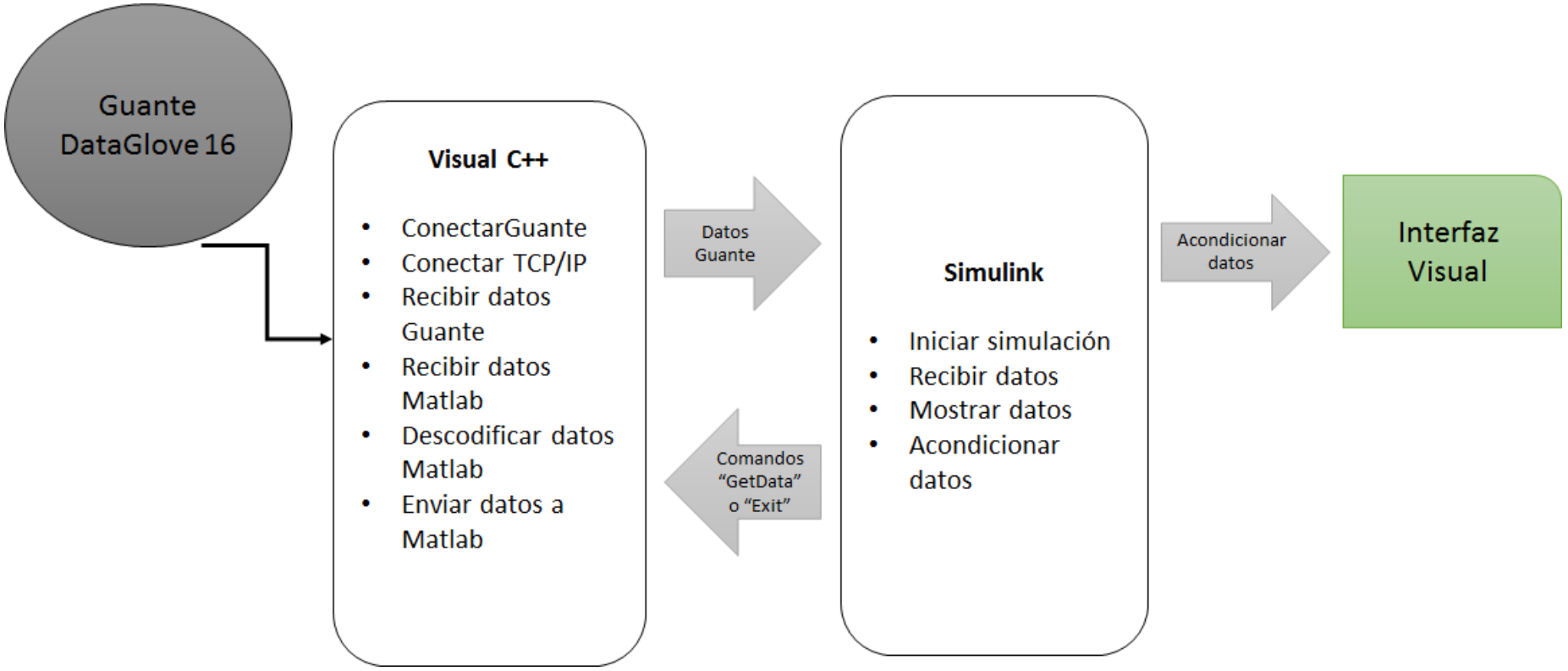


Figura 25. Diseño mano

Para el diseño esta mano se planteó utilizar el Toolbox SimMechanics de Simulink. Esta herramienta se utiliza para modelar y simular sistemas mecánicos de forma muy fácil y eficiente en Simulink con gráficos de diseño más reales. Pero nos decantamos por utilizar el Toolbox Virtual Reality ya que SimMechanics requiere de un estudio más amplio y de la utilización de otro software, que sale del alcance de este proyecto.

5.4 Estado de funcionamiento en línea

Este proyecto tuvo su función en interconectar los datos del guante en el entorno de Matlab, para ellos hemos utilizado múltiples herramientas como la programación en C++ y programación en Matlab y un diseño gráfico en VRLM. Su función básica era la de extraer los datos del guante y llevarlos mediante protocolo TCP/IP hacia Matlab-Simulink, donde se acondicionarían para controlar el movimiento de la mano virtual. Por lo tanto la comunicación entre programas se realizará por una serie de reglas definidas como la recepción asíncrona, el envío de comandos y el acondicionamiento de los datos.


Guante
DataGlove 16
Visual C++
ConectarGuante
Conectar TCP/IP
Recibir datos Guante
Recibir datos Matlab
Descodificar datos Matlab
Enviar datos a Matlab
Datos Guante
Comandos "GetData" o "Exit"
Simulink
Iniciar simulación
Recibir datos
Mostrar datos
Acondicionar datos
Acondicionar datos
Interfaz Visual

Capítulo 6
Conclusiones y futuros desarrollos

6.1 Conclusiones

La interrelación entre diversas plataformas de software permitió realizar la construcción de una interfaz gráfica con realidad virtual, mediante protocolos y programación se logró conectar el guante al entorno virtual.

Podremos decir que el movimiento de la mano no se traduce en el entorno virtual de manera exacta, sino que presenta un cierto error en la medida. Esto es debido a la utilización de sensores de flexión que presentan cierto ruido en la medida, además de que los sensores se encuentran entre articulaciones resultando una medición pobre.

Otra cosa a tener en cuenta es que el guante es un dispositivo bastante antiguo y los nuevos modelos ya presentan avances de medición con sensores de giroscópicos y trasmisión inalámbrica e incluso otros dispositivos con tecnología infrarroja, como el dispositivo Leap Motion, obtienen una mayor precisión en las medidas.

En definitiva los guantes de realidad virtual todavía representan una opción rentable en sistemas de control y visualización virtual, pero poco a poco se van quedando en desuso con la aparición de nuevas tecnologías, y por lo tanto habrá que adaptarse a los nuevos dispositivos.

6.2 Futuros desarrollos

Como ya hemos comentado anteriormente este proyecto está diseñado para el ámbito de sistemas virtuales, en espacios donde queremos simular y comprobar de antemano como va funcionar un sistema. Sin embargo ya hemos visto que este dispositivo puede adaptarse a múltiples funciones, desde la composición de música al control por ordenador, esta aplicación, a pesar de haberse diseñado para visualizar el movimiento de la mano, con varios cambios en su programación y comunicación podría controlar otros elementos ya sean industriales (trenes de carga, pinzas, carretillas, detección de objetos etc.) como en el ámbito del ocio (juguetes, control de ordenadores, video juegos, etc.).

Anexo A
Funciones de referencia

---

fdGlove * fdopen (char * pPort)
Inicializa el dispositivo de los guantes en el puerto especificado**.**

Valor de retorno
Devuelve un puntero al dispositivo guante (fdGlove *). Se devuelve NULL si ocurre un error.

Parámetro
pPort
Puntero a una cadena terminada en cero ASCII que contiene el nombre del puerto de comunicación.
Los valores válidos en rango de Windows son de "COM1" a "COM8". Nombres de puerto de Unix/Linux serán diferentes.

Observaciones
No intente alterar el contenido del puntero devuelto directamente, utilice las funciones previstas en su lugar.

---

int fdClose (fdGlove * pFG)
Libera el puerto del dispositivo guante y las comunicaciones.

Valor de retorno
Devuelve distinto de cero en caso de éxito, cero si se ha producido un error.

Parámetros
pFG
Puntero a un dispositivo de guante. éste es el valor devuelto por fdopen ().

Observaciones
Es importante llamar a esta función cuando haya terminado de usar el guante.

---

int fdGetGloveHand (fdGlove * pFG)
Obtiene la lateralidad (izquierda o derecha) del guante.

Valor de retorno
Devoluciones sea *FD_HAND_LEFT* o *FD_HAND_RIGHT*, tal como se define por el tipo *EfdGloveHand*.

Parámetros
pFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdOpen ().

Observaciones

Ninguno

---

int fdGetGloveType (fdGlove * pFG)
Obtiene el tipo del guante conectado actualmente.

Valor de retorno
Devuelve uno de *FD_GLOVENONE, FD_GLOVE7, FD_GLOVE7W, FD_GLOVE16 o FD_GLOVE16W*, según lo definido por las EfdGloveTypes tipo enumerado.

Parámetros.
pFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Observaciones
FD_GLOVE7 y *FD_GLOVE7W* refieren a 5 sensores + 2 (ángulos de inclinación) del guante original del sensor (5DT datos Guante 5). El sufijo W indica un modelo inalámbrico. FD_GLOVE16 y FD_GLOVE16W refieren al guante de 16 sensores. Con el fin de dar cabida a ambos tipos de guantes los fdGetNumSensors () función devuelve actualmente 18 sensores. Los dos sensores adicionales se definen como los ángulos de inclinación originales que no están presentes en el guante 16 - sensor. Véase la descripción de fdGetNumSensors () para más detalles.

---

int fdGetNumSensors (fdGlove * pFG)
Obtiene el número de sensores disponibles valorando el guante a disposición.

Valor de retorno
Devuelve el número de sensores. Actualmente se fija en 18, pero sobre controladores futuros pueden diferir.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Observaciones
Aunque el guante de 5 sensores puede medir sólo la flexión media, el driver intentará rellenar los valores que faltan. El número de sensores devuelto por lo tanto puede ser de una dimensión superior. Los *EfdSensors* tipo enumeradas define el mapeo dedo para cada sensor.

---

void fdGetSensorRawAll (fdGlove * pFG , unsigned short *pData)
Obtiene los valores más recientes de los sensores en un rango de valores del guante.

Valor de retorno
Ninguno.

Parámetros
pFG

Puntero a un dispositivo de guante. Éste es el valor devuelto por fdOpen ().

pData

Puntero a una matriz de enteros de 16 bits que contendrán los valores de los sensores. El tamaño de la matriz siempre debe coincidir con el valor devuelto por fdGetNumSensors ().

Observaciones

Actualmente las muestras de sensores son todos los valores sin signo de 12 bits. Por consiguiente, el rango es de 0 a 4095. Obsérvese que éste no es el rango dinámico de los sensores. No se puede encontrar valores asociados con cada sensor de desplazamiento. Los EfdSensors tipo enumeradas define el mapeo dedo para cada sensor.

---

int fdGetGloveType (fdGlove * pFG)

Obtiene el tipo del guante conectado actualmente.

Valor de retorno

Devuelve uno de *FD_GLOVENONE, FD_GLOVE7, FD_GLOVE7W, FD_GLOVE16 o FD_GLOVE16W*, según lo definido por las EfdGloveTypes tipo enumerado.

Parámetros

pFG

Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Observaciones

FD_GLOVE7 y *FD_GLOVE7W* refieren al 5 + 2 (ángulos de inclinación) guante original del sensor (5DT DataGlove 5). El sufijo W indica un modelo inalámbrico. *FD_GLOVE16* y *FD_GLOVE16W* refieren al guante 16 sensores. Con el fin de dar cabida a ambos tipos de guantes los fdGetNumSensors () función devuelve actualmente 18 sensores. Los dos sensores adicionales se definen como los ángulos de inclinación originales que no están presentes en el guante 16-sensor. Véase la descripción de fdGetNumSensors () para más detalles.

---

int fdGetNumSensors (fdGlove * pFG)

Obtiene el número de sensores disponibles que el driver puede poner a disposición.

Valor de retorno

Devuelve el número de sensores. Actualmente se fija en 18 años, pero sobre controladores futuros pueden diferir.

Parámetros

pFG

Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Observaciones

Aunque el guante 5-sensor puede medir sólo la flexión media, el driver intentará rellenar los valores que faltan. El número de sensores devuelto por lo tanto puede ser de

una dimensión superior. Los EfdSensors tipo enumeradas define el mapeo dedo para cada sensor.

---

void fdSetSensorRawAll (fdGlove * PFG, unsigned short pData *)
Fuerza el valor bruto de todos los sensores.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

pData
Puntero a una matriz de enteros de 16 bits que contendrán los valores de los sensores primas. El tamaño de la matriz siempre debe coincidir con el valor devuelto por fdGetNumSensors ().

Observaciones
Actualmente las muestras de sensores primas son todos los valores sin signo de 12 bits. Por lo tanto, de 0 a 4095. Los EfdSensors tipo enumerado El rango es define la correspondencia dedo para cada sensor. Forzar a un valor del sensor resultará en una salida de crudo y escalado distinto al predeterminado cero. Los valores que se pueden asignar serán reemplazados, lo que hace el vacío valor forzado.

---

void fdSetSensorRaw (fdGlove * PFG, int nSensor, unsigned short nRaw)
Fuerza el valor nominal de un sensor específico.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Falla del sensor
Índice del sensor que se está fijando. El valor debe estar en el intervalo dado por los EfdSensors tipo enumerado, o alternativamente desde cero hasta el valor devuelto por fdGetNumSensors () menos uno.

nRaw
16 bit de valor en bruto del sensor. Si el sensor está sin asignar, los cálculos de escalamiento procederán de forma normal.

Observaciones
Los EfdSensors tipo enumeradas define el mapeo dedo para cada sensor. Esta función sólo es útil para los sensores que no se pueden asignar por un dispositivo de hardware

específico. Forzar a un valor del sensor resultará en una salida de crudo y escalado distinto al predeterminado cero.

Los valores que se pueden asignar serán reemplazados, lo que hace el vacío valor forzado.

---

void fdGetSensorScaledAll (fdGlove * PFG, float * pData)
Obtiene los valores de los sensores (auto-calibrado) escaladas más recientes del guante conectado actualmente.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

pData
Puntero a una serie de números de punto flotante que contendrán los valores de los sensores a escala.
El tamaño de la matriz siempre debe coincidir con el valor devuelto por fdGetNumSensors ().

Observaciones
El alcance del sensor es un valor de cero al valor definido por el (funciones) *fdSetSensorMax () y fdSetSensorMaxAll*. Los valores predeterminados del Data Glove es un rango de [0 ... 1].
El proceso de calibración automática se describe en la sección 2.10. Los EfdSensors tipo enumerado define el mapeo dedo para cada sensor.

---

float fdGetSensorScaled (fdGlove * PFG, int nSensor)
Obtiene el valor más reciente reducido (auto-calibrado) para un sensor específico de la guantera conectado actualmente.

Valor de Retorno
Devuelve un valor de sensor de punto flotante.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Fallo del sensor
Índice del sensor que se está fijando. El valor debe estar en el intervalo dado por los EfdSensors tipo enumerado, o alternativamente desde cero hasta el valor devuelto por fdGetNumSensors () menos uno.

Observaciones

El alcance del sensor es un valor de cero al valor definido por el (funciones) fdSetSensorMax () y fdSetSensorMaxAll. Los valores predeterminados del Data Glove a un rango de [0 ... 1]. El proceso de calibración automática se describe en la sección 6. Los EfdSensors tipo enumerado define el mapeo dedo para cada sensor.

---

int fdGetNumGestures (fdGlove * pFG)
Obtiene el número de gestos disponibles que pueden ser reconocidos por el driver del guante.

Valor de Retorno
Devuelve el número de gestos disponibles. Actualmente se admiten 16 gestos diferentes. Consulte la sección 5 para más detalles.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Observaciones
Ninguno.

---

int fdGetGesture (fdGlove * PFG)
Obtiene el gesto actual se está realizando.

Valor de Retorno
Devuelve el gesto actual se está realizando. Consulte la sección 5 para más detalles.

Parámetros PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Observaciones
Ninguno.

---

void fdGetCalibrationAll (fdGlove * PFG, unsigned short * pUpper, unsigned short * pLower)
Obtiene la configuración automática de calibración actual del controlador.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

pUpper y pLower
Las matrices de enteros sin signo de 16 bits que contendrán los valores máximos y mínimos del sensor prima. El tamaño de cada matriz siempre debe coincidir con el valor devuelto por fdGetNumSensors (). Consulte la sección 6 para más detalles.

Observaciones
Ninguno.

---

void fdGetCalibration (fdGlove * PFG, int nSensor, unsigned short * pUpper, unsigned short * pLower)
Obtiene la configuración automática de calibración actual del controlador para un sensor específico.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Falla del sensor
Índice del sensor que se está fijando. El valor debe estar en el intervalo dado por los EfdSensors tipo enumerado, o alternativamente desde cero hasta el valor devuelto por fdGetNumSensors () menos uno.

Pupper y plower
Los punteros a enteros sin signo de 16 bits que contienen los valores máximos y mínimos del sensor prima. Consulte la sección 6 para más detalles.

Observaciones
Ninguna.

---

void fdSetCalibrationAll (fdGlove * pFG, unsigned short * pUpper, unsigned short * pLower)
Restablece la configuración de auto-calibración actuales del controlador a los valores definidos por el usuario.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

pUpper y pLower
Las matrices de enteros sin signo de 16 bits que contienen los valores máximos y mínimos del sensor prima. El tamaño de cada matriz siempre debe coincidir con el valor devuelto por fdGetNumSensors (). Consulte la sección 6 para más detalles.

Observaciones
Para los sensores sin asignar sería razonable para establecer los ajustes de calibración superior e inferior por encima y por debajo del valor bruto forzada con fdSetSensorRaw () y fdSetSensorRawAll ().

---

void fdSetCalibration (fdGlove * PFG, int nSensor, unsigned short nUpper, unsigned short nLower)

Restablece la configuración de auto-calibración actuales del controlador para un sensor específico a los valores definidos por el usuario.

Regreso valor

Ninguno.

Parámetros

PFG

Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Falla del sensor

Índice del sensor que se está fijando. El valor debe estar en el intervalo dado por los EfdSensors tipo enumerado, o alternativamente desde cero hasta el valor devuelto por fdGetNumSensors () menos uno.

nUpper y nLower

Enteros sin signo de 16 bits que contienen los valores máximos y mínimos del sensor prima. Consulte la sección 6 para más detalles.

Observaciones

Para los sensores sin asignar sería razonable para establecer los ajustes de calibración superior e inferior por encima y por debajo del valor bruto forzada con fdSetSensorRaw () y fdSetSensorRawAll ().

---

void fdResetCalibration (fdGlove * PFG)

Restablece la configuración de auto-calibración interna del controlador a los valores por defecto correspondientes.

Valor de Retorno

Ninguno.

Parámetros

PFG

Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Observaciones

Esta función es similar a la función fdSetCalibrationAll () con cada uno de los valores de matriz de calibración superior e inferior establecidos a 0 y 4095 respectivamente. Esta función, o cualquiera de las otras funciones de calibración, se debe llamar cada vez que se inicia la aplicación o el guante cambia usuarios durante el tiempo de ejecución. Para los sensores sin asignar los valores de calibración superior e inferior se establecen en 4,095 y 0, respectivamente, que es el inverso de la configuración de auto-calibración.

---

void fdGetSensorMaxAll (fdGlove * pFG, float * Pmax)

Obtiene el valor máximo reducido para cada sensor.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Pmax
Matriz de valores de punto flotante que contendrán la máxima escala valores de los sensores. El tamaño de la matriz siempre debe coincidir con el valor devuelto por fdGetNumSensors ().

Observaciones
Los valores predeterminados del Data Glove a un valor máximo a escala de 1 para cada sensor.

---

void fdGetSensorMax (fdGlove * PFG, int nSensor)
Obtiene el valor máximo a escala para un sensor específico.

Valor de Retorno
Devuelve los valores máximos a escala del sensor.

Parametros
pFG
Pointer to a glove device. This is the value returned by fdOpen().

nSensor
Index of the sensor that is being queried. The value must lie in the range given by the enumerated type EfdSensors, or alternatively from zero to the value returned by fdGetNumSensors() minus one.
Remarks The glove driver defaults to a maximum scaled value of 1 for each sensor.

---

void fdSetSensorMaxAll(fdGlove *pFG, float *pMax)
Sets the maximum scaled value for each sensor.
Return value
Ninguno

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

nSensor
Índice del sensor que se está consultando. El valor debe estar en el intervalo dado por los EfdSensors tipo enumerado, o alternativamente desde cero hasta el valor devuelto por fdGetNumSensors () menos uno.

Observaciones
Los valores predeterminados del Data Glove a un valor máximo a escala de 1 para cada sensor.

---

void fdSetSensorMaxAll (fdGlove * pFG, float * Pmax)
Establece el valor máximo a escala para cada sensor.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

Pmax
Arsenal de los valores de los puntos que contiene los valores máximos a escala sensor flotante. El tamaño de la matriz siempre debe coincidir con el valor devuelto por fdGetNumSensors ().

Observaciones
Los valores predeterminados del Data Glove a un valor máximo a escala de 1 para cada sensor.

---

void fdSetSensorMax (fdGlove * pFG, int nSensor, float fMax)
Establece el valor máximo a escala para un sensor específico.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

nSensor
Índice del sensor que se está fijando. El valor debe estar en el intervalo dado por los EfdSensors tipo enumerado, o alternativamente desde cero hasta el valor devuelto por fdGetNumSensors () menos uno.

fMax
Un valor de coma flotante que contiene el valor máximo reducido sensor.

Observaciones
Los valores predeterminados del Data Glove a un valor máximo a escala de 1 para cada sensor.

---

void fdGetThresholdAll (fdGlove * pFG, float * pupper, float * plower)

Obtiene la configuración de umbral de reconocimiento de gestos actuales del controlador.

Valor de Retorno
Ninguno.

Parámetros
pFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

pupper y plower
Las matrices de números de punto que contendrán los valores mínimos de umbral máximo y flotante. El tamaño de cada matriz siempre debe coincidir con el valor devuelto por fdGetNumSensors (). Consulte la sección 6 para más detalles.

Observaciones
Ninguno.

---

void fdGetThreshold (fdGlove * PFG, int nSensor, float * pUpper, float * pLower)
Obtiene la configuración de umbral de reconocimiento de gestos actuales del controlador para un sensor específico.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

nSensor
Índice del sensor que se está consultando. El valor debe estar en el intervalo dado por los EfdSensors tipo enumerado, o alternativamente desde cero hasta el valor devuelto por fdGetNumSensors () menos uno.

pUpper y pLower
Punteros a números de punto que contendrán los valores mínimos de umbral máximo y flotante. Consulte la sección 6 para más detalles.

Observaciones
Ninguno.

---

void fdSetThresholdAll (fdGlove * pFG, float * pUpper, float * pLower)
Establece los valores de umbral de reconocimiento de gestos actuales del controlador.

Valor de Retorno
Ninguno.

Parámetros

pFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

pUpper y pLower
Las matrices de números de punto que contiene los valores máximos y mínimos de umbral flotante. El tamaño de cada matriz siempre debe coincidir con el valor devuelto por fdGetNumSensors (). Consulte la sección 6 para más detalles.

Observaciones
Ninguno.

---

void fdSetThreshold (fdGlove * pFG, int nSensor, float fUpper, float fLower)
Establece los valores de umbral de reconocimiento de gestos actuales del controlador para un sensor específico.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

nSensor
Índice del sensor que se está fijando. El valor debe estar en el intervalo dado por los EfdSensors tipo enumerado, o alternativamente desde cero hasta el valor devuelto por fdGetNumSensors () menos uno.

fUpper y fLower
Números de punto flotante que contienen los valores mínimo y máximo y umbral. Consulte la sección 6 para más detalles.

Observaciones
Ninguno.

---

void fdGetGloveInfo (fdGlove * pFG, unsigned char * pData)
Obtiene el bloque de datos de información del guante conectado actualmente.

Valor de Retorno
Ninguno.

Parámetros
pFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

pData
Arsenal de 32 bytes que contendrán los datos de información.

Observaciones

Los datos de información se especifican en el manual del usuario guante. El tamaño del bloque de información siempre es 32 bytes.

---

void fdGetDriverInfo (fdGlove * PFG , unsigned char * pData)
Obtiene el bloque de datos de información del driver.

Valor de Retorno
Ninguno.

Parámetros
PFG
Puntero a un dispositivo de guante. Éste es el valor devuelto por fdopen ().

pData
Array de 32 bytes que contendrán los datos de información.

Observaciones
Los datos de la información es una cadena terminada en cero que contiene información para el driver. El tamaño del bloque de información siempre es 32 bytes.

Definición de Gesto

El conjunto definido actualmente de gestos se compone de configuraciones binarias abiertas/cerradas de los dedos excluyendo el pulgar. Hay 24 = 16 tales combinaciones posibles. Gesto número 0 se define como todos los dedos (excluyendo el pulgar) está cerrado, y el gesto número 15 se define como todos los dedos abiertos. El dedo índice indica el bit menos significativo. Por ejemplo, el gesto del dedo índice será por lo tanto el número 1, y el gesto para señalar con el dedo será 8. Un irreconocible gesto no válido se devolverá como el valor -1.

Un valor del sensor a escala de más alto que el valor del umbral superior indicará un dedo cerrado, mientras que un valor del sensor a escala del menor que el valor del umbral inferior indicará un dedo abierto. Un valor en el medio no es válido y se traducirá en un gesto no válido. En el caso de mediciones del ángulo de articulación del dedo múltiples (tales como el guante 16 sensor), se toma el máximo de los valores de los sensores conjunta individuales para obtener un gesto cerrado y el mínimo para obtener un gesto abierto. Gestos cerrados tienen prioridad, es decir, doblando sólo una articulación de un dedo contará como un gesto cerrado.

Las capacidades integradas de reconocimiento de gestos del driver del guante están limitado en su alcance y la independencia del usuario. Los algoritmos de reconocimiento de gestos de alto nivel que se basan en secuencias de entrenamiento se sugieren para aplicaciones avanzadas.

are suggested for advanced applications.

Finger:	Little	Ring	Middle	Index
5DT Data Glove 5 sensor:	E	D	C	B
5DT Data Glove 16 sensor:	12,13	9,10	6,7	3,4
Driver sensor index:	12,13#	9,10#	6,7#	3,4#

Gesture Number	Flexure (0=flexed, 1=unflexed)				Gesture Description	Fig.
0	0	0	0	0	Fist	16.0
1	0	0	0	1	Index finger point	16.1
2	0	0	1	0	Up yours	16.2
3	0	0	1	1	Two finger point	16.3
4	0	1	0	0	Ring finger point	16.4
5	0	1	0	1	Ring index point	16.5
6	0	1	1	0	Ring middle point	16.6
7	0	1	1	1	Three finger point (or not little point)	16.7
8	1	0	0	0	Little finger point	16.8
9	1	0	0	1	Howzit	16.9
10	1	0	1	0	Little middle point	16.10
11	1	0	1	1	Not ring finger point	16.11
12	1	1	0	0	Little ring point	16.12
13	1	1	0	1	Not up yours	16.13
14	1	1	1	0	Not index finger point	16.14
15	1	1	1	1	Flat hand	16.15

Table 1. Gesture definition scheme as implemented for the 5DT Data Glove

Cuando se utiliza el 5DT Data Glove 6, los índices de sensores controlador devolverá valores diferentes. El máximo de los dos valores se utiliza para comprobar si un gesto flexionada (cerrado), y el valor mínimo de los dos será utilizada para la prueba de un (abierto) gesto no flexionada.

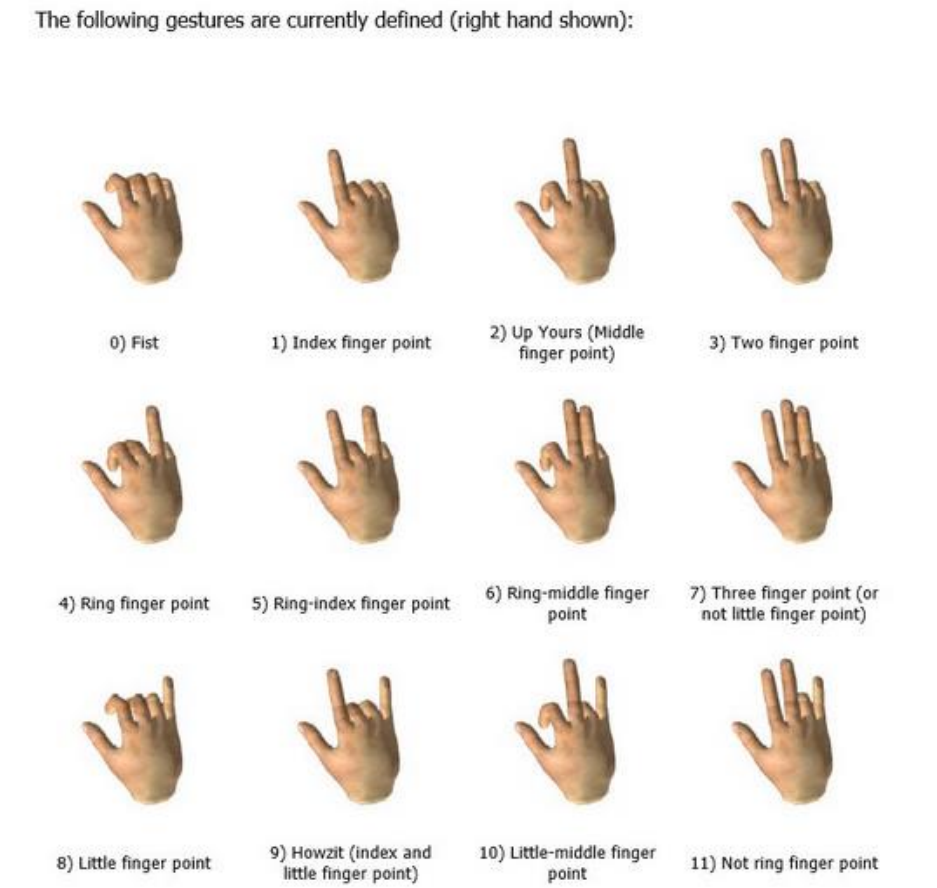


Figura 19. Gestos

Bibliografia

1. System Simulation Techniques with MATLAB and Simulink. Dingyü Xue, YangQuan Chen

2. Introduction to Simulink with Engineering Application. Steven T. Karris

3. BBC Mundo, “Microsoft crea sensor que replica los movimientos de la mano” 10 octubre 2012

4. BIENETEC, “CyberGlove Systems CyberGlove III”

5. Diego Hernando Rodríguez Gaitán (Octubre, 2005). Tutorial Virtual Reality toolbox. UNIVERSIDAD DE IBAGUE – CORUNIVERSITARIA.

6. Ceballos Sierra F. J., (2006). Enciclopedia del lenguaje C, Octava reimpresión, Alfa Omega, México.

7. Matlab Tutorial. Javier García de Jalón, José Ignacio Rodríguez, Jesús Vidal

8. M.G. Victor Hugo, P.B. Victor Hugo, “Manipulación de una mano virtual utilizando el guante P5”, Febrero 2006

9. Guía Beej de programación en redes - http://www.unlu.edu.ar/~tyr/tyr/TYR-trab/satobigal/documentacion/beej/clientserver.html

10. Realidad Virtual - http://www.jeuazarru.com/docs/RealidadVirtual.pdf

11. http://www.microsoft.com

12. Avances en la realidad virtual - http://www.puntogeek.com/2010/04/29/avances-en-la-realidad-virtual/

13. Matlab Tutorial. Javier García de Jalón, José Ignacio Rodríguez, Jesús Vidal

14. Carl Kenner " GlovePie 0.29" 4 Enero 2007

15. http://www.docstoc.com/docs/15520625/GlovePIE-GUI

16. Diego Hernando Rodríguez Gaitán (Octubre, 2005). Tutorial Virtual Reality toolbox. UNIVERSIDAD DE IBAGUE – CORUNIVERSITARIA.

17. Ceballos Sierra F. J., (2006). Enciclopedia del lenguaje C, Octava reimpresión, Alfa Omega, México.

18. MathWorks, (2014). Ejemplos, http://www.mathworks.com.

19 Data Glove. www.5dt.com