SISTEMA DE INTERFAZ HAPTICA PARA EL CONTROL DE POSICION EN UN ESPACIO TRIDIMENSIONAL VIRTUAL

TITO MANUEL PIAMBA MAMIAN

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Línea de I+D en Robótica y Control
Ingeniería en Automática Industrial

Popayán, Febrero 2011

SISTEMA DE INTERFAZ HAPTICA PARA EL CONTROL DE POSICION EN UN ESPACIO TRIDIMENSIONAL VIRTUAL

TITO MANUEL PIAMBA MAMIAN

Tesis presentada a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca para la obtención del Título de

Ingeniero en Automática Industrial

Director:
Ing. Vladimir Trujillo Arias

Popayán, Febrero 2011

Hoja de Aprobación

_____
_____
_____

Director _____
Ing. Vladimir Trujillo Arias

Jurado _____

Jurado _____

Fecha de sustentación: Popayán,

A mi familia que siempre lucha
y que me hace estar orgulloso
cada día de ellos.
A mis compañeros y
tutores que me guiaron en
este camino.

Agradecimientos

A Dios que me permite vivir y disfrutar este camino y culminación de mis estudios de pregrado.

A mi familia, que nunca deja de apoyarme en mis decisiones.

A él “angelito” que cuida de mí desde el cielo

A mi ahijada Valentina, que por su foto a mi lado, me apoyo en todo momento.

A Sergio Salinas y Andrés Vivas por su gran colaboración en el inicio y culminación del proyecto.

A mis compañeros de estudio que ayudaron con sus aportes y apoyo en la culminación del proyecto.

A mis amigos, Jhon, Felipe, Yamir, Marien, Kelly y Ermilso, que me ayudaron a seguir en este proyecto.

A los evaluadores de este proyecto.

A Vladimir Trujillo por su labor de Director de Trabajo de Grado.

A la universidad del Cauca por sus enseñanzas dadas en el trascurso de mi carrera.

Resumen

HAPTIC 3R (Interfaz háptica), es el robot diseñado en este proyecto. Su estructura se basa en interfaces hápticas existentes. Ejemplo de estas, están la PHANTOM PREMIUM 3.0, FREEDOM 6S Y FALCON que se encuentran en comercialización para su adquisición.

HAPTIC 3R es modelado matemáticamente de forma geométrica y dinámica para comprobar el seguimiento de trayectorias deseadas y obtener un Modelo Geométrico Inverso que tendrá la interfaz en su control (hardware) o en su interfaz 3D (software).

Para HAPTIC 3R se desarrolla una interfaz de usuario y una interfaz 3D para recrear un ambiente tridimensional, usando para la primera, la plataforma de programación de Microsoft® Visual Studio C# y para la interfaz 3D se agrega a la anterior plataforma la librería grafica gratuita OPENGL. Para la comunicación USB se utilizara la librería gratuita de MICROCHIP® mpusbapi.dll. EL diseño de las estructura, piezas y cálculos de parámetros matemáticos se realiza en el software CAD SolidEdge®.

La simulación tridimensional o interfaz 3D, permite verificar la retroalimentación de fuerza realizada por la interfaz HAPTIC 3R en el momento que un evento de colisión ocurra y la interfaz de usuario informara los datos de las consignas entregadas por el Modelo Geométrico Inverso y la manipulación de posición de cada microservomotor.

Abstract

HAPTIC 3R (haptic interface), is the robot designed in this project. Its structure is based on existing haptic interfaces. Examples of this are the PHANTOM Premium 3.0, FREDOM 6S y FALCON found in marketing for purchase.

HAPTIC 3R is mathematically modeled geometry and dynamic test track desired trajectories and obtain a Geometric Model Inverse which will control (hardware) or in this 3D interface (software).

For HAPTIC 3R develop a user interface and 3D interface to recreate a three dimensional environment, using the first, the software platform Microsoft® Visual Studio C# and the 3D interface is added to the previous platform graphics library free OPENGL. USB communication to be used free library free MICROCHIP® mpusbapi.dll. The design of the structure, parts and mathematical calculations of parameters is done in SolidEdge® CAD software.

The simulated three-dimensional or 3D interface to verifying the force-feedback by HAPTIC 3R at the time that a collision event occurs and the user interface data report released by the slogans Inverse Geometric Model and manipulation of position each microservomotor.

TABLA DE CONTENIDO

Lista de Tablas

Lista de Figuras

Lista de Abreviaturas

CMI:	Cirugía Mini-Invasiva
3D:	Tridimensional.
CTC:	(*Computed Torque Control*) Control por Par Calculado.
MDD:	Modelo Dinámico Directo.
MDI:	Modelo Dinámico Inverso.
MGD:	Modelo Geométrico Directo.
MGI:	Modelo Geométrico Inverso.
SYMORO®:	(Symbolic Modeling of Robots)
HAPTIC 3R:	Interfaz Háptica.
Microservo:	Microservomotor.
PIC:	Microcontrolador (MICROHIP®)
OpenGL:	(Open Graphics Library) Librería Gráfica de Libre Uso.
Ms:	Milisegundos.
V:	Voltaje (voltios).

Lista De Símbolos

A	Matriz de inercia.
$\hat{A}$	Matriz de inercia aproximada.
*iA*j	Matriz de orientación.
C	Vector de fuerzas de Coriolis y centrífugas.
Ci	cos(*θj*)
Cij	cos(*θi* + *θj*)
Si	*sin*(*θj*)
Sij	sin(*θi* + *θj*)
D, R	Distancias fijas [m].
H	Matriz de fuerzas de Coriolis, centrífugas y la gravedad.
$\hat{H}$	Matriz aproximada de Coriolis, centrífugas y la gravedad.
Ia	Inercia de motor [kg*m2].
j	Número de la articulación.
Kp	Vector de ganancias proporcionales.
Kv	Vector de ganancias derivativas.
Mj	Masa de la articulación j [kg].
MS	*[MX MY MZ]* Primer momento de inercia [kg*m].
Pi	Vector de posición.
q	Vector de variables articulares.

INTRODUCCIÓN

El humano es un ser perceptivo, recibe estímulos de nuestro entorno a través de sus sentidos, pero se sabe que sus dos sentidos principales son la audición y la visión, ya que son los canales de captura de estímulos más efectivos. Luego está el gusto y el olfato y por ultimo esta la percepción por medio del tacto que es el principio fundamental de la háptica.

El termino háptica no aparece en el diccionario de la lengua española, sin embargo se puede identificar desde su origen griego “Hapthys” que se refiere al tacto o percepción del tacto, pero algunos teóricos como Herbert Read [1] han ampliado este concepto de manera que esta hace alusión por exclusión a todo el conjunto de sensaciones no visuales y no auditivas que experimenta un individuo y según Juan Manuel Gutiérrez [2], Hàptica significa “tacto activo, por lo tanto es la entrada de datos de forma activa y consciente”.

Gibson [3] define el sistema háptico como "la percepción del individuo del mundo adyacente a su cuerpo mediante el uso de su propio cuerpo". El sistema de percepción háptica es especial porque puede incluir los receptores sensoriales ubicados en todo el cuerpo y está estrechamente relacionado con el movimiento del cuerpo, de forma que puede tener un efecto directo sobre el mundo que está percibiendo.

Si nos referimos al término interfaz háptica, se alude a aquellos dispositivos que permiten al usuario tocar, sentir o manipular objetos que son recreados bajo simulaciones en entornos 3D o entornos virtuales.

En la actualidad, estos dispositivos son utilizados en diferentes aplicaciones, tales como videojuegos [6], industria [7], medicina [8] e inclusive en lo que se conoce como robótica pedagógica [9], sin descartar posibles aplicaciones futuras. Vale reconocer que la aplicación más importante conocida hasta hoy de estos dispositivos está dirigida al campo de la medicina, donde se encuentran tanto en el campo de la rehabilitación [10], la telecirugía y en entrenadores quirúrgicos [11].

Este proyecto pretende proponer un prototipo de interfaz háptica presentando una estructura basada en los análisis de las estructuras o prototipos existentes, construir esta propuesta cinemática (o estructura), modelar su geometría y su dinámica y simular su comportamiento frente a una trayectoria deseada. También se diseñara una interfaz de usuario (hombre-computador) que se compondrá de

parte de información (datos ADC, valores del MGI, comportamiento de cada microservo) y una interfaz 3D para comprobar el funcionamiento de la interfaz háptica construida, la cual de aquí en adelante se llamara HAPTIC 3R por tener 3 grados de libertad para posicionamiento.

El trabajo está compuesto por cinco capítulos: el primer capítulo incluye conceptos básicos sobre interfaces hápticas, clasificación de éstas y entornos tridimensionales virtuales, el segundo capítulo se encuentra el modelado matemático y estructura del HAPTIC 3R, En el tercer capítulo se realiza la construcción de HAPTIC 3R, selección de materiales, actuadores y el diseño de la tarjeta de control. El capítulo cuatro muestra el desarrollo del software para establecer la comunicación USB, Interfaz de usuario e Interfaz 3D, y por último el capítulo cinco donde quedan consignadas las conclusiones del proyecto.

1. INTERFACES HAPTICAS

Las interfaces hápticas permiten al operador mantener la sensación de interacción con el entorno remoto. En ocasiones, y gracias a la diferencia en la frecuencia de las señales kinestesicas que son las percepciones del equilibrio y de la posición de las partes del cuerpo, visual y sonora, la realimentación háptica se combina fácilmente con realimentación óptica o acústica. Para obtener un buen comportamiento del sistema completo, se deben considerar tres factores que afectan principalmente a la sensación de interacción:

- La naturaleza de la señal realimentada, que puede provenir de un entorno físico real, un entorno virtual generado por computador o una combinación de ambos.

- El control del propio dispositivo háptico.

- Las características mecánicas del propio dispositivo.

Los sistemas de teleoperación con realimentación son esquemas bilaterales que incluyen una zona remota, una zona local y una comunicación entre ambas, de forma que el operador, colocado en la zona remota, pueda interactuar con los objetos de la zona local.

De las definiciones anteriores se desprende que la única diferencia entre un sistema háptico y los sistemas teleoperados reales afecta a la generación de la señal de realimentación. En el primer caso, la zona remota no existe físicamente, y es simulado vía software, mientras que en el segundo caso, la zona remota está compuesta por un robot manipulador, su controladora y el propio entorno con sus leyes físicas.

1.1 CLASIFICACION DE LAS INTERFACES HAPTICAS

Las interfaces hápticas se pueden clasificar en [12] :

- **Pasivas:** generalmente diseñadas para ofrecer resistencia al movimiento del usuario

- **Activas:** el intercambio de energía entre el usuario y la maquina se encuentra en función de la retroalimentación que es aplicada, o sea puede ofrecer respuesta al movimiento del usuario.

1.2 APLICACIONES DE LAS INTERFACES HAPTICAS

Las principales aplicaciones de las interfaces hápticas son:

- **Medicina:** simuladores quirúrgicos para el entrenamiento médico y micro-robots para cirugía mínimamente invasiva [13] .

- **Educacional**: para entrenamiento de técnicos, donde proporciona a los estudiantes la posibilidad de experimentar fenómenos a escalas micrométricas y macrométricas (escalas astronómicas) [14] .

- **Entretenimiento**: juegos de video y simuladores que permiten al usuario sentir y manipular objetos virtuales [15] .

- **Industria:** integración de interfaces hápticas en los sistemas CAD de tal forma que el usuario pueda manipular libremente los componentes de un conjunto en un entorno inmersivo [7].

- **Artes plásticas:** para tener la posibilidad de sentir físicamente exhibiciones virtuales de arte y esculturas en museos [16] .

1.3 INTERFACES HAPTICAS EXISTENTES

Entre los dispositivos comerciales y prototipos de retroalimentación de esfuerzo que se destacan se encuentran:

- **Interfaz Phantom:** es comercializada por la empresa *Sensable Tecnologies*, tiene entre 3 y 6 grados de libertad para posicionamiento. A la izquierda de la Figura1.1 se muestra una versión con realimentación de tres grados de libertad. De igual forma la fuerza de los distintos modelos varia considerablemente, por ejemplo, la fuerza máxima que puede proporcionar

el modelo Premium 3.0 de la derecha de la misma figura, es de 22 Newton y la fuerza sostenida es de 3 Newton [17].

Figura1.1. Phantom Sensable (izquierda), Phantom Premium 3.0 (derecha) [17].

- **Interfaz Falcon 3DOF:** ofrecido por la empresa Novint como un Joystick para video juegos que consiste en un manipulador de 3 grados de libertad, que ha sido aplicado como una interfaz háptica en proyectos de medicina [18] (Ver Figura 1.2).

Figura 1.2. Falcon #DOF[18].

- **Impulse Engine:** modelo actualmente comercializado por Inmersión CO. que se puede observar en la Figura 1.3, posee 2 grados de libertad, su valor de fuerza máxima es de aproximadamente 9 Newton y su ancho de banda es de 650Hz [19].

Figura 1.3. Impulse Engine Inmersion [19].

- **Haptic Master:** consiste en un brazo robótico (Figura 1.4) diseñado por FCS Control System, que se usa para medir fuerzas dinámicas y tele asistencia quirúrgica [20].

Figura 1.4. Haptic Master [20].

- **Freedom 6S:** interfaz de la Figura 1.5, construida por la empresa MPB Technologies, posee 6 grados de libertad y una fricción de aproximadamente 0.1Newton en cada dirección. La inercia resultante en el extremo varía entre 0.09 y 0.15Kg [21].

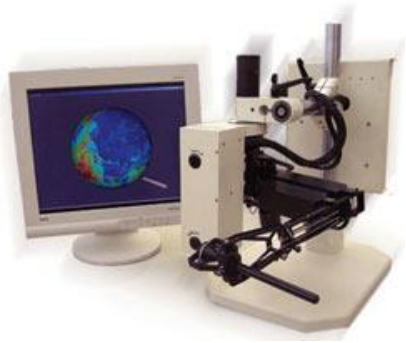

Figura 1.5. Freedom 6S [21].

- **Guantes con Feedback de Fuerza:** el único guante disponible comercialmente es el Cybergroup de la Figura 1.6, fabricado por Inmersion Co. El Cybergroup consiste de una estructura exoesquelética fijada a la parte posterior de la mano, que es accionada por actuadores instalados fuera de ésta, en una caja de control, con el objetivo de facilitar su manejo aligerando su peso que es de aproximadamente 450gr. La fuerza máxima que puede aplicar sobre cada dedo es de 12 Newton [22].

Figura 1.6. The Cybergroup [22].

- **CyberTouch Inmersion Co:** son guantes más ligeros que los que poseen feedback, estos emplean normalmente vibradores electromecánicos para proporcionar datos de texturas o rugosidades. El Cybertouch pesa 144 gramos. Usa 6 vibradores electromecánicos y actúan aproximadamente 1.2 Newton de fuerza (ver Figura 1.7) [23].

Figura 1.7. The CyberTouch [23].

- **Virtuose 6D Desktop:** el Virtuose 6D [24] posee tres cadenas articuladas, esta interfaz tiene una cinemática de 6 grados de libertad. Su fuerza máxima es de 7 a 10 Newton (ver Figura 1.8).

Figura 1.8. Virtuose Deskopt 6D [24].

- **Virtuose 6D35-45:** en la Figura 1.9 se muestra el Virtuose 6D35-45, este ofrece una fuerza de retroalimentación en sus 6 grados de libertad. Su diseño es modular, o sea puede ser adquirido como un dispositivo de 3 grados de libertad y luego ampliarlo a 6 grados de libertad. Su fuerza máxima es de 35 Newton [25].

Figura 1.9. Virtuose 6D35-45 [25].

- **Virtuose 6D40-40:** es un maestro para la teleoperación con retroalimentación de fuerza. Su arquitectura simple tipo serie se acondiciona a espacios de trabajos en ámbitos de cirugías [26]. Su fuerza máxima es de 100 Newton (ver Figura 1.10).

Figura 1.10. Virtuose 6D40-40 [26].

1.4 DESARROLLO DE APLICACIONES PARA LAS INTERFACEZ HAPTICAS

A continuación se muestran algunas de las aplicaciones o proyectos de las interfaces hápticas en entornos tridimensionales.

- **Proyecto Enactive**. Expertos europeos en robótica, realidad virtual, psicología experimental y neurociencia desarrollan las llamadas Interfaces Enactive, basadas en el uso de las manos para el aprendizaje a través de la acción, introduciendo un nuevo paradigma de la educación asistida por las tecnologías de la información [27]. Usando estas interfaces, con una pantalla, un guante y alguna herramienta, se puede aprender carpintería de la misma forma que si el aprendiz estuviera en un taller real. Las Interfaces Enactive también podrían utilizarse para rehabilitaciones, prácticas de cirugía o exploración espacial [27].

Este proyecto ha permitido que en los últimos años se desarrolle una próspera comunidad de investigación que trabaja en el conocimiento enactivo asistido por ordenadores. La interacción convencional con la información que proporcionan los ordenadores se basa mayormente en el conocimiento simbólico (palabras, símbolos matemático, etc.) o icónico (imágenes visuales como diagramas e ilustraciones), pero la interacción con los PC que promueve Enactive estaría basada en el uso activo de las manos. El coordinador del proyecto Massimo Bergamaso [28], profesor de Mecánica Aplicada en la Facultad de Ciencias Experimentales de la Scuola Superiore Sant'Anna, de Pisa (Italia), afirma que hace unos años había grupos de investigación de distintas disciplinas trabajando por separado en interfaces o enacción, pero que sólo gracias a Enactive sus esfuerzos se han reunido [28].

- **Proyecto Haptex** (Percepción Virtual Háptica de Productos Textiles). Este proyecto está enmarcado dentro del programa de tecnologías futuras y emergentes (FET) de la unión Europea (FGT/IST), inició en 2004 y finalizó en 2007. Fue desarrollado en los laboratorios de Miralab perteneciente a la Universidad de Ginebra y se basa en la percepción multimodal de los textiles en un entorno virtual. Haptex permiten a una persona percibir y manipular los textiles por medio de un robot (brazo) de seis grados de libertad [29].

- **Planeación Neuroquirúrgica y Navegación Estereotáxica**. Este proyecto presenta la fundamentación técnica, médica y matemática de los aspectos sobresalientes para el desarrollo de una interfaz visuo-háptica para la planeación y navegación estereotáxica de cirugías y el entrenamiento de neurocirujanos [30].

- **Texturas y detección de colisiones**. Son aplicaciones en las cuales son desarrolladas en OPENGL. Estas consisten en la detección de eventos de colisión y de identificación de texturas. En la Figura 1.11 se tiene un ejemplo de detección de colisiones aplicado a la interfaz háptica FALCON 3DOF [31].

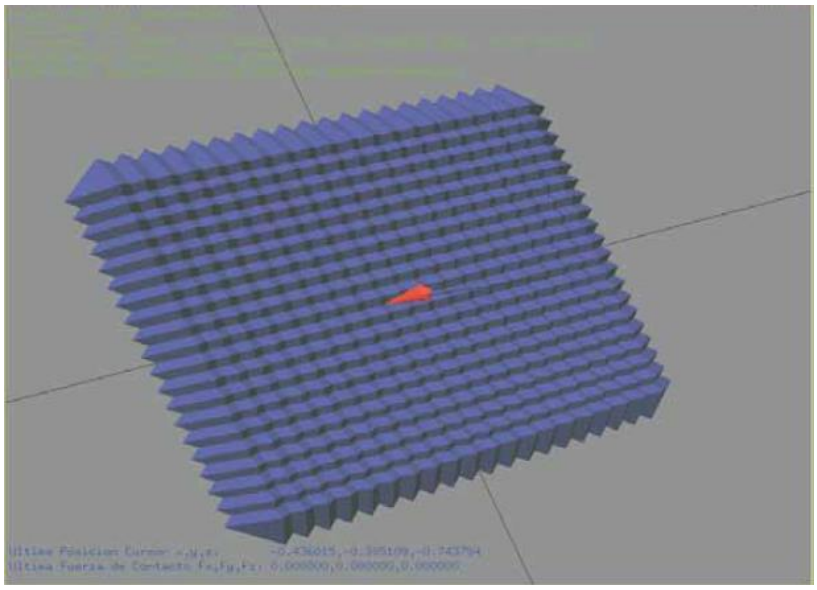

Figura 1.11. Detección de texturas en OPENGL [31].

2. MODELADO DEL HAPTIC 3R

Este capítulo muestra la obtención del modelo matemático del HAPTIC 3R para la comprobación del comportamiento de esta mediante simulación, y la obtención del MGI para entregar la posición geométrica dada por la interfaz a un objeto simulado en un mundo virtual tridimensional.

2.1 ESTRUCTURA FUNCIONAL DEL HAPTIC 3R

A continuación se definen los requisitos mínimos que debe cumplir la estructura del HAPTIC 3R a desarrollar.

- Tres grados de libertad activos que proporcionen la posibilidad de posicionar algún objeto tridimensional en un espacio de trabajo, en este caso el movimiento de una esfera en un espacio de trabajo definido como un cubo.

- Un grado de libertad activo que proporciones rotación al objeto tridimensional seleccionado.

- La retroalimentación de fuerza en el momento de detección de colisiones, esta debe ser capaz de detener o ofrecer un esfuerzo mecánico en el momento de un evento de colisión.

  En la Figura 2.1 se observa la estructura cinemática seleccionada para el HAPTIC 3R, las distancias están etiquetadas con letras D y R (valores constantes), la articulación de rotación se representa con cilindros, cada una con su respectivo número (en la Tabla 2.1, se muestra los valores de sus parámetros geométricos).

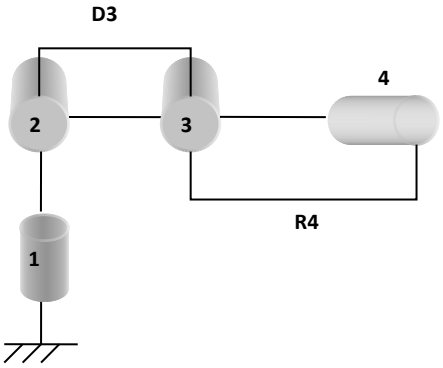

Figura 2.1. Estructura cinemática del HAPTC 3R.

2.2 MODELADO MATEMATICO DEL HAPTIC 3R

Luego de seleccionar la estructura de HAPTIC 3R, se procede a calcular los modelos matemáticos para verificar que este realice una trayectoria deseada. Estos modelos permiten representar geométricamente la interfaz. Utilizando el método de Khalil-Dombre [32] y como resultado del análisis de la estructura cinemática del HAPTIC 3R se obtiene la siguiente tabla de parámetros (Tabla 2.1).

Tabla 2.1. Parámetros geométricos del HAPTIC 3R

j	σj	A	θj	dj	Rj
1	0	0	Θ1	0	0
2	0	90	Θ2	0	0
3	0	0	Θ3	D3	0
4	0	90	Θ4	0	R4

Definición de los parámetros geométricos:

- **j:** Numero de la articulación
- **σ:** Indica si la articulación es rotoide (0) o prismática (1).

- **θ:** Angulo entre los ejes X_{j-1} y X_j correspondiente a una rotación alrededor de Z_j.

- **d:** distancia entre Z_{j-1} y Z_j a lo largo de X_{j-1}.

- **r**: distancia entre X_{j-1} y X_j a lo largo de Z_j

A partir de la tabla de parámetros geométricos de HAPTIC 3R se hallan los modelos geométricos directo (MGD) y geométrico inverso (MGI).

2.2.1 Modelo Geométrico Directo (MGD)

El MGD da a conocer la posición y orientación del efector final en el espacio cartesiano (***x, y, z***) a partir de las posiciones articulares (***θ y r***) de cada articulación del robot, haciendo uso de las matrices de transformación. El MGD del HAPTIC 3R fue obtenido a partir de la Tabla 2.1, de la cual se obtiene las matrices de transformación, en la ecuación (2.1) muestra la matriz de transformación donde A es una matriz 3x3 que representa la orientación y P(x, y, z) es el vector columna que representa la posición del efector final.

$$ {}^{i}T_{j} = \begin{bmatrix} & {}^{i}A_{j} & & {}^{i}P_{j} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.1) $$

El MGD del HAPTIC 3R está determinado por la matriz de transformación ${}^{0}T_{4}$, de la cual se obtuvieron las siguientes ecuaciones donde involucran la posición del efector final con las variables articulares:

$$ X = D3C1C2 + R4C1S23 \qquad (2.2) $$

$$ Y = D3C2S1 + R4S1S23 \qquad (2.3) $$

$$ Z = -R4C23 + D3S2 \qquad (2.4) $$

Donde:

$$ Ci = \cos(\theta j) $$

$$ Si = \operatorname{sen}(\theta j) $$

$$ Cij = \cos(\theta i + \theta j) $$

$$Sij = sen(\theta i + \theta j)$$

Ahora, al comparar la matriz de orientación $.^{0}A_{4}.$ con la forma general (ecuación (2.5))

$$^{i}A_{j} = \begin{bmatrix} S_x & n_x & a_x \\ S_y & n_y & a_y \\ S_z & n_z & a_z \end{bmatrix} \qquad (2.5)$$

Se encuentran las ecuaciones que representan la orientación del efector final.

$$s_x = C1C23C4 + S1S4 \qquad (2.6)$$

$$s_y = C23C4s1 - C1S4 \qquad (2.7)$$

$$s_z = C4S23 \qquad (2.8)$$

$$n_x = C4S1 - C1C23S4 \qquad (2.9)$$

$$n_y = -C1C4 - C23S1S4 \qquad (2.10)$$

$$n_z = -S23S4 \qquad (2.11)$$

$$a_x = C1S23 \qquad (2.12)$$

$$a_Y = S1S23 \qquad (2.13)$$

$$a_z = C23 \qquad (2.14)$$

Ahora las matrices de transformación:

$$^{0}T_{1} = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ {}^{1}\mathrm{T}_{2} = \begin{bmatrix} \mathrm{C2} & -\mathrm{S2} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \mathrm{S2} & \mathrm{C2} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}\mathrm{T}_{3} = \begin{bmatrix} \mathrm{C3} & -\mathrm{S3} & 1 & \mathrm{D4} \\ \mathrm{S3} & \mathrm{C3} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}\mathrm{T}_{4} = \begin{bmatrix} \mathrm{C4} & -\mathrm{S4} & 0 & 0 \\ 0 & 0 & -1 & -\mathrm{R4} \\ \mathrm{S4} & \mathrm{C4} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

2.2.2 Modelo Geométrico Inverso (MGI)

El MGI permite encontrar el valor de las variables (**θ**) y (**r**) por cada articulación conociendo la posición y orientación en el sistema cartesiano ($\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$) del efector final. Para hallar el MGI se utiliza el método de Paul [33] el cual define una ecuación de igualdad entre el MGD y la matriz U_0 deseada (ecuación (2.15)).

$$ \mathrm{U}_{0} = \begin{bmatrix} \mathrm{S_x} & \mathrm{n_x} & \mathrm{a_x} & \mathrm{P_x} \\ \mathrm{S_y} & \mathrm{n_y} & \mathrm{a_y} & \mathrm{P_y} \\ \mathrm{S_Z} & \mathrm{n_Z} & \mathrm{a_Z} & \mathrm{P_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.15) $$

Ahora se busca resolver el sistema:

$$ \mathrm{U}_{0} = {}^{0}\mathrm{T}_{1}\ {}^{1}\mathrm{T}_{2}\ \ldots\ {}^{\mathrm{n}-1}\mathrm{T}_{\mathrm{n}} $$

En este caso:

$$ \mathrm{U}_{0} = {}^{0}\mathrm{T}_{1}\ {}^{1}\mathrm{T}_{2}\ {}^{2}\mathrm{T}_{3}\ {}^{3}\mathrm{T}_{4} $$

Se multiplica a la izquierda por ${}^{1}T_{0}$:

$$ {}^{1}T_{0}\ U_{0} = {}^{1}T_{2}\ {}^{2}T_{3}\ {}^{3}T_{4} \quad = \mathrm{U}_{1} $$

Por tanto, el término de la izquierda estará en función de los elementos de U_0 y de la variable *θ1.* La sucesión del cálculo para las demás variables cartesianas (*θ2, θ3* y *θ4)* será:

$$U_1 = {}^1T_0 \; U_0$$

$$U_2 = {}^1T_2 \; U_1 = {}^2T_3 \; {}^3T_4$$

$$U_3 = {}^2T_3 \; U_2 = {}^3T_4$$

Partiendo de:

$${}^0T_4 = U_0$$

Se obtiene la ecuación

$${}^0T_4 = \begin{bmatrix} S_x & n_x & a_x & X \\ S_y & n_y & a_y & Y \\ S_Z & n_Z & a_Z & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.16)$$

La matriz ${}^0T_4$ está compuesta por las ecuaciones (2.6) hasta la ecuación (2.14), resolviendo esta igualdad (ecuación (2.16)) se obtiene los valores de *θ1, θ2, θ3, θ4.*

Se tiene:

$$\theta_1 = \text{atan2}(y, x)$$

$$\theta_2 = \text{atan2}(A, B)$$

$$\theta_3 = \text{atan2}((D3S2S1 - S1z), (y - D3C2S1))$$

$\theta_4 = \text{atan2}(n_Z/S_Z)$ (Obtenido de la matriz de orientación)

Donde:

$$A = X1Z1 + Y1(X1^2 + Y1^2 - Z1^2)/(X1^2 + Y1^2)$$

$$B = X1Z1 - Y1(X1^2 + Y1^2 - Z1^2)/(X1^2 + Y1^2)$$

$$Z1 = (R4C1)2 - (x)2 - (C1z)2 - (D3C1)2$$

$$y1 = -2xD3C1$$

$$X1 = -2z\,(C1)^2 2D3$$

Las distancias D3 y R4 se definieron con valores de 0.11 m y .010 m respectivamente, la selección de material se especifica en el capítulo 3.

2.2.3 Modelos Dinámicos

Los modelos dinámicos representan la relación entre las fuerzas ejercidas por los motores, posición, velocidad y aceleración de las articulaciones del robot. Para el cálculo de estos modelos es necesario definir parámetros inerciales de base que son el conjunto mínimo de valores de inercias y masas del modelo dinámico del robot, calcular las fuerzas que requiere los motores para mover los cuerpos del robot, identificar los parámetros inerciales y el control del robot [32].

Al calcular los parámetros de base, estos permiten la simplificación de ecuaciones para eliminar los parámetros que no tienen efecto sobre el modelo. Los modelos a calcular son el Modelo Dinámico Inverso (MDI) que representa el esfuerzo del robot y el Modelo Dinámico Directo (MDD), este representa la relación entre las aceleraciones articulares y posición, velocidad y fuerza de las articulaciones. El cálculo de estos valores se realizó mediante la herramienta CAD SolidEdge®. Obtenidos estos valores de inercia y masa se procede a calcular los modelos MDI y MDD mediante el software de modelado simbólico de robots SYMORO® [34]. Este software proporciona los parámetros mínimos del robot HAPTIC 3R, estas se muestran en la Tabla 2.2. Las ecuaciones de reagrupamiento se pueden observar en el Anexo A.

Tabla 2.2. Parámetros de tensor de inercia del HAPTIC 3R.

j	XX	XY	XZ	YY	YZ	ZZ	MX	MY	MZ	M	Ia
1	0	0	0	0	0	ZZIR	0	0	0	0	0
2	XX2R	XY2	XZ2R	0	YZ2	ZZ2R	MX2R	MY2	0	0	IA2
3	XX3R	XY3	XZ3	0	YZ3R	ZZ3R	MX3	MY3R	0	0	IA3
4	XX4R	XY4	XZ4	0	YZ4	ZZ4	MX4	MY4	0	0	IA4

Donde:

- $\boldsymbol{M_j}$: Masa del cuerpo de la articulación *j*.
- $\boldsymbol{MX_j, MY_j, MZ_j}$: Vector columna del primer momento de inercia de la articulación *j*.
- $\boldsymbol{XX_j, XY_j, XZ_j, YY_j, YZ_j, ZZ_j}$: Elementos que representan el tensor de inercia de la articulación *j*.
- ***Iaj****:* Inercia del motor de la articulación *j.*

La letra R al final de algunos parámetros significa que dicho parámetro se encuentra agrupado con algunos que aparecen como nulos.
Aplicando nuevamente el software SYMORO® obtenemos el MDI, el cual calcula los pares mecánicos a aplicar en los motores en términos de la posición, velocidad y aceleración de cada articulación (ecuación (2.17))

$$\sigma = \mathrm{A(q)\ddot{q} + C(q,\dot{q})\dot{q} + Q(q)} \qquad (2.17)$$

Donde:

σ**:** Pares aplicados a los motores.
A**:** Matriz de inercias.
$C(q,\dot{q})$**:** Matriz de fuerzas Coriolis y centrifugas.
$Q(q)$**:** Vector de fuerzas centrifugas.
q**:** Posiciones articulares.

$\dot{q}$**:** Velocidades articulares.
$\ddot{q}$**:** Aceleraciones articulares.

De la ecuación (2.17) se despeja las aceleraciones articulares resultando la ecuación (2.18):

$$\ddot{\mathrm{q}} = \mathrm{inv}\big(\mathrm{A}(\mathrm{q})\big) * \big(\sigma - \mathrm{C}(\mathrm{q}, \dot{\mathrm{q}})\dot{\mathrm{q}} - \mathrm{Q}(\mathrm{q})\big) \qquad (2.18)$$

A continuación se muestra los valores dinámicos físicos obtenidos por el software SolidEdge® (ver Tabla 2.3) bajo su herramienta *propiedades físicas* (link 3.1.1).

Tabla 2.3. Parámetros inerciales reagrupados de base del HAPTIC 3R.

Parámetro	Valor	Parámetro	Valor
XX2R	-0.0030	ZZ2R	0.06015
XX3R	0.4407	ZZ3R	23.6292
XX4R	0.8219	MX2R	-0.4962
XZ2R	-1.2837	MY3R	-1.2837
YZ3R	0		
ZZ1R	6.0807		

2.3 CONTROL Y SIMULACION DEL HAPTIC 3R

Después de obtener los modelos matemáticos del HAPTIC 3R, el siguiente paso es realizar una simulación la cual esta debe entregar unas trayectorias deseadas y se compararan con las trayectorias entregadas por la simulación y obtener un error menor a 10% por cada parámetro.

2.3.1 Diseño de un Control por Par Calculado (CTC)

Esta técnica de control, por teoría asegura la liberalización mediante realimentación y el desacople de las ecuaciones del modelo dinámico del sistema,

entregando un comportamiento uniforme en cualquier configuración del robot en un determinado momento [34] - [36].

La ecuación (2.18) puede ser re-escrita como en la ecuación (2.19), donde la matriz ***H*** incluirá los términos de Coriolis, fuerzas centrífugas y términos gravitacionales.

$$\sigma = \mathrm{A(q)\ddot{q} + H(q, \dot{q})} \qquad (2.19)$$

Asumiendo que las matrices $A\ y\ H$ se pueden estimar por medio de $\widehat{A}$ y $\widehat{H}$, se escoge una señal de entrada al robot σ como aparece en la ecuación (2.20), que será el vector de pares a aplicar en los motores del robot.

$$\sigma = \mathrm{\widehat{A}\,(q)W(t) + \widehat{H}(q, \dot{q})} \qquad (2.20)$$

Donde $W(t)$ representa la nueva señal de control.

Como $\widehat{A}$ y $\widehat{H}$ son estimaciones de las dinámicas reales del manipulador ($A\ y\ H$, respectivamente), en el caso ideal y sin presencia de perturbaciones, el problema se reduce entonces a un caso de control lineal, donde $\widehat{A} = A$ y $\widehat{H} = H$, de tal forma que al conectarlo con el robot se logra hacer un desacoplamiento de las dependencias de las articulaciones y una compensación de las altas no linealidades del robot. Se puede así aplicar un controlador lineal para obtener la señal de control $W(t)$, usando las posiciones articulares deseadas q^d y las velocidades articulares deseadas $\dot{q}^{\boldsymbol{d}}$, como aparece en la ecuación (2.21) y en la Figura 2.2.

$$\mathrm{W(t) = Kp\left(q^{d} - q\right) + \ Kv\left(\dot{q}^{\mathbf{d}} - \dot{q}\right)} \qquad (2.21)$$

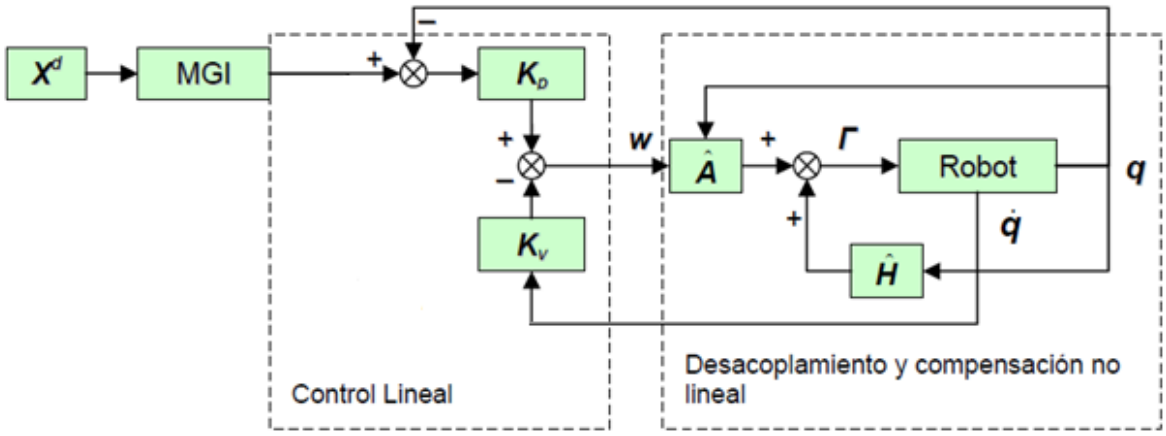

Figura 2.2. Esquema de control CTC.

Donde $Kp\ y\ Kv$ son las ganancias proporcionales y derivativas del controlador lineal respectivamente. En la Figura.2.3, el bloque del modelo geométrico inverso (MGI) se encarga de convertir la consigna cartesiana deseada en consigna articular.

El esquema del control CTC consta de dos bucles de control, el primero de ellos (lazo interior) se basa en el modelo dinámico del manipulador, cuya función es obtener la deseada relación entrada/salida lineal y desacoplada. El lazo exterior, por su parte, es el encargado de procesar el error, se encarga de estabilizar todo el sistema llevándolo a alcanzar el comportamiento deseado.

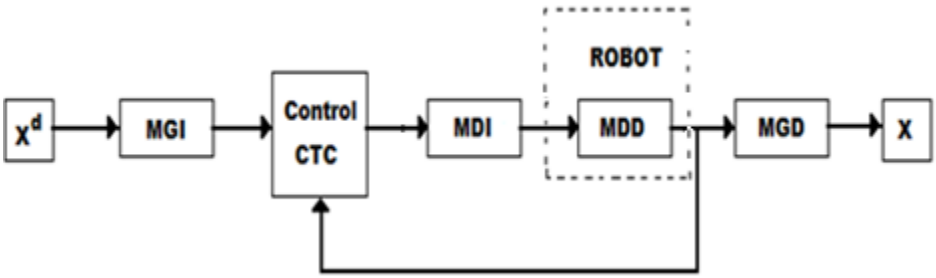

Figura.2.3. Esquema del controlador CTC.

En el esquema de control se hace de suma importancia la utilización del modelo geométrico directo (MGD), ya que éste permite obtener el movimiento del robot en

el espacio cartesiano $\boldsymbol{X}$ y verificar que está siguiéndose la trayectoria deseada $\boldsymbol{X^d}$.

En el esquema se analiza cómo la posición cartesiana deseada ($\boldsymbol{X^d}$) del efector final pasa por el bloque de ecuaciones del modelo geométrico inverso (MGI). Esto con el fin de obtener las variables articulares deseadas ($\boldsymbol{q^d}$), de tal forma que el controlador CTC garantice un mínimo error entre $\boldsymbol{q}$ y $\boldsymbol{q^d}$, así como un mínimo error entre $\boldsymbol{X}$ y $\boldsymbol{X^d}$.

2.3.2 Sintonización del Control CTC

La sintonización del controlador CTC se realiza por el método manual planteado en [36], de esta manera se obtienen los valores de ganancias presentados en la Tabla 2.4. Estos valores corresponden a las articulaciones activas (motorizadas).

Tabla 2.4. Ganancias para el controlador CTC.

$\boldsymbol{j}$	1	2	3	4
$\boldsymbol{Kp}$	55000	80000	80000	80000
$\boldsymbol{Kv}$	200	150	150	200

2.3.3 Seguimiento de Trayectorias

Para verificar el correcto funcionamiento de los modelos y del controlador, se realiza una prueba, la cual es ingresar una trayectoria que consiste en:

- Un circulo sobre el plano XY con radio de 0.2 y centro en (0.4, 0.4). En la Figura 2.4 se muestra el círculo con las trayectorias deseadas.

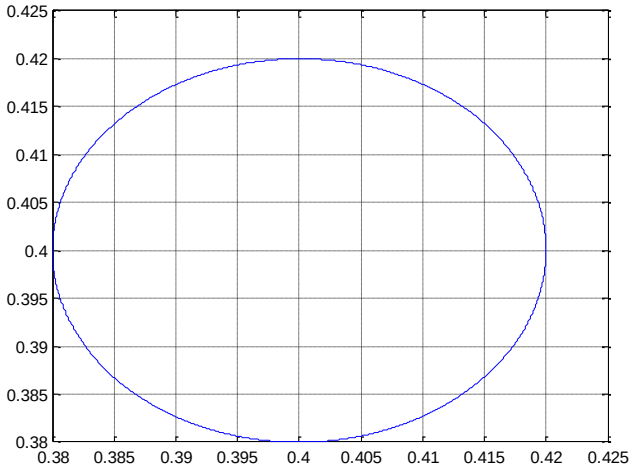

Figura 2.4. Trayectoria circular deseada.

Ahora se ingresa esta trayectoria deseadas en el controlador del HAPTIC 3R obteniendo una trayectoria real y un error cartesiano mostrado en la Figura 2.5.

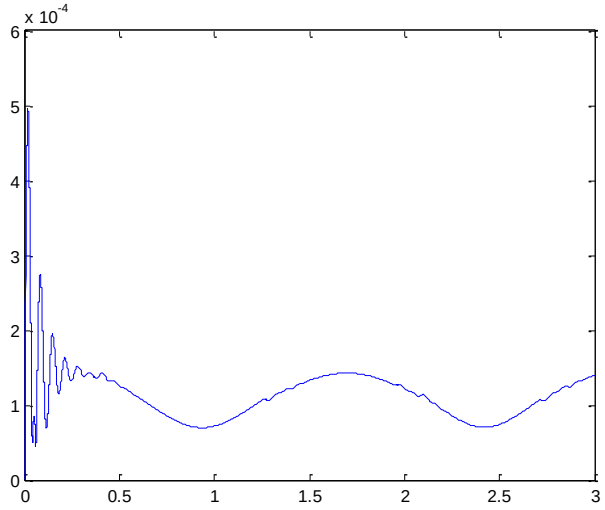

Figura 2.5. Error cartesiano.

El error de posición obtenido en la trayectoria cartesiana circular se visualiza en la figura (derecha), donde el valor de este error es menor de $5x10^{-4}$. Por tanto el HAPTIC 3R es un robot eficiente.

3. DISEÑO Y CONSTRUCCION DEL HAPTIC 3R

Este capítulo muestra el diseño, construcción y selección de materiales del HAPTIC 3R, para esta parte, se eligió el diseño de un robot tipo serie o antropomórfico, este está formado por tres ejes rotacionales (Figura 2.6), con el primer eje perpendicular al suelo y los otros dos perpendiculares a éste y paralelos entre sí.

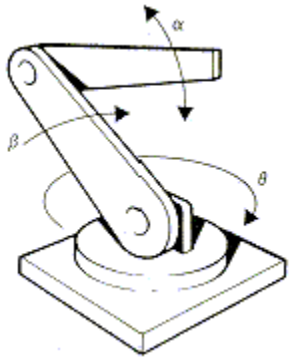

Figura 2.6. Estructura serie o tipo angular de 3 GDL [37].

Las ventajas de un robot con configuración angular presentan gran maniobrabilidad y accesibilidad a zonas con obstáculos, ocupan poco espacio con relación a su alcance, son robots rápidos, que permiten trayectorias muy complejas [37].

3.1 DISEÑO MECANICO DEL DISPOSITIVO

El diseño de un mecanismo suele presentar dificultades en la parte de control, sin embargo la aparición de herramientas computacionales como software tipo CAD proporcionan ayuda a la hora de enfrentar dicho reto. La simplicidad de algunas de estas, invita a diseñadores a explorar las capacidades para solucionar el problema de “papel en blanco”. La Figura 2.7 muestra el diseño del HAPTIC 3R mediante el software SolidEdge®.

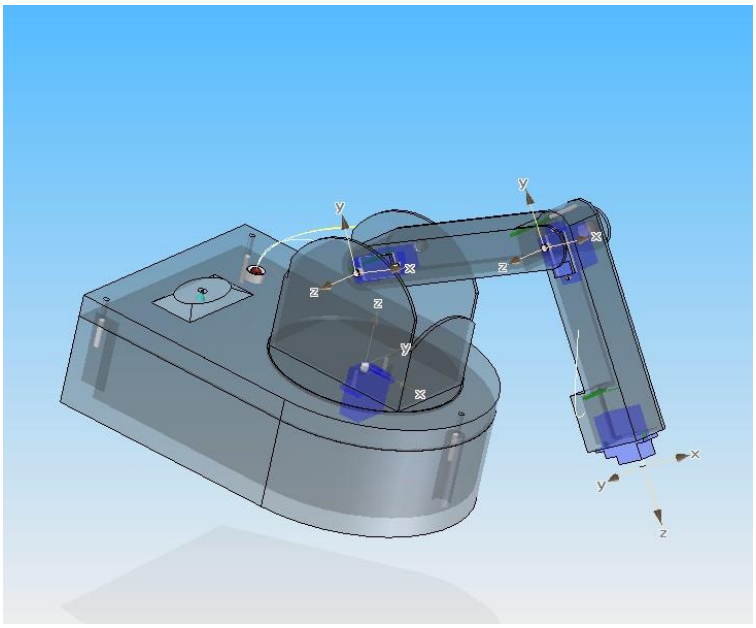

Figura 2.7. Diseño mediante herramienta CAD.

La razón de esta simulación es de obtener los datos necesarios para realizar un correcto dimensionamiento de este. Para ello los pasos realizados consisten en:

- Obtener un modelo lo más cercano posible a la estructura final.
- Definir la geometría y datos tales como masa, inercias de los eslabones.
- Recrear y definir movimientos relativos entre los eslabones del mecanismo utilizando la opción *motion* del software CAD.

A continuación se muestran los diseños en el software CAD SolidEdge® de la base (Figura 2.8) y las tres articulaciones para la rotación de los ejes θ1, θ2 y θ3 (Figura 2.9).

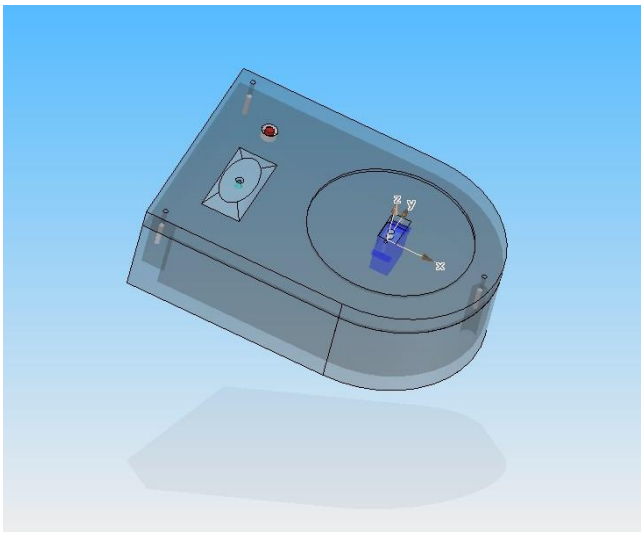

Figura 2.8. Diseños en CAD SolidEdge® de la base.

Cuerpo de rotación de θ1 Cuerpo de rotación de θ2 Cuerpo de rotación de θ3

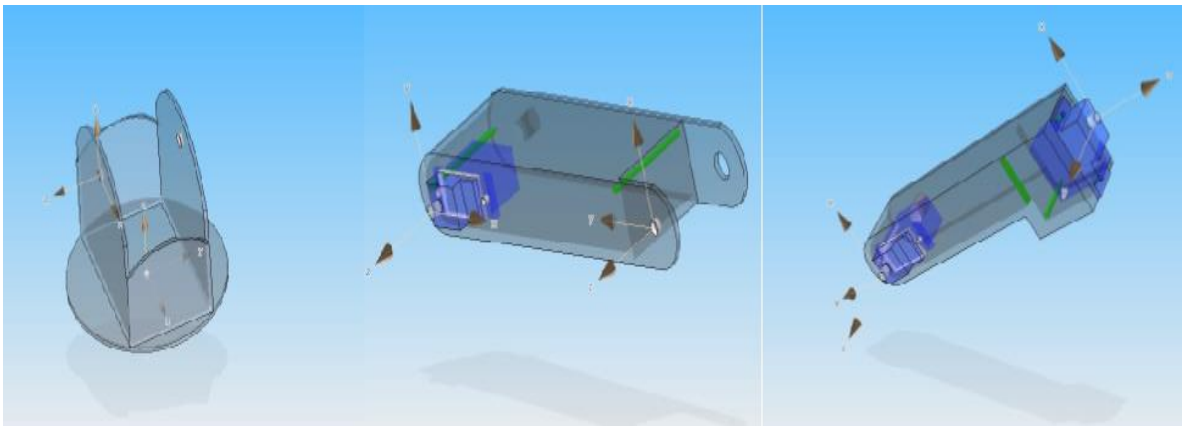

Figura 2.9. Diseños de los cuerpos para θ1, θ2 y θ3.

3.1.1 Selección del Material para la Construcción

Luego de obtener el modelo CAD, el siguiente paso es elegir el material para la construcción, por tanto el material seleccionado es de tipo *polimetilmetacrilato* o *Polipropileno, propósito general,* también es conocido como *acrílico* que es un material que presenta varias ventajas, las cuales tenemos:

- De dureza igual que el aluminio.
- Alta resistencia al impacto.
- Presenta peso ligero.
- Fácil manipulación en presencia de calor.

En la Figura 2.10, se presenta un ejemplo de los parámetros físicos dados por el software CAD utilizados en el capítulo 2 (Modelos Dinámicos).

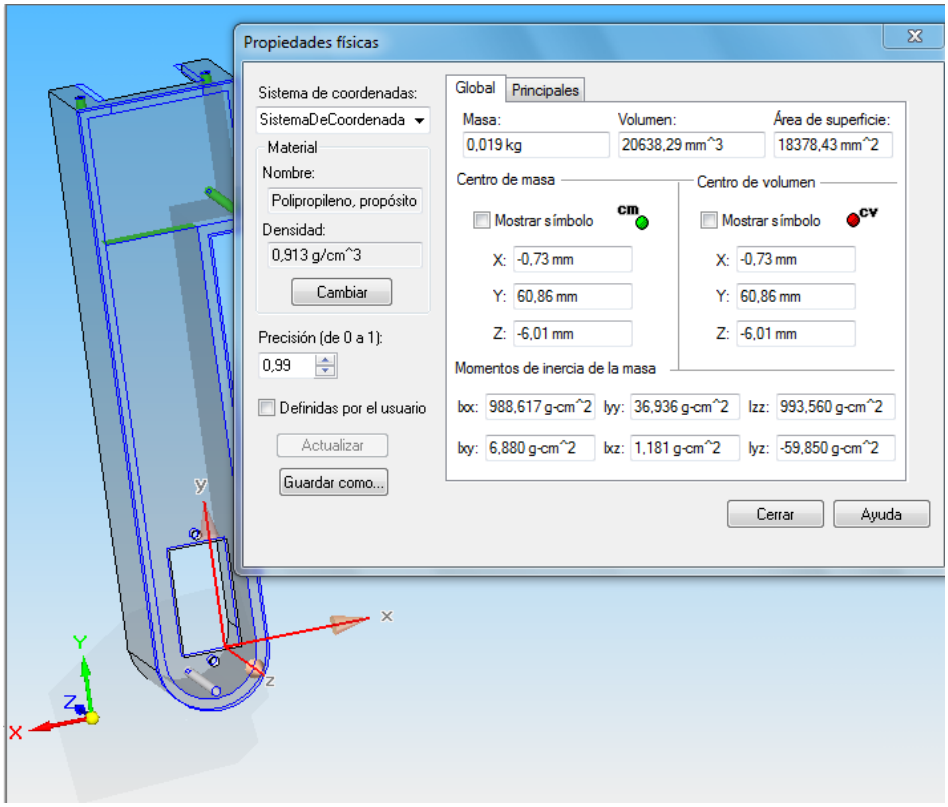

Figura 2.10. Propiedades físicas generadas por SolidEdge®.

3.1.2 Espacio de trabajo

A partir del modelos desarrollado, se inicia un proceso de “prueba y error” basado en mover el dispositivo hasta sus posiciones extremas con diferentes

orientaciones que permite realizar un estudio previo del espacio de trabajo como lo muestra la Figura 2.11 [38] .

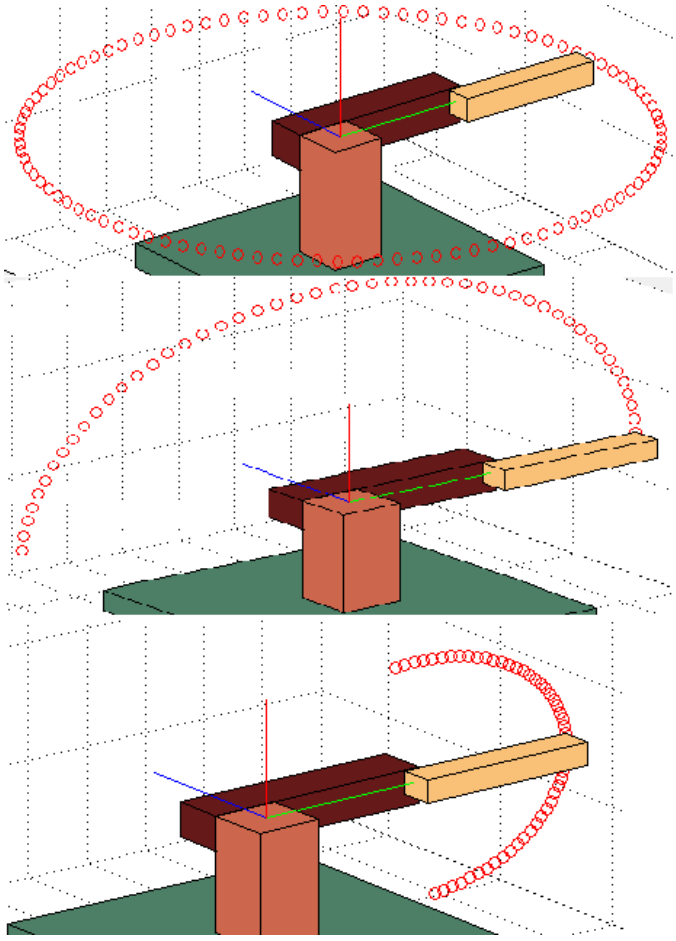

Figura 2.11. Estudio del espacio de trabajo (x,y,z) mediante simulación grafica.

El HAPTIC 3R realiza un circulo de radio12 cm (plano XY) y un semicírculo de 12cm (ejes XZ) debido a las medidas de R4 (0.10m) y D3 (0.11m).

3.1.3 Selección de Actuadores

Con los resultados de las simulaciones anteriores y además de crear un par lo suficiente para el frenado de un eslabón en un momento deseado, se decidió escoger el microservo (ver Figura 2.12). Un microservo es un dispositivo pequeño que tiene un eje de rendimiento controlado. Este puede ser llevado a posiciones angulares específicas al enviar una señal codificada. Con tal de que una señal codificada exista en la línea de entrada, el servo mantendrá la posición angular del

engranaje. Cuando la señal codificada cambia, la posición angular de los piñones cambia.

A continuación se describen las ventajas más relevantes de los microservo:

Figura 2.12. Microservos utilizados en este proyecto.

- Su tamaño. Este microservo presenta unas dimensiones de 12mm X 22mm X 31mm como se muestra en la Figura 2.13.

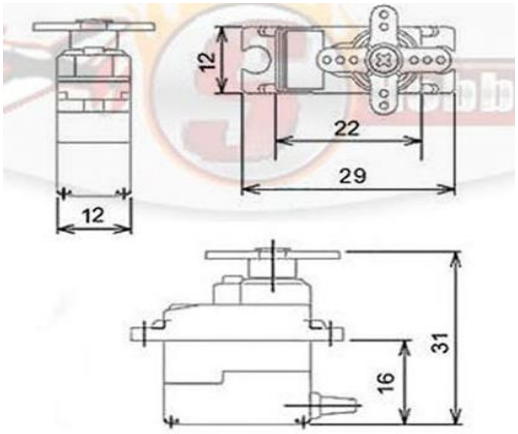

Figura 2.13. Dimensiones de los microservos.

- Su par: A pesar de sus dimensiones, este microservo aplica una fuerza 1.5Kg/cm a 4.8v en su eje de rotación, su velocidad es de 0.12 seg X 60°. Su voltaje varía de 3.0 a 6 voltios DC y un peso aproximado de 12grs.

- Control interno. Posee los suficientes elementos de control como para que se puedan monitorizar los parámetros de su actuación mecánica, como su posición, velocidad, par, etc.

Estos microservos poseen tres cables: 2 para alimentación (positivo y negativo/ground), y otro para señal de control (PWM) que determina la posición que se requiere.

Un microservo tiene un circuito de control y un potenciómetro lineal que está conectado al eje central del microservo. La función de este potenciómetro permite a la circuitería de control, supervisar el ángulo de posición actual. Si la posición no es correcta, el motor girara en la dirección adecuada hasta llegar al punto correcto. El eje del servo es capaz de girar hasta 180° o realizar un giro de 210° (según el fabricante).

La posición de Angulo es determinada por la duración de un pulso o señal PWM. Esta señal tiene un periodo de 1.75ms a 20ms (dependiendo del fabricante), y longitud del pulso en este periodo determina la posición del motor, por ejemplo, una longitud de 1.5ms hace que el microservo se posiciones a 90° como se muestra en la Figura 2.14.

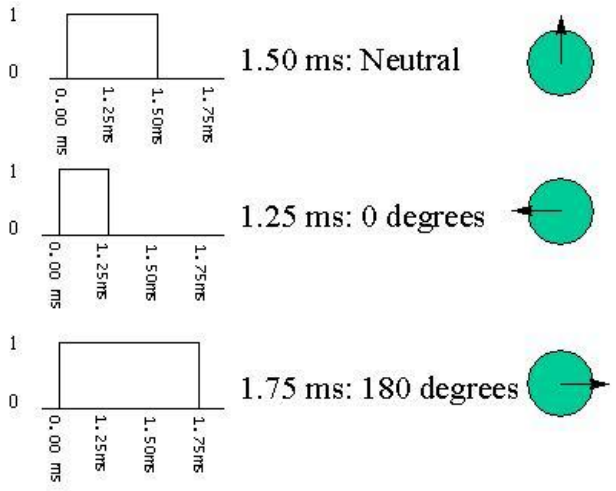

Figura 2.14. Posición del servo a un cambio de amplitud de la señal PWM.

En teoría el espacio de trabajo no presenta limitación, pero vale recordar que los microservos poseen un movimiento limitado que es de 180° o 210°, lo cual su movimiento se ve limitado.

Como resumen del diseño mecánico del dispositivo, la Tabla 3. presenta las principales características geométricas del prototipo a diseñar:

Tabla 3.1. Características Geométricas del HAPTIC 3R.

Datos geométricos	Unidades en mm
Base	200 X 140 X 65
Diámetro base inferior	104
Diámetro base Superior	100
Longitud entre base y eje eslabón 1	60
Longitud entre base y eje eslabón 2	170
Longitud entre base y efector final	275
Longitud eslabón 0	60
Longitud eslabón 1	144
Longitud eslabón 2	127
Altura máxima de extensión del dispositivo	340
Máxima rotación eje base	180°
Máxima rotación eslabón 1	180°
Máxima rotación eslabón 2	180°

3.2 DISEÑO DE LA TARJETA DE CONTROL DEL HAPTIC 3R

Este capítulo muestra el diseño de la tarjeta de control del HAPTIC 3R utilizando software de construcción de PCB Eagle®, descripción del microcontrolador utilizado, software de programación.

3.2.1 Diseño de la Tarjeta de Control

Para el desarrollo de la tarjeta de control se estableció con anterioridad las entradas y salidas que se requerían para el manejo del HAPTIC 3R, en la Tabla 3.2 se listan las entradas y salidas tanto analógicas como digitales requeridas en la tarjeta de control.

Tabla 3.2. Lista de entradas y salidas de la tarjeta de control.

Entradas	Tipo	Salidas	Tipo
Comunicación USB Requeridas 1 (Rx)	Comunicación	Señal servomotores Requeridas:4	PWM
Señal de voltaje entregada por los sensores para posición. Requeridos: 4	Analógica	Indicadores lumínicos (leds) Requeridos:6	Discreta
		Comunicación USB Requeridas 1 (Tx)	comunicación

A continuación se describe el uso de las entradas y salidas utilizadas.

- **Señal servomotores**: como ya se ha mencionado en el capítulo 2, el HAPTIC 3R utiliza cuatro servomotores, tres para posición y uno de

orientación. Cada uno de estos motores con controlados por señales PWM independientes, en este caso se requiere generar cuatro señales.

- **Indicadores lumínicos**: estos son los encargados de indicar al usuario eventos realizados, tales son:

  - Un indicador (biled) para establecer si la comunicación USB (PC-HAPTIR 3R) esta activa o no.
  - Un indicador para el servomotor 1.
  - Un indicador para el servomotor 2.
  - Un indicador para el servomotor 3.
  - Un indicador para el servomotor 4.
  - Un indicador que indica que el HAPTIC 3R se puede manejar libremente (PWM=0).

- **Señal de voltaje entregada por los sensores de posición**: estos sensores indican la posición actual de los servomotores, para dicho tarea, se decide escoger como sensores los potenciómetros lineales internos de los servomotores, con el fin de evitar instalar sensores como encoders. En total son 4 sensores o señales leídas para el control de posición y orientación.

- **Comunicación USB**: se decidió utilizar la comunicación USB 2.0. USB (Universal Serial Bus) es una interfaz plug&play entre la PC y ciertos dispositivos tales como teclados, mouses, scanner, impresoras, módems, placas de sonido, cámaras, etc) .

Una característica importante del USB es que permite a los dispositivos trabajar a velocidades mayores, en promedio a unos 12 Mbps, esto es más o menos de 3 a 5 veces más rápido que un dispositivo de puerto paralelo y de 20 a 40 veces más rápido que un dispositivo de puerto serial.

Trabaja como interfaz para transmisión de datos y distribución de energía, que ha sido introducida en el mercado de PC´s y periféricos para mejorar las lentas interfaces serie (RS-232) y paralelo. Esta interfaz de 4 hilos, 12 Mbps y "plug and play", distribuye 5V para alimentación, transmite datos y está siendo adoptada rápidamente por la industria informática [39].

Teniendo en cuenta las anteriores características y del tipo de comunicación a manejar, se decide utilizar una gama alta de microcontroladores de la casa

fabricante MICROCHIP® el cual dispone la familia de los PIC18F**50, que son microcontroladores que utilizan el protocolo de comunicación USB. Ahora, teniendo en cuenta las entras y salidas a utilizar, se concluye en utilizar un PIC DIL40 (40 pines), esto con el fin de cubrir todas las entradas y salidas necesitadas por la tarjeta de control, dicho microcontrolador seleccionado es el PIC18f4550.

El microcontrolador PIC18f4550, pertenece a los controladores PIC18 de gama alta. Posee una arquitectura *RISC* (Reduced Instruction Set Computer) de 16 bits de longitud de instrucciones y 8 bits de datos. La Tabla 3.3 resume las características fundamentales de este microcontrolador, en la Figura 2.15 se muestra la arquitectura física del 18f4550.

Tabla 3.3. Características principales del PIC 18f4550.

Características	**PIC 18F4550**
Frecuencia de operación	Hasta 48 Mhz
Memoria de Programa (bytes)	24.576
Memoria Ram de datos(bytes)	2.048
Memoria EEPROM datos (bytes)	256
Interrupciones	20
Líneas de E/S	35
Temporizadores	4
Módulos de comparación/captura/PWM(CCP)	1
Módulos de comparación/captura/PWM mejorado (ECCP)	1
Canales de comunicación serie	MSSP.EUSAR
Canal USB	1
Puerto paralelo de transmisión	1

de datos (SPP)	
Canales de conversión A/D de 10 bits	13 canales
Comparadores analogicos	2
Juego de instrucciones	75(83 ext)
Encapsulados	PDIP 40 pines QFN 40 pines TQFP 40 pines

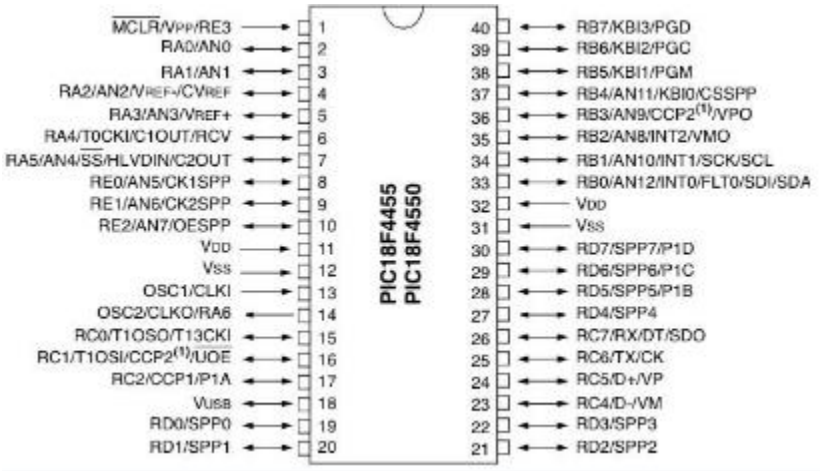

Figura 2.15. Distribución de pines del PIC 18f4550.

Luego de seleccionar el microcontrolador, se dispone al diseño de la tarjeta bajo el software para PCB llamado EAGLE®, para dicho diseño se requieren todos los componentes electrónicos a utilizar que son mostrados en la Tabla 3., para dicho diseño se tuvo en cuenta la dimensión de la base, voltaje de alimentación, localización de los componentes de alimentación, programación del microcontrolador y conector del puerto USB. En la Tabla 3. se listan los demás componentes utilizados.

Tabla 3.4. Lista de componentes electrónicos.

Cantidad	Elemento electrónico
2	Condensadores cerámicos (27 pf)
2	Condensadores cerámicos (104 pf)
1	Condensador Electrolítico (47 uf)
1	Condensador Electrolítico (470uf)
6	Resistencias (220ohm)
1	Resistencia (15 kohm)
1	(biled) led bicolor con ground común
5	Leds
1	Microcontrolador (18f4550)
1	Regulador de voltaje 7805 (+5v DC)
1	Cristal (20 Mhz)

Tabla 3.5. Lista de componentes no electrónicos.

Cantidad	Componente
1	Switch (NA)
1	Conector (comunicación USB)
1	Conector RJ11(programación)
1	Conector (alimentación +9 V DC)
4	Conectores (4pines)
1	Conector (3pines)
6	Conectores
1	Base DIL40 (base microcontrolador)

En la Figura 2.16 se tiene el diagrama de bloques de la tarjeta de control cuya dimensiones físicas son (135x95 mm) además, en la Figura 2.17 se muestra como un diseño preliminar en 3D de la tarjeta de control (en el Anexo D se pueden detallar los planos esquemáticos de la tarjeta de control).

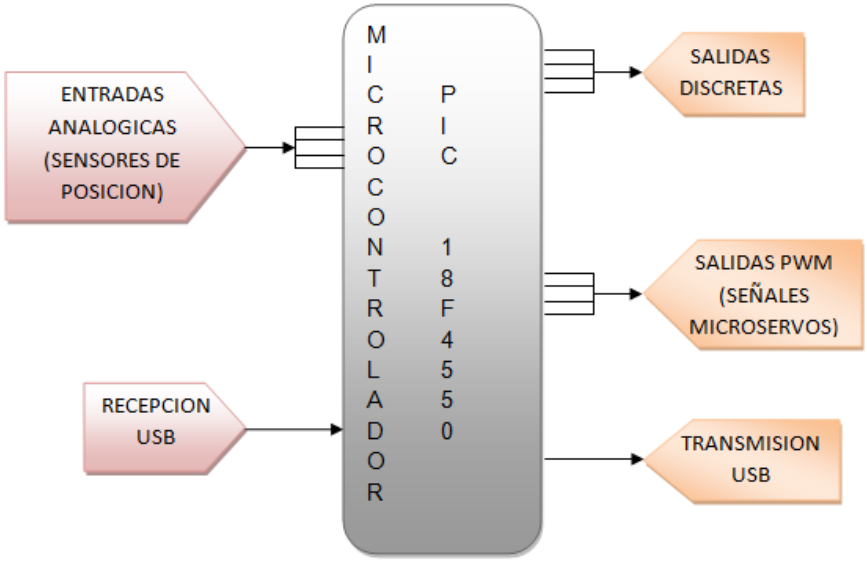

Figura 2.16. Diagrama de bloques de la tarjeta de control.

Figura 2.17. Diseño 3D de la tarjeta de control.

3.3 DESARROLLO DEL PROGRAMA (SOFTWARE) PARA EL MICROCONTROLADOR

Esta etapa del proyecto describe la utilización de herramientas software para crear el código requerido para el control de HAPTIC 3R. MICROCHIP® entrega gratuitamente un paquete de programación, en la cual se encuentran las plataformas de programación MPLAB, PIC C y PICKIT2 que es el software para el programador de microcontroladores PICKIT2.

En el Anexo B se explica cómo se incluye PIC C como compilador principal a MPLAB. PIC C es un software para realizar código en lenguaje C para microcontroladores, y la razón de anexar este a MPLAB es para utilizar las herramientas que proporciona MPLAB para realizar comprobación de código (Figura 2.18).

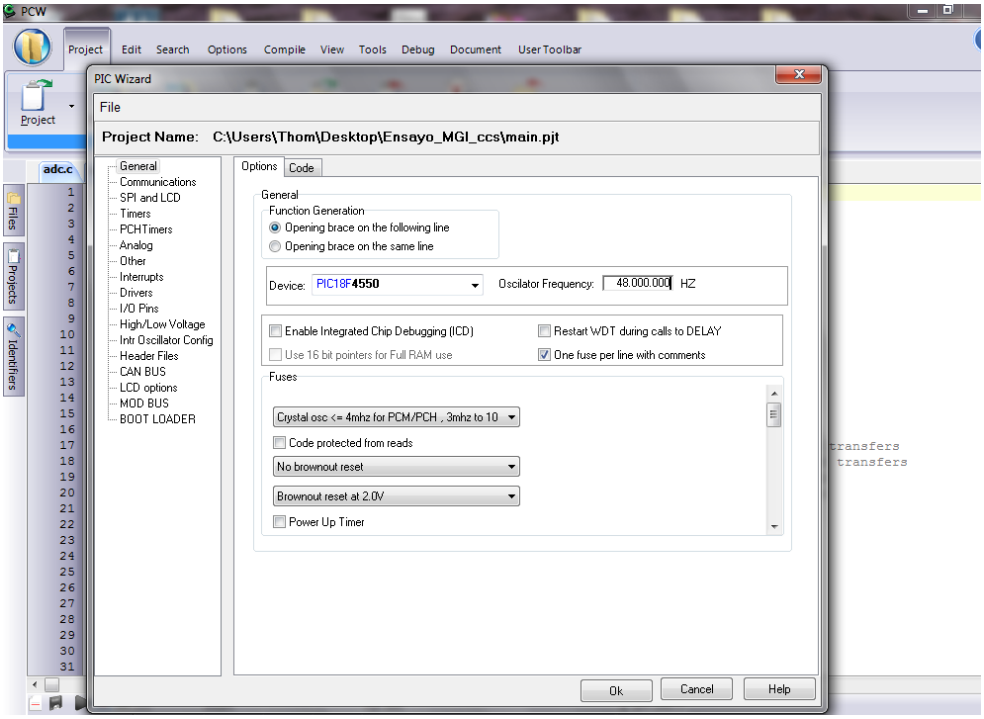

Figura 2.18. Pantalla de opciones de PIC C (CCS)

En la parte de simulación del código, se utiliza el software PROTEUS®, esta posee librerías que simulan el comportamiento de componentes electrónicos, tales como resistencias, capacitores y microcontroladores, además tiene la opción de

instalar un puerto USB virtual, el cual es utilizado para simular la comunicación USB entre el microcontrolador virtual y computador.

Luego de crear los archivos HAPTIC_3R.c (archivo principal) y HAPTIC_3R.h (archivo de cabecera) en PIC C y anexarlos al proyecto creado en MPLAB (ver Anexo B), se procede a desarrollar el código. El encabezado de este debe tener habilitado las opciones del ADC (8bits) y comunicación USB.

3.3.1 Comunicación USB (Configuración en el Microcontrolador)

La configuración del modulo USB en el microcontrolador se realiza configurando mediante descriptores (configurado en el archivo HAPTIC_3R.h) y de las configuraciones internas del (archivo HAPTIC_3R.c).

3.4 DESCRIPCION DEL ALGORITMO

3.4.1 Creación de Señales PWM para los Microservos

Para resumir lo ya dicho en el link 3.1.3 sobre servomotor, se tiene que este se controla mediante una señal PWM con una frecuencias de 50Hz, con un periodo de 20ms y con un Duty Cycle que varía entre 6% y 15% entre un periodo en alto de unos 0.5ms a 2.5ms.

Con un pulso de aproximadamente 0.7 ms (0º ó +90º), el microservo se posiciona en un extremo de su recorrido, con un pulso de 2.2ms (180º ó -90º) se posiciona en el extremo contrario, y con un pulso de 1.5ms (90º ó 0º) se posiciona en el centro de su recorrido (Figura 2.19).

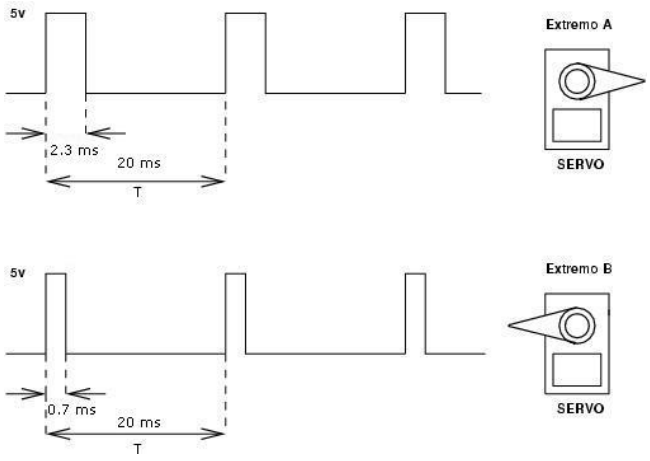

Figura 2.19. Posición de un servo en variación de la señal PWM.

Para iniciar el cálculo de tiempos se debe tener en cuenta la frecuencia del pulso PWM de cada microservo (50 Hz). Luego cada microservo tiene un periodo o tiempo que transcurre en dos flancos sucesivos (ecuación (3.1)):

$$\mathrm{T} = \frac{1}{\mathrm{F}} = \frac{1}{50} = 0.02\mathrm{s} = 20\mathrm{ms} \qquad (3.1)$$

Este proyecto tiene la capacidad de manejar 8 servos, por tanto el periodo se puede dividir entre 8 para obtener lo que se denominara como "ventana" para cada servo (ecuación (3.2)).

$$\mathrm{W} = \frac{20}{8} = 2.5\mathrm{ms} \qquad (3.2)$$

Por cada una de estas ventanas de tiempo (2.5ms) de duración se controlara cada servo, uno por uno hasta completar 8 servos. Esto implica que independientemente del tiempo que este en alto el pulso del microservo y luego de que ha transcurrido 2.5ms, el pulso se pone en bajo para el siguiente servo. En este caso son 8 servos por tanto son 8 ventanas de 2.5 ms. Esto significa el uso de una única interrupción generada por el Timer1 (rodando a 16 bits de precisión).

En la Figura 2.20 se muestra las señales generadas de cada uno de los microservos de forma sucesiva pero separadas cada una de la siguiente en 2.5 ms hasta completar los 20 ms con los 8 servos [40].

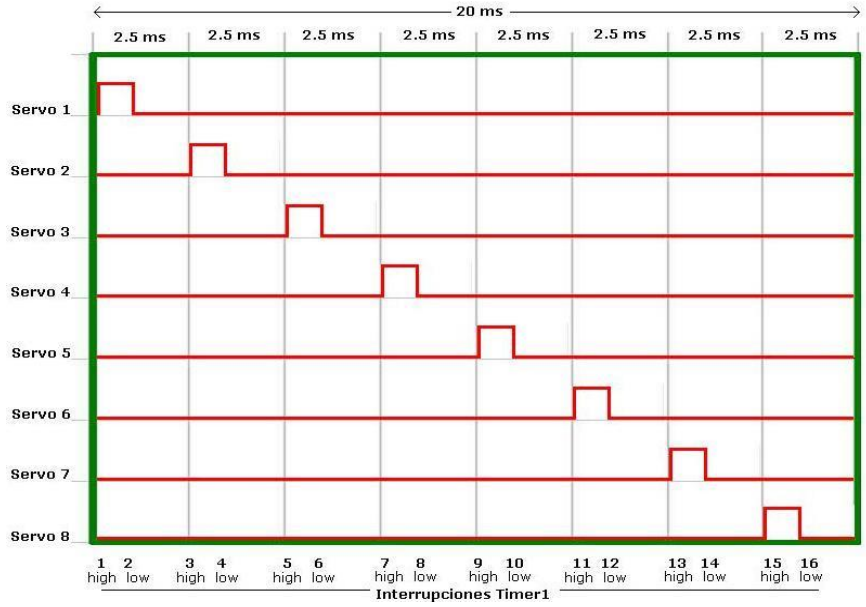

Figura 2.20. Interrupciones mediante Timer1.

Para el cálculo del Timer 1 configurado como contador de 16 bits, se utilizo el software gratuito Timer Calculator, el cual se configura para un cristal de 20Mhz y un preescaler de 1:1 (Figura 2.21).

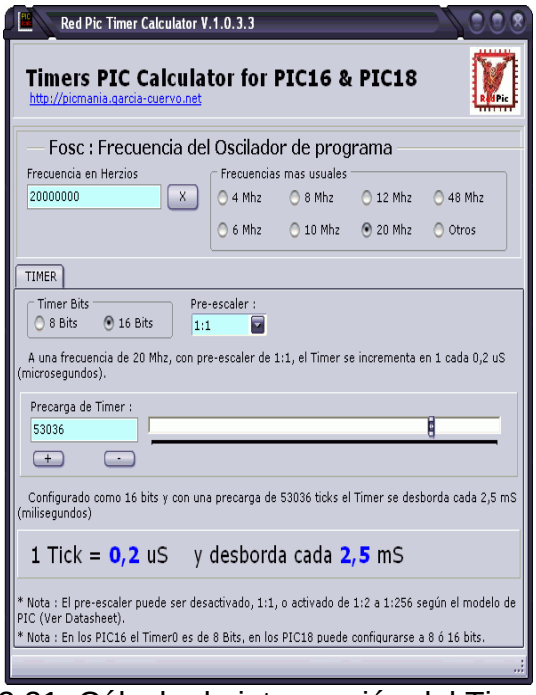

Figura 2.21. Cálculo de interrupción del Timer1 [41].

El incremento de tiempo o denominado Tick (denominado por el creador del software Timer Calculator [41]) del Timer1 es de 0.2us (0.0002 ms). Así se expresaría en unidades de Ticks los tiempos en alto de cada microservo, por ejemplo, si todos los servos estuvieran en el centro de su recorrido, su respectiva señal PWM debería ser de 1.5 ms en alto entonces en la ecuación (3.3) se muestra la cantidad de ticks requeridos:

$$Tx = \frac{1.5ms}{0.0002} = 7500\ Ticks \qquad (3.3)$$

Por tanto, si se habilita la interrupción por desbordamiento del Timer1, en su primer desborde, se coloca en alto el pin correspondiente al microservo1 y si se precarga el timer1 con 7500 tickets antes del desborde (este desborde se genera cada 65535 o cada 16 bits), entonces se coloca el Timer1 a 58035 (65535 -7500), de esta manera se obtiene la siguiente interrupción cuando transcurra los 7500 ticks.

Cuando la interrupción salta de nuevo y repite el ciclo por segunda vez, se tiene que prefijar el tiempo que ha transcurrido para completar la ventana de 2.5 ms hasta el comienzo del control siguiente del microservo. Anteriormente se planteo un ejemplo utilizando un valor de 1.5 ms o 7500 tickets ahora se tiene que esperar 1ms (2.5ms – 1.5 ms) o 5000 tickets, por lo que se pre-cargara de nuevo el timer1 con 60535(65535-5000) y así la interrupción saltara de transcurrido 1ms.

3.4.2 Descripción Archivo haptic_3r.c

Para la realización del código en C, se utilizara la directiva #fuse fast_io() fijando el funcionamiento de los pines con el set_tris_(puerto)(). Esto reduce instrucciones innecesarias como output_low() o output_high(). En el Anexo B se lista el código correspondiente del HAPTIC 3R

La configuración del HAPTIC_3R.c se realiza en las siguientes líneas de código:

- **#fuses HSPLL y PLL5:** la frecuencia de oscilación necesaria para el USB 2.0 es de 48Mhz, pero como el HAPTIC 3R utiliza un cristal de 20 Mhz se necesita hacer uso del modulo *PLL* interno del PIC. Para ello se utiliza el *FUSE HSPLL*. Como el modulo *PLL* requiere una oscilación de entrada de 4 Mhz, se debe utilizar el divisor 1:5 indicado con el *FUSE PLL5* para obtención de 20:5=4Mhz requeridos.

- **USB_ENABLE_BULK y SIZE 32:** el método de transferencia masiva (datos) se activa mediante la configuración de los *Endpoint* de transmisión y recepción *USB_EP1_TX_ENABLE* y *USB_EP1_RX_ENABLE* indicándolo con *USB_ENABLE_BULK*. Se recomienda deshabilitar el método *HID* (Human Interface Device). El tamaño de buffer de transferencia se puede ajustar entre 1 a 32 bytes como máximo. En este caso se configura a 32 bytes (*SIZE32*).

- **main():** aquí se encuentran las funciones *usb_init(), usb_task, usb_wait_for_enumeration(), usb_enumerated, usb_kbhit(), usb_get_packet() y usb_put_packet()* que son las funciones utilizadas para el manejo del USB 2.0 y estan implementadas en los archivos *usb.c, usb.h, pic18_usb.c y pic18_usb.h* que están por defecto en el compilador (PIC C)

Las funciones *usb_init(), usb_task() y usb_wait_for_enumeration()* se utilizan para establecer la comunicación y se ejecutan al iniciar el funcionamiento del microcontrolador. La comunicación entre microcontrolador y PC esta activa cuando la funcion *usb_enumerated()* nos devuelve un valor verdadero (true). Para verificar de forma visual si la comunicación esta activa, se realiza el encendido o apagado de un led que indique el estado de esta, en la Figura 2.22 se muestra un ejemplo en el cual, al encender la alimentación del microcontrolador se encenderá el led (LEDR) y se apagara el led LEDV, luego si hay interrupción o comunicación USB, la función usb_wait_for _enumeration() es verdadera (true) encendiendo el led LEDV y apagando el LEDR. (LEDV y LEDR se especifican (comentarios de código) en el Anexo B).

```
LED_OFF(LEDV);  //encendemos led que indica USB OFF
LED_ON(LEDR);

usb_init();                       //inicializamos el USB

usb_task();                       //habilita periferico usb e interrupciones
usb_wait_for_enumeration();       //esperamos hasta que el PicUSB sea configurado por el host

LED_OFF(LEDR);
LED_ON(LEDV);                     ///encendemos led que indica USB ON

while (TRUE)
{

   if(usb_enumerated())           //si el PicUSB está configurado
   {
      if (usb_kbhit(1))           //si el endpoint de salida contiene datos del host
      {
```

Figura 2.22. Configuración e indicación visual del estado USB.

Para detectar algún comando enviado desde el PC hacia el microcontrolador, se utiliza la función usb_kbhit() que si retorna verdadero nos indicara que tiene algo pendiente por recibir y en caso de no incluir esta función, se pueden recibir datos erróneos. Para almacenar estos datos se realiza mediante la función usb_get_packet() quedando disponible en recbuf. La configuración de esta función es usb_get_packet (endpoint, ptr, max), donde:

- **endpoint**: es el buffer de entrada donde llega el paquete de datos.
- **ptr**: apuntador donde se guardara los datos leídos.
- **max**: tamaño del paquete en bytes

Para el envió de datos, se emplea la función *usb_put_packet(endopoint,ptr,len,toggle),* donde:

- **endpoint**: buffer de entrada y salida.

- **Ptr:** apunta a la dirección del dato a enviar (variable declarada).

- **Len**: tamaño del paquete en bytes (máx. 64).

- **Toogle**: parte de transmisión de datos, los paquetes de datos se encuentran en grupos de paquetes de datos, y dentro de estos existen unos llamados *DATA0, DATA1*. Existe un proceso llamado sincronización del *data toggle*. Esto es un método de validación de paquetes y su función es el de enviar alternadamente a *DATA0 Y DATA1* en una secuencia seguido de su ACK respectivo. Lo anterior es con el objetivo de mantener la sincronización entre transmisor y receptor.

3.4.3 Descripción Archivo haptic_3r.h

En este archivo se incluyen las estructuras y parámetros necesarios para una correcta conexión con el PC. A continuación se explican los conceptos de *VID&PID* y de la modificación de la tabla *USB_STRING_DESC[].*

- **El VID** es un número de 16 bits que significa *Vendor Identification* o código que identifica al fabricante del hardware a conectar. Se utilizara el numero 04d8h que identifica al MICROCHIP®.

- **El PID** es un número de 16 bits que significa *Product Identification* o código que identifica al dispositivo o hardware a conectar con el PC. En este caso se utiliza el numero 000Bh que identifica a la familia de los PIC18fxxxx.

- **USB_STRING_DESC[]:** la tabla del *USB_STRING_DESC[]* contiene la descripción del dispositivo detectado por el Driver de Windows . consta de dos partes o tablas*: USB_STRING_DESC* que contiene las descripciones requeridas y *USB_STRING_DESC_OFFSET* que contiene los offset, o desplazamientos con respecto al inicio de USB_STRING_DESC en donde se encuentran las correspondientes cadenas.

En el Anexo B se enlista el código correspondiente al archivo HAPTIC_3R.h al igual de un pequeño tutorial para la instalación de los drivers bajo WINDOWS XP (32 bits)

4. DISEÑO DELA INTERFAZ DE USUARIO E INTERFAZ 3D PARA EL HAPTIC 3R

Luego de tener la configuración del microcontrolador para comunicación USB, control de los micros servos y conversores ADC, se desarrollo una interfaz grafica de usuario y una interfaz 3D con el fin de interactuar y comprobar la detección de colisiones con el HAPTIC 3R.

4.1 INTERFAZ DE USUARIO E INTERFAZ 3D

Al realizar el análisis de los requerimientos del proyecto y de posibles proyectos posteriores, se concluye que la herramienta computacional para dicha interfaz de usuario e interfaz 3D debe cumplir con las siguientes características mínimas.

- De distribución libre, que permita usarlo bajo una licencia gratuita, de tal forma que el sistema a desarrollar quede libre de compromisos con empresas desarrolladoras de este tipo de software.

- Que permita generar un archivo ejecutable para que la interfaz de usuario pueda ejecutarse sin necesidad de compilar el código cada vez que se requiera.

- Que tenga la opción de crear interfaces gráficas amigables para que sea comprensible y manipulable por cualquier persona.

- Que soporte las 2 principales API (Interfaces de Programación de Aplicaciones) graficas que son Opengl (Open Graphics Library) y Direct3d, de tal forma que pueda ser usado en ambiente Windows.

Teniendo en cuenta las anteriores características, se decidió escoger la plataforma de programación MICROSOFT® Visual Studio Express 2008 (Figura 4.1) que es un programa de desarrollo en entornos de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, ASP.NET y Visual Basic [42]. Es de carácter gratuito. y Opengl que es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D [43].

Figura 4.1. Visual Studio Express 2008.

4.2 CONSTRUCCION DE LA INTERFAZ DE USUARIO

Para realizar la interfaz de usuario utilizando Visual Studio Express 2008, se utilizo el lenguaje C# que es un lenguaje de programación diseñado para crear una amplia gama de aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Con sus diversas innovaciones, C# permite desarrollar aplicaciones rápidamente y mantiene la expresividad y elegancia de los lenguajes de tipo C.

Visual Studio admite Visual C# con un editor de código completo, plantillas de proyecto, diseñadores, asistentes para código, un depurador eficaz y fácil de usar, además de otras herramientas. La biblioteca de clases .NET Framework ofrece acceso a una amplia gama de servicios de sistema operativo y a otras clases útiles y adecuadamente diseñadas que aceleran el ciclo de desarrollo de manera significativa [44].

4.2.1 Creación de una Ventana en Visual C#

Después de crear un proyecto o una Aplicación de Windows Forms (Ver Anexo C), se procede a crear TextBox y ProgressBar (Figura 4.2). Estos tiene la función de mostrar al usuario el valor digital enviado por el microcontrolador (ADC) de la lectura de los potenciómetros internos de los microservos, esto con el fin de relacionar estos valores con el MGI que cuyos valores son mostrados en los Texbox (Figura 4.3).

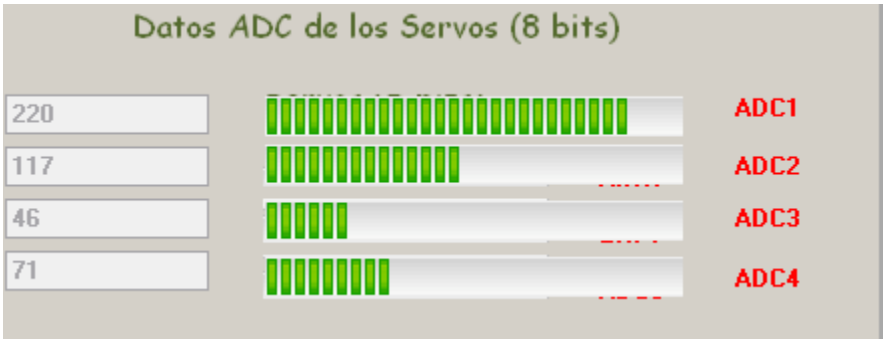

Figura 4.2. Datos del ADC mostrados en TextBox y en ProgressBar.

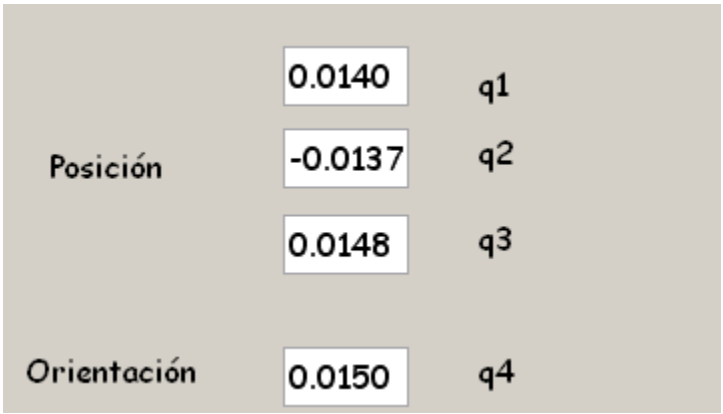

Figura 4.3. Valores entregados por el MGI.

Para el observar el cambio de los microservos por parte del usuario, se implementan *TrackBar* o *barra de desplazamiento* con su respectivo *Textbox* para

indicar el Ticket actual o deseado para que el servo se posicione en este (Figura 4.4).

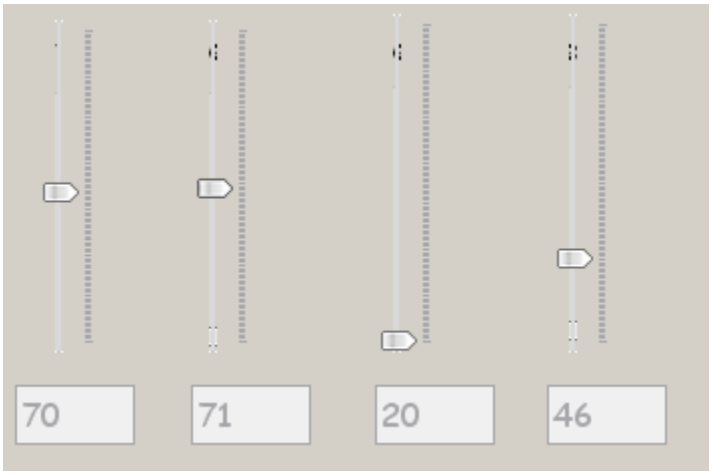

Figura 4.4. Lectura de las posiciones de los microservos.

En la Figura 4.5 se muestra la interfaz de Usuario final.

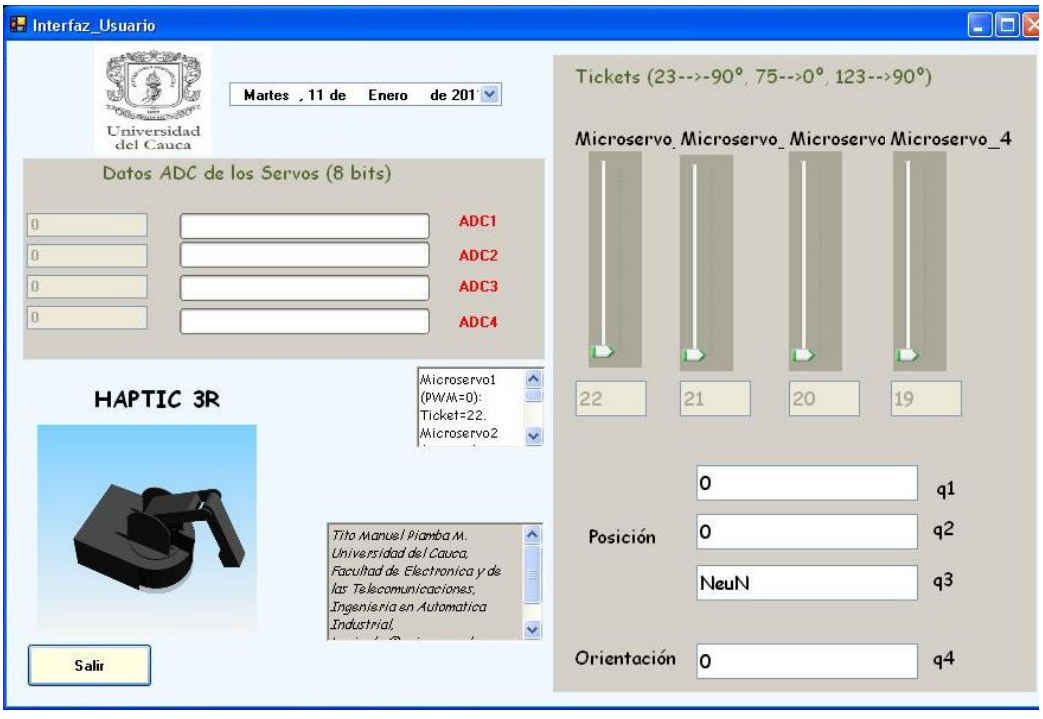

Figura 4.5. Interfaz de Usuario del HAPTIC 3R.

4.2.2 Configuración USB (pc-microcontrolador)

Después de obtener la ventana de interfaz de usuario, se procede a enlazar la librería gratuita de MICROCHIP® llamada mpusbapi.dll que es la librería encargada de vincular a la familia de los PIC18f**50 como dispositivo de Windows. En el proyecto de Visual C# creado anteriormente, se agregó el archivo *PicUSBAPI.cs*, el cual tiene la configuración de la librería mpusbapi.dll para interactuar con la interfaz de usuario. Esta configuración inicia agregando la clase (`using System.Runtime.InteropServices;`) que se encarga de importar alguna DLL. Ahora se configura la librería para habilitar la transmisión/recepción, velocidad de transmisión/recepción, datos a enviar/recibir (bits) en las siguientes líneas de código:

```
        #region Funciones importadas de la DLL: mpusbapi.dll
         [DllImport("mpusbapi.dll")]
         private static extern DWORD _MPUSBGetDLLVersion();
         [DllImport("mpusbapi.dll")]
         private static extern DWORD _MPUSBGetDeviceCount(string
pVID_PID);
         [DllImport("mpusbapi.dll")]
         private static extern void* _MPUSBOpen(DWORD instance, string
pVID_PID, string pEP, DWORD dwDir, DWORD dwReserved);
         [DllImport("mpusbapi.dll")]
         private static extern DWORD _MPUSBRead(void* handle, void* pData,
DWORD dwLen, DWORD* pLength, DWORD dwMilliseconds);
         [DllImport("mpusbapi.dll")]
         private static extern DWORD _MPUSBWrite(void* handle, void*
pData, DWORD dwLen, DWORD* pLength, DWORD dwMilliseconds);
         [DllImport("mpusbapi.dll")]
         private static extern DWORD _MPUSBReadInt(void* handle, DWORD*
pData, DWORD dwLen, DWORD* pLength, DWORD dwMilliseconds);
         [DllImport("mpusbapi.dll")]
         private static extern bool _MPUSBClose(void* handle);
         #endregion
```

Para habilitar el puerto se debe utilizar la función:

```
         public void OpenPipes()
         {
             DWORD selection = 0;

          myOutPipe = _MPUSBOpen(selection, vid_pid_norm, out_pipe, 0, 0);
           myInPipe = _MPUSBOpen(selection, vid_pid_norm, in_pipe, 1, 0);
         }
```

Luego de habilitado el puerto, se procede a recepcionar :

```
       private void ReceivePacket(byte* ReceiveData, DWORD
*ReceiveLength)
```

```
            {
                  uint ReceiveDelay=1000;

                  DWORD ExpectedReceiveLength = *ReceiveLength;

                  OpenPipes();
            _MPUSBRead(myInPipe, (void*)ReceiveData,
ExpectedReceiveLength, ReceiveLength, ReceiveDelay);
            ClosePipes();
        }
```

Y la escritura es habilitada en la función:

```
        private void SendPacket(byte* SendData, DWORD SendLength)
        {
            uint SendDelay = 1000;

            DWORD SentDataLength;

            OpenPipes();
            _MPUSBWrite(myOutPipe, (void*)SendData, SendLength,
&SentDataLength, SendDelay);
            ClosePipes();
        }
```

Ahora, luego de obtener la escritura y lectura de datos, se procede a cerrar el puerto, de lo contrario se puede obtener información errónea o perdida de la comunicación USB.

```
        public void ClosePipes()// Cerramos el puerto
        {
            _MPUSBClose(myOutPipe);
            _MPUSBClose(myInPipe);
        }
```

Para enlazar el dispositivo con la ventana creada anteriormente (Interfaz de Usuario), se crean las funciones de lectura/escritura y se envían a la configuración anterior. Para la lectura de datos procedentes del microcontrolador se crea una función que especifique la variable en donde se almacenaran, el tamaño del envió y el retorno de datos:

```
public uint LeeADC2()
        {
            uint result1 = 0;
            uint result2 = 0;
            uint result3 = 0;
            uint result4 = 0;
            byte* receive_buf = stackalloc byte[2];

            DWORD RecvLength = 4;
```

```
            ReceivePacket(receive_buf, &RecvLength);
            result1 = receive_buf[0];
            result2 = receive_buf[1];
            result3 = receive_buf[2];
            result4 = receive_buf[3];
            return result2;
            //return result2;
        }
```

La función *LEEADC()* obtiene la lectura de 4 datos, y retorna uno de ellos (`return result2;`). La invocación se realiza desde el archivo creado al hacer doble click en alguna de las herramientas de la ventana creada (*TextBox, TrackBar, etc*).

Para el envió de datos desde el PC hacia el PIC, se realiza creando la función que se encarga de almacenar y enviar información cuando el puerto USB esta activo:

```
public void PideADC(uint posicion, uint posicion2, uint posicion3, uint
posicion4)
        {
            byte* send_buf = stackalloc byte[4];
            send_buf[0] = (byte)posicion;
            send_buf[1] = (byte)posicion2;
            send_buf[2] = (byte)posicion3;
            send_buf[3]=(byte) posicion4;
            SendPacket(send_buf, 4);
        }
```

Esta función almacena cuatro datos enviados desde la PC y los almacena en el archivo (send_buf[]).

Luego de crear las funciones de enlazamiento, se procedió a crear una función que fuera capaz de leer/enviar datos continuamente entre PC-PIC, por tanto se utilizó un TIMER que se encuentra en el cuadro de herramientas(ver Figura 4.6), este se debe habilitar (Enabled:True), indicarle el tiempo (en ms) que se desea para que sea invocado (Interval:(time ms)) y luego en la función propia del timer, se realiza la lectura continua enviada desde el microcontrolador y la escritura continua enviada desde el PC.

```
private void timer1_Tick(object sender, EventArgs e)
        {
            int lectura;
            posicion4.Text = trackBar4.Value.ToString();
            posicion3.Text = trackBar3.Value.ToString();
            posicion2.Text = trackBar2.Value.ToString();
            posicion.Text = trackBar1.Value.ToString();
            usbapi.PideADC(uint.Parse(posicion.Text),
uint.Parse(posicion2.Text), uint.Parse(posicion3.Text),
uint.Parse(posicion4.Text));
```

```
            lectura=(int)usbapi.LeeADC();
            progressBar.Value =lectura;
            adctxt.Text = lectura.ToString();
        }
```

La anterior es invocada cada tiempo que se le es declarado, en el momento de ser llamada, este envía las posiciones (ticks) generadas por los TrackBar `(usbapi.PideADC(uint.Parse(posicion.Text), uint.Parse(posicion2.Text), uint.Parse(posicion3.Text), uint.Parse(posicion4.Text));)` hacia el PIC y en `(lectura=(int)usbapi.LeeADC();)` almacena en (lectura) el primer ADC enviado desde el PIC.

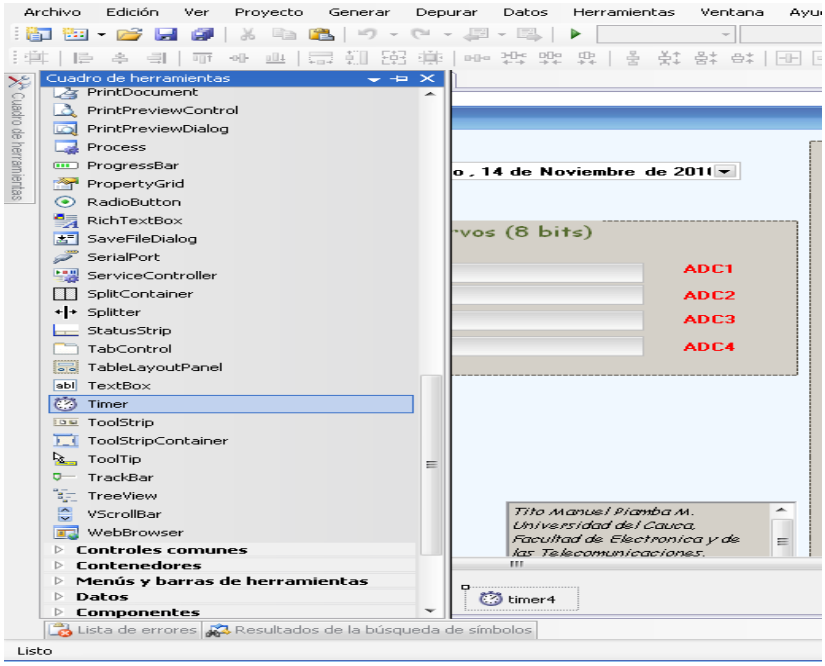

Figura 4.6. Configuración del Timer en C#.

4.2.3 Simulación del la Interfaz de Usuario

Para verificar la comunicación USB entre el Microcontrolador y el PC, se realizó una simulación creada desde el software PROTEUS (ISIS) que es una compilación de programas de diseño y simulación electrónica, desarrollado por LABCENTER ELECTRONICS.

En la Figura 4.7 se implementó el circuito de la tarjeta de control (ítem 3.2.1), con la diferencia que los potenciómetros internos de los microservos se remplazaron

por potenciómetros normales, ya que ISIS no tiene la opción de obtener esta señal de lo microservos virtuales.

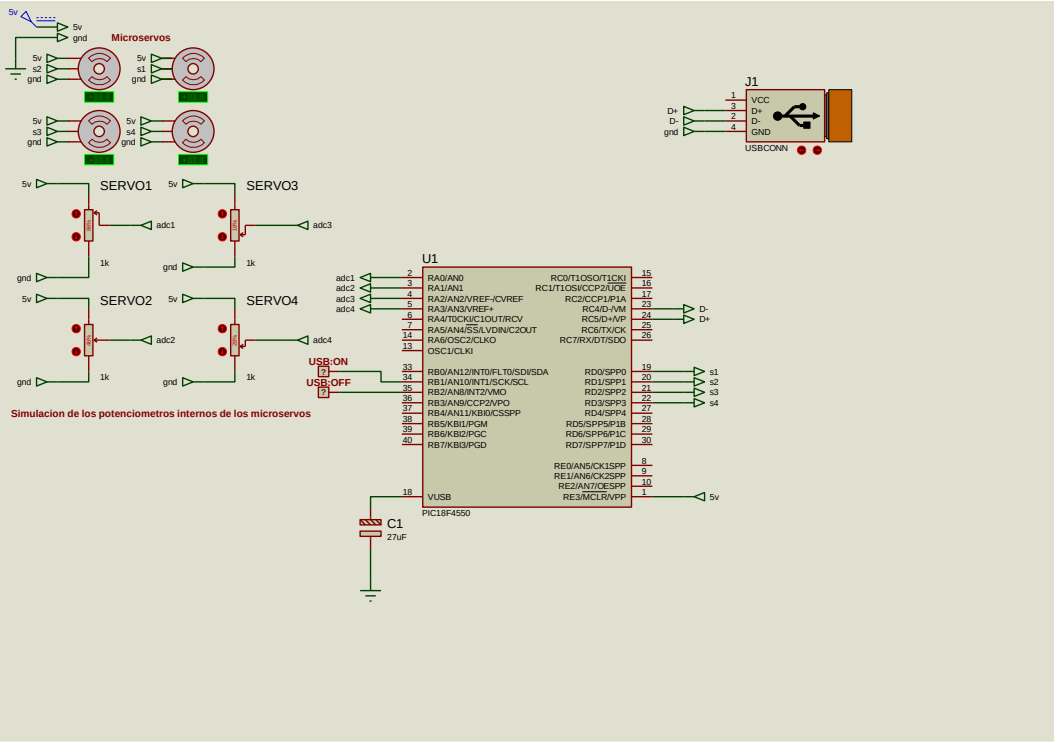

Figura 4.7. Simulación de la tarjeta de control mediante ISIS.

Luego de configurar el Microcontrolador virtual, se procede a instalar el driver del USB VIRTUAL de PROTEUS, el cual permite comunicar este PIC virtual con la Interfaz de Usuario (Figura 4.8).

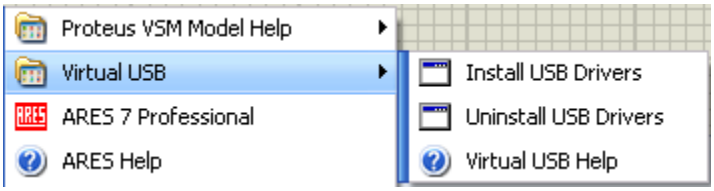

Figura 4.8. Instalación del driver virtual USB.

Teniendo el puerto USB (virtual) instalado, se procede a la simulación con la Interfaz de Usuario. Efectivamente, mediante la simulación, la interfaz de usuario responde al microcontrolador virtual (Figura 4.9). En el ítem 3.3.1 se indica la

confirmación visual del estado del USB, en la Figura 4.10, se tiene dos estados de dos pines del PIC, el de la izquierda indica que no hay comunicación, y en la derecha se muestra que la comunicación USB se ha establecido.

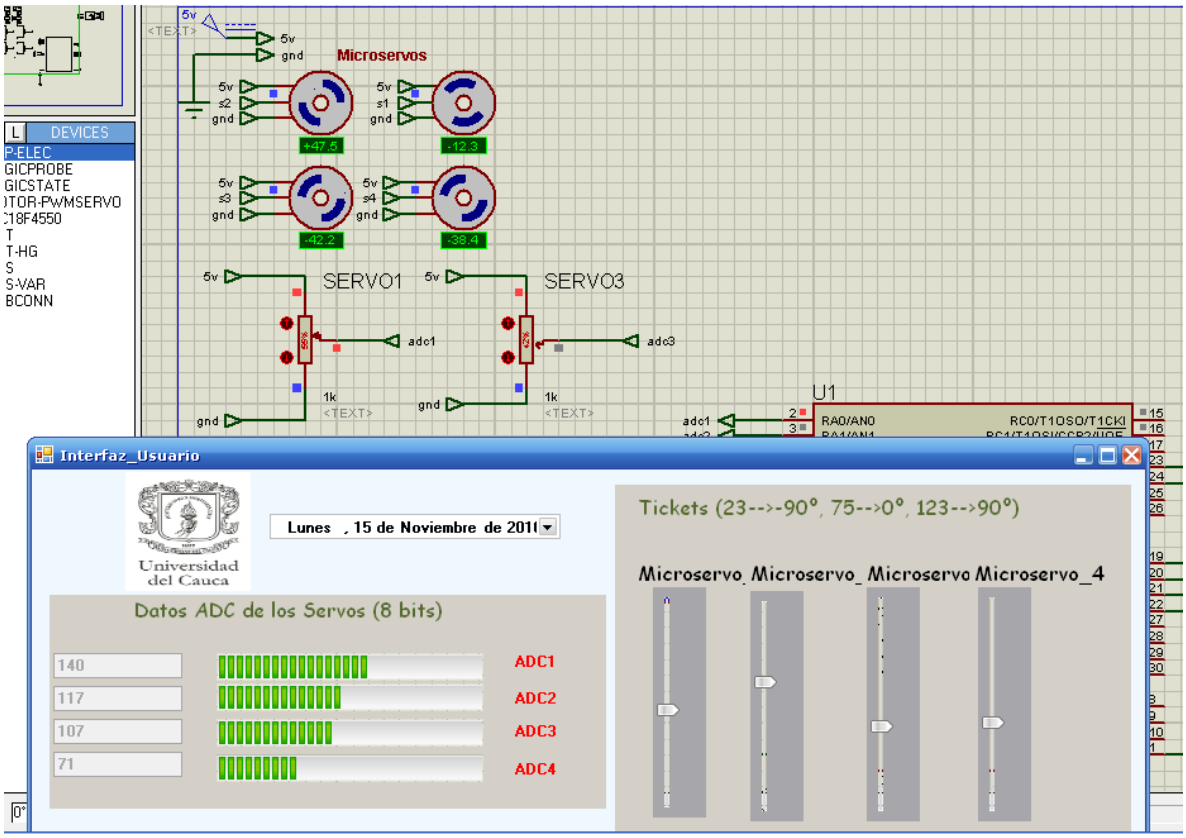

Figura 4.9. Simulación de la Interfaz de usuario con el PIC virtual.

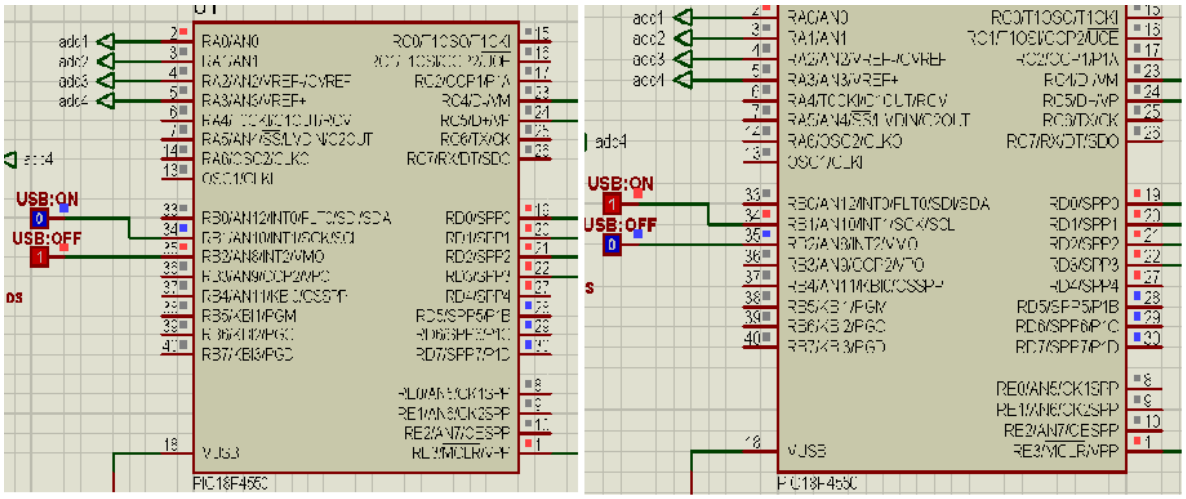

Figura 4.10. Estado de la comunicación USB: Izquierda (OFF), derecha (ON).

4.3 CONSTRUCCION DE LA INTERFAZ 3D

Para esta etapa, se decidió utilizar Opengl vinculándolo a Visual C#, para dicha tarea se deben tener el paquete de librerías gratuitas Tao Framework, esta nos permite utilizar las funciones de diversas librerías, enfocadas principalmente a la programación de gráficos y de videojuegos, con la gran ventaja de que es multiplataforma, permitiéndonos escribir programas que funcionaran tanto en Windows, como en Linux.

Este paquete no es una recopilación de las librerías que la componen, sino que simplemente es una interfaz (conocida como *wrapper*) para poder utilizar directamente esas librerías, por lo que es preciso tener, además de las DLLs del *tao Framework*, los archivos de las librerías que debamos utilizar. Esto no genera ninguno problema ya que estas vienen incluidas con el paquete para Windows, o bien se pueden encontrar con la facilidad para el resto de programas.

El paquete, por ahora y en su versión 2.1 nos dispone de las siguientes librerías [45]:

- **OpenGL:** conocida librería grafica para 2D y 3D.
- **OpenAl:** gestiona audio en 3D.
- **FreeGLUT:** permite realizar algunas funciones complejas de OpenGL de forma simple.
- **GLFW:** librería que permite realizar funciones complejas con OpenGL y de manejar eventos y dispositivos de forma simple.
- **FreeType:** permite utilizar tipos de letra TrueType.
- **PhysFS:** permite manejar archivos comprimidos.
- **SDL:** completa librería para manejar gráficos, audio, eventos y dispositivos.
- **ODE:** integra física dinámica.
- **Lua:** permite insertar un motor para scripts en lenguaje Lua.
- **DevIL:** permite procesamiento de imágenes, aplicando filtros y transformaciones.

- **CG:** permite realizar efectos y transformaciones a los gráficos realizados con OpenGL.

- **FFmpeg:** librería que permite abrir multitud de formatos de audio y video.

4.3.1 Configuración de tao framework en visual c#

Luego de crear un proyecto (Anexo C) y de tener la configuración USB (link 4.2.2), se procede a agregar las librerías a utilizar para la creación de la Interfaz 3D (ver Figura 4.11.).

Estas librerías son:

- Tao.Freeglut.dll

- Tao.OpenGl.dll

- Tao.Platform.Windows

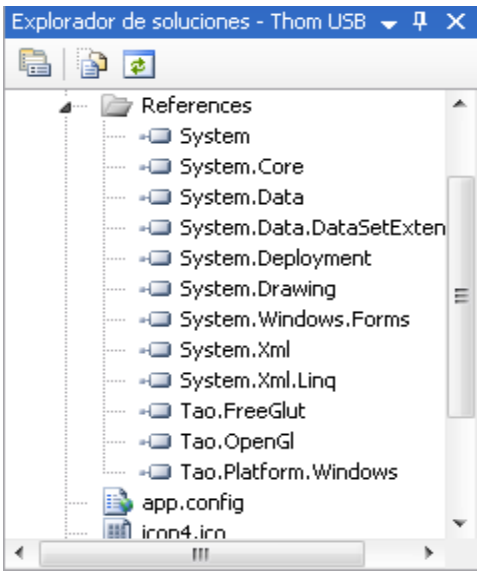

Figura 4.11. Inserción de librerías en Visual C#.

Ahora se debe copiar el archivo adjunto a estas librerías (freeglut.dll) y pegar en la ruta (…\WINDOWS\system32).

4.3.2 Creación de un Mundo Tridimensional

Para la creación del la Interfaz 3D, se debe realizar el código en el archivo program.cs, ya que este es la clase objeto de inicio de Windows Form de Visual C#.

En este archivo se incluyen las clases correspondientes a las librerías anteriormente vinculadas, como se muestra a continuación:

```
using Tao.OpenGl;
using Tao.FreeGlut;
using Tao.Platform.Windows;
```

Ahora se crea la ventana, con su respectivo titulo y dimensiones.

```
Glut.glutInit();
Glut.glutInitDisplayMode(Glut.GLUT_DOUBLE | Glut.GLUT_RGB);
Glut.glutInitWindowPosition(100, 100);
Glut.glutInitWindowSize(600, 600);
Glut.glutCreateWindow("Interfaz 3D");
```

En la interfaz 3D, se requiere recrear eventos de retroalimentación de colisión hacia la HAPTIC 3R, por tanto se realizó un cubo y una esfera. La esfera se podrá mover libremente dentro del cubo (movida por la HAPTIC 3R), en caso de que la esfera choque con alguna pared del cubo, esta debe enviar la posición actual al microcontrolador, y este a su vez debe interpretar esta posición con sus respetivos tickets, que una vez encontrados, estos envían la orden de detención a los microservos hasta que la colisión de la esfera desaparezca.

4.3.3 Creación del Cubo

Para crear el cubo, debe tener en cuenta, las dimensiones, origen, color (por caras), transparencia (para poder ver la esfera actuando dentro de él) y rotación/traslación mediante mouse o teclado.

Se debe implementar una función que dibuje las 6 caras del cubo, en cada cara se debe especificar el color `(Gl.glColor4f(R,G,B,A);)` dando valores (int (0-225), float(0.0f-1.0f)) a R,G,B (colores primarios) y A, (alpha, especifica el nivel de transparencia de cada cara, se debe especificar la figura a dibujar `(Gl.glBegin(Gl.GL_QUADS);)` en este caso un cuadrado y por último las coordenadas de cada vértice de la cara `(Gl.glVertex3f(X,Y,Z))`.
A continuación se muestra un ejemplo de la construcción de una cara:

```
Gl.glColor4f(0.0f, 1.0f, 0.0f, 0.5f);
Gl.glBegin(Gl.GL_QUADS);
Gl.glVertex3f(2.0f, -2.0f, -2.0f);
Gl.glVertex3f(-2.0f, -2.0f, -2.0f);
Gl.glVertex3f(-2.0f, 2.0f, -2.0f);
Gl.glVertex3f(2.0f, 2.0f, -2.0f);
Gl.glEnd();
```

Ya obtenidos los registros de construcción del cubo, estos se debe guardar en una matriz (`Gl.glLoadIdentity();`) para la construcción en la ventana creada. Luego se debe especificar el centro de origen del cubo `(Gl.glTranslated(x, y, z);).` por último invocar la función que construye las caras del cubo, en este caso `(drawcube();).`

Para trasladar el cubo de posición, se debe cambiar los valores de x, y, z `(Gl.glTranslated(x, y, z);),` para dicho caso, se utilizo los eventos del mouse para realizar translación en el eje X e Y como se muestra en la Figura 4.12. En la primera imagen esta el cubo en su posición inicial, la segunda imagen es la del cubo con translación en el eje X y la tercera imagen con translación en el eje Y.

```
static void mouse(int btn, int state, int x, int y)
        {
            if (state == Glut.GLUT_DOWN)
            {
                if (btn == Glut.GLUT_LEFT_BUTTON)
                    angulocuboX += 2.0f;
                else if (btn == Glut.GLUT_RIGHT_BUTTON)
                    angulocuboY += 2.0f;

            }


           }
```

Figura 4.12. Translación del cubo en el eje X e Y.

4.3.4 Creación de la Esfera

Para este caso, se debe llamar a la función (`Glut.glutSolidSphere(radius , slices , stacks);`) que recibe como parámetros:

- **radius**: radio de la esfera
- **slices**: número de subdivisiones en torno al eje Z (similar a las líneas de longitud.
- **stacks**: número de subdivisiones a lo largo del eje Z (similar a las líneas de latitud).

Anterior a esta función, se le debe definir el color, posición inicial y por último llamar a la matriz para que guarde sus valores de construcción.

```
Gl.glTranslated(x, y, z);
Gl.glColor4f(color, 0.0f, 1.0f, 0.8f);//color de la esfera
Glut.glutSolidSphere(0.3f, 8, 16);
Gl.glLoadIdentity();
```

En `(Gl.glTranslated(angulox, anguloy, anguloz);)` se modifican los valores x, y, z para trasladar la esfera en cualquier posición (ejes X, Y, Z) (Figura 4.13).

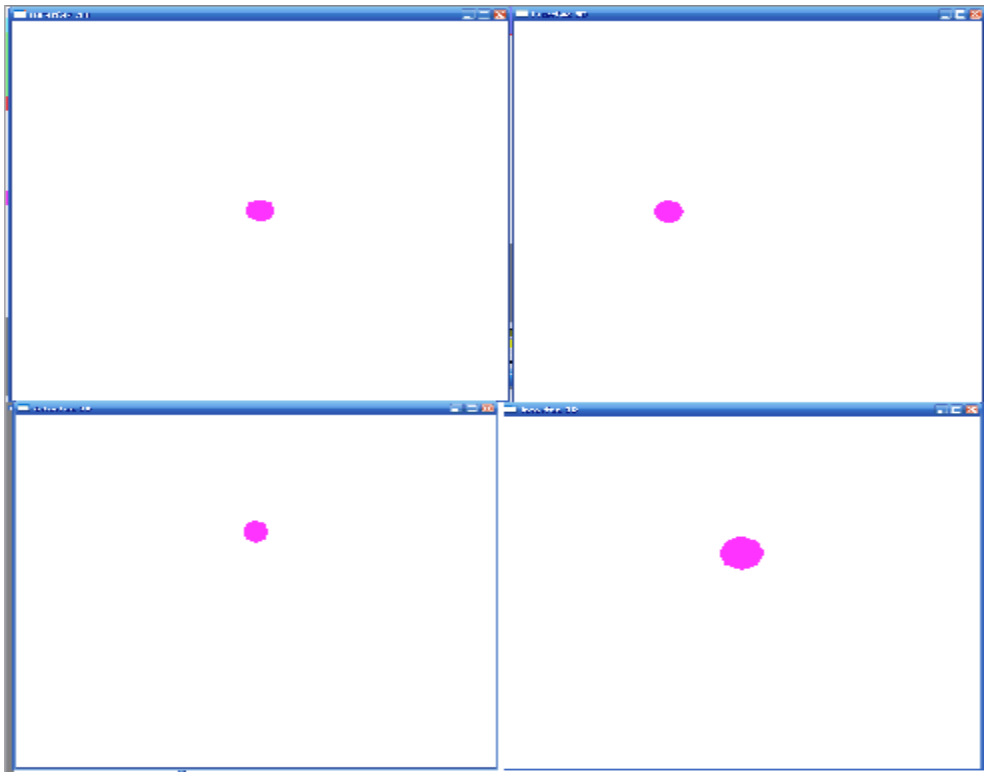

Figura 4.13. Translación de la esfera en los ejes X, Y, Z.

Para integrar estas dos figuras, se debe considerar que OPENGL crea figuras con solo la matriz (`Gl.glLoadIdentity`();), esto involucra que no se pueden crear dos figuras con las misma matriz porque puede ocurrir que salga la primera figura llamada, la segunda figura o ninguna, para solucionar este inconveniente, se debe refrescar esta matriz cada vez que se requiera para graficar, de esta manera se garantiza que las dos o más figuras salgan en la misma ventana como se indica a continuación (ver Figura 4.14):

```
//REFRESCA MATRIZ PARA ESFERA
Gl.glLoadIdentity();
Gl.glTranslated(angulox, anguloy, anguloz);
Gl.glColor4f(color, 0.0f, 1.0f, 0.8f);//color de la esfera
Glut.glutSolidSphere(0.3f, 8, 16);//Crea esfera
// REFRESCA MATRIZ PARA CUBO
Gl.glLoadIdentity();
Gl.glTranslated(0.0f, 0.0f, -5.0f);
Gl.glRotatef(angulocuboX, 1.0f, 0.0f, 0.0f);
Gl.glRotatef(angulocuboY, 0.0f, 1.0f, 0.0f);
drawcube();// crea cubo
```

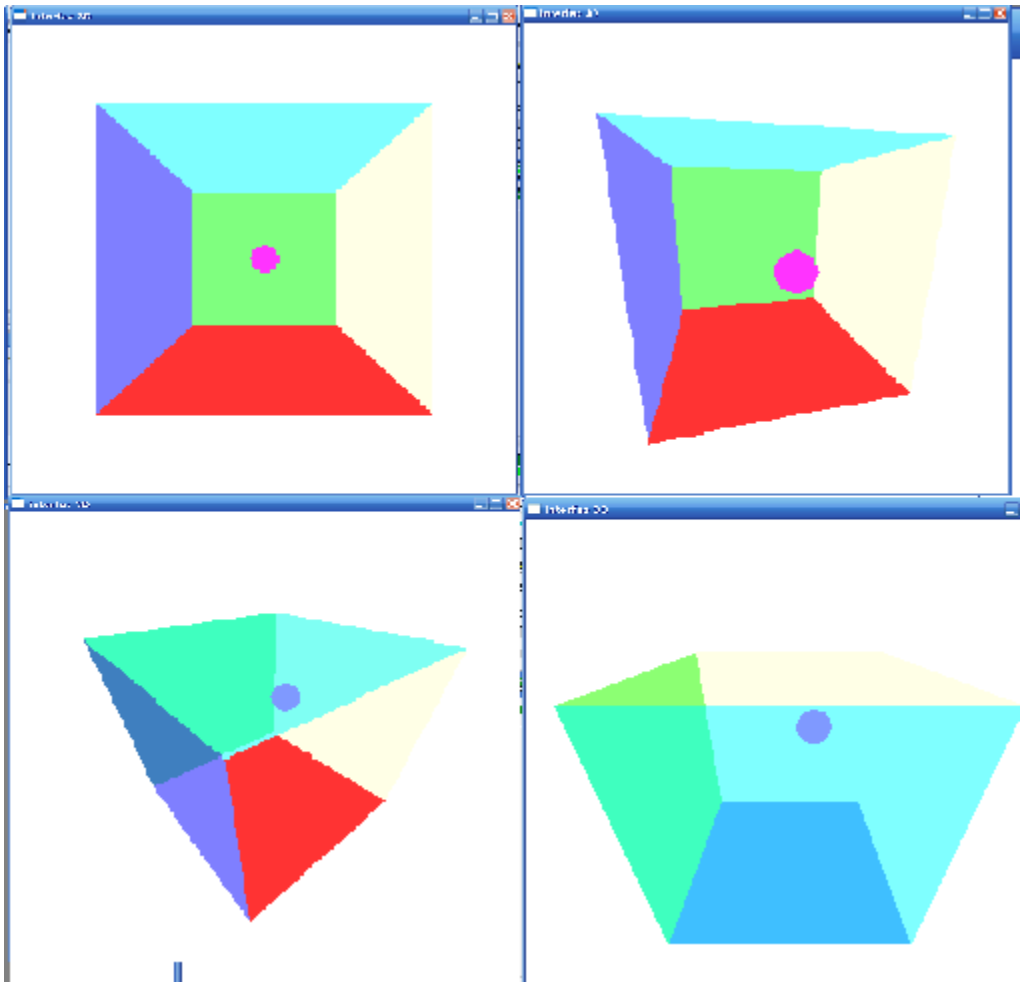

Figura 4.14. Interfaz 3D.

4.3.5 Translación y rotación de un objeto 3d mediante eventos usb

Como se planteó en el ítem 4.2.2, el archivo PicUSBAPI.cs tiene la configuración para la comunicación USB, pero esta se construyó bajo funciones públicas, y la Interfaz 3D se desarrolló bajo funciones *staticas*, en Visual c# no se permite llamar funciones *públicas* desde funciones *staticas*, por tal motivo se debe crear una instancia de dicha clase, para crear instancias de objetos e invocar métodos públicos se realiza lo siguiente.

```
PicUSBAPI usbapi = new PicUSBAPI();
```

Con esto ya se puede acceder al archivo PicUSBAPI y enlazar Opengl con el microcontrolador

La esfera tiene como posición inicial (Gl.glTranslated(x,y,z);) donde los valores de x, y, z definen la posición deseada por el usuario de la esfera en el plano cartesiano. Luego la forma mover la esfera mediante eventos USB es la siguiente:

```
lectura = new uint[4];
lectura = usbapi.LeeADC();
```

En donde *lectura* es un vector de tamaño 1x4 que almacena los valores entregados por el MGI, luego de almacenar datos, se entregan a las coordenadas x, y, z de la esfera:

```
x= lectura[0];

y=lectura[1];

z= lectura [2];
```

y `lectura[4]` que corresponde al valor de rotación.

4.4 DETECCION DE COLISIONES

En una aplicación de realidad virtual que utiliza una interfaz háptica, para dar un nivel de realismo superior en la interacción con los objetos de la escena, debe tenerse un adecuado manejo de la geometría, cuya representación puede ser en dos o tres dimensiones, a través de una punto, una línea, o en general un conjunto de primitivas [30].

4.4.1 Algoritmos para la Detección de Colisiones

El objetivo principal de estos algoritmos, es calcular las interacciones geométricas entre los objetos, sin importar el número ni la complejidad que pueda tener. Se presentan dos técnicas de detección de colisiones [31]:

- **Técnica AABB: Axis Aligned Bounding Box**. Si se suponen 2 objetos encerrados dentro de dos cajas alineadas como en la Figura 4.15, entonces la detección de colisión se realiza comparando las posiciones de los ejes coordenados con respecto al centro de cada objeto, así en una sola dimensión por ejemplo [31]:

  - Si la posición X-máxima de A es menor que la posición X-mínima de B entonces no hay colisión.

  - Si la posición de X-máxima de B es menor que la posición X-mínima de A entonces no hay colisión.

- Se repite el proceso para cada eje.

- Si ninguna condición es verdadera, entonces se dará la colisión entre los dos objetos.

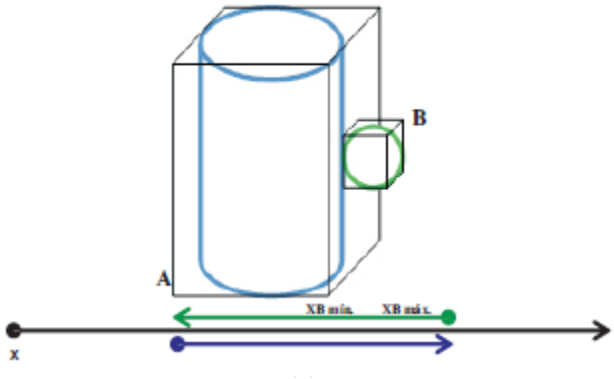

Figura 4.15. Objetos alineados para la detección de colisiones[47].

Una desventaja de la técnica AABB es que puede crear reportes falsos si los objetos encerrados son delgados y están inclinados, rotados o tienen alguna deformación (ver Figura 4.16).

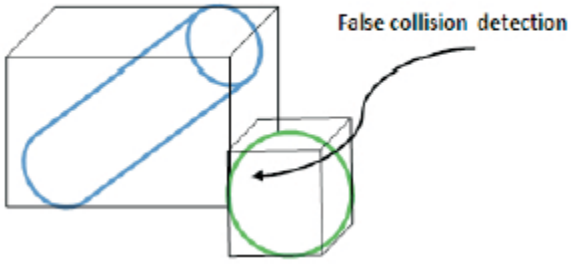

Figura 4.16. Falsa detección de colisión[47].

- **Técnica OBB: Oriented Boundig Box:** a través de esta técnica, se encierra un modelo geométrico a colisionar, dentro de una caja que está alineada con las dimensiones máximas del objeto (ver Figura 4.17), de esta forma se calcula la existencia de colisiones con la caja, en lugar de realizar el cálculo con cada punto del objeto.

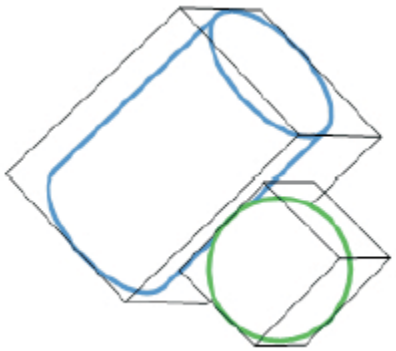

Figura 4.17. Técnica OBB Oriented Bounding Box[47].

La interfaz 3D utilizó la técnica AABB (técnica de colisión por distancias) porque en este caso la geometría de las figuras (cubo y esfera) es conocida o regular. Por tanto en la Figura 4.18 se muestra la detección de l colisión en la simulación. En la imagen de la izquierda, la esfera permanece en azul porque no hay detección de colisión, en cambio en la figura de la derecha, la esfera cambia de color al detectar colisión.

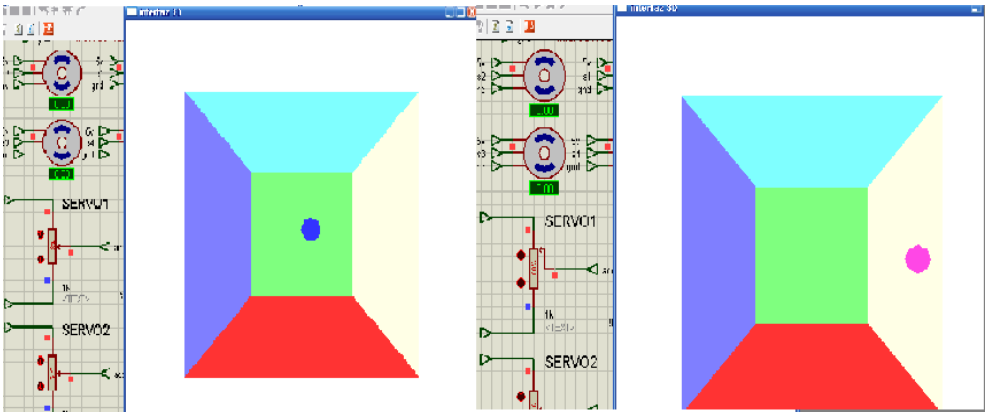

Figura 4.18. Detección de colisión mediante simulación.

Para interpretar la detección de colisión por el HAPTIC 3R se debe conocer el valor de la conversión ADC (a 8 bits), relacionarla con un tick y este creara la señal PWM que será enviada a cada microservo para obtener retroalimentación de

esfuerzo mecánico. En la tabla 4.1 se indica el valor del tick correspondiente a cada valor ADC obtenido.

Tabla 4.5. Interpretación de la lectura ADC para la detección de colisiones.

Valores ADC (8 bits)	Valor Tick	Posición Angular (grados)	Valores ADC (8 bits)	Valor Tick	Posición Angular (grados)
23	23	0	74	74	91.8
24	24	1.8	75	75	93.6
25	25	3.6	76	76	95.4
26	26	5.4	77	77	97.2
27	27	7.2	78	78	99
28	28	9	79	79	100.8
29	29	10.8	80	80	102.6
30	30	12.6	81	81	104.4
31	31	14.4	82	82	106.2
32	32	16.2	83	83	108
33	33	18	84	84	109.8
34	34	19.8	85	85	111.6
35	35	21.6	86	86	113.4
36	36	23.4	87	87	115.2
37	37	25.2	88	88	117
38	38	27	89	89	118.8
39	39	28.8	90	90	120.6
40	40	30.6	91	91	122.4
41	41	32.4	92	92	124.2
42	42	34.2	93	93	126
43	43	36	94	94	127.8
44	44	37.8	95	95	129.6
45	45	39.6	96	96	131.4
46	46	41.4	97	97	133.2
47	47	43.2	98	98	135
48	48	45	99	99	136.8
49	49	46.8	100	100	138.6
50	50	48.6	101	101	140.4
51	51	50.4	102	102	142.2
52	52	52.2	103	103	144

53	53	54	104	104	145.8
54	54	55.8	105	105	147.6
55	55	57.6	106	106	149.4
56	56	59.4	107	107	151.2
57	57	61.2	108	108	153
58	58	63	109	109	154.8
59	59	64.8	110	110	156.6
60	60	66.6	111	111	158.4
61	61	68.4	112	112	160.2
62	62	70.2	113	113	162
63	63	72	114	114	163.8
64	64	73.8	115	115	165.6
65	65	75.6	116	116	167.4
66	66	77.4	117	117	169.2
67	67	79.2	118	118	171
68	68	81	119	119	172.8
69	69	82.8	120	120	174.6
70	70	84.6	121	121	176.4
71	71	86.4	122	122	178.2
72	72	88.2	123	123	180
73	73	90			

4.5 INTEGRACIÓN DEL HAPTIC 3R CON LA INTERFAZ DE USUARIO E INTERFAZ 3D

Los resultados finales de la integración son las siguientes.

El HAPTIC 3R se construyó bajo los parámetros planteados anteriormente, como sus medidas, material, tarjeta de control.

Se conectó la interfaz en sistemas operativos, tales como Windows Xp (32/64b), Windows Vista (32/64b) y Windows 7 (64b), teniendo éxito en la instalación de drivers, pero se presento dificultad en el momento de abrir los ejecutables Interfaz_Usuario.exe y Interfaz_3D.exe en Windows 7 (32/64b), es este sistema operativo se recomienda utilizar la versión VISUAL STUDIO 2010.

La conectar el HAPTIC 3R con la Interfaz_Usuario e Interfaz_3D, se obtuvo respuesta inmediata en la comunicación USB (buenos tiempos de recepción/envió de datos).

Por ser un primer prototipo, se recomienda no forzar el HAPTIC 3R en sus límites de trabajo (espacio de trabajo) y en el momento de detección de colisión, ya que los microservos cuentan con piñones de material plástico.

Para el obtener los valores de Modelo Geométrico Inverso (MGI), los datos entregados por el ADC se escalizaron en un rango entre (-1) y 1.

5. CONCLUSIONES

Se diseñó una estructura robótica angular o antropomórfica basada en algunas interfaces hápticas como la Phantom [17] y Virtuose 6D35-45 [25], y se comprobó que esta estructura permite al robot moverse de manera adecuada dentro de un espacio de trabajo destinado para realizar tareas propias de un robot tipo interfaz háptico. La estructura física diseñada consta de 4 grados de libertad, de los cuales tres son para posición y un cuarto para orientación.

Para calcular el modelo geométrico directo (MGD), se calcularon las matrices de transformación en base de la tabla de parámetros geométricos del HAPTIC 3R. Con estas matrices, se cálculo el modelo geométrico inverso (MGI). Para esto se utilizo el método de Paul.

También se calcularon los modelos dinámicos (MDD y MDI), usando la aplicación de computador SYMORO®.

Se diseñó las piezas del HAPTIC 3R en el software CAD SolidEdge®, además esta herramienta nos estregó los parámetros dinámicos, inercias y masas del HAPTIC 3R.

Se comprobó el modelo geométrico inverso (MGI) mediante una simulación realizada en MATLAB/Simulink® con trayectorias cartesianas predefinidas, como hacer un circulo con un radio y centro determinado. Además se introdujo un esquema de control por par calculado (CTC).

Se implementó una tarjeta de control capaz controlar microservos, leer continuamente valores entregados por los potenciómetros internos de los servos, hacer la conversión ADC, implementar comunicación USB y recibir la posición (detección de colisión) para detener mecánicamente los microservos.

Para visualizar el comportamiento del microcontrolador frente a la comunicación USB con el computador, se implementaron simulaciones en PROTEUS (ISIS), y se comprobó el buen comportamiento frente a tiempos de envió/recepción.

Se desarrolló una interfaz de usuario en Visual C#, enlazada con la librería mpusbapi.dll que es la librería que instala el microcontrolador como dispositivo de Windows. Esta interfaz entrega información tal como los valores ADC, enviar una posición cualquiera hacia los microservos, y leer los datos entregados por el modelo geométrico inverso (MGI).

Se implementó una interfaz 3D para verificar el comportamiento del HAPTIC 3R en un espacio tridimensional utilizando la librería gratuita OpenGL vinculada mediante el paquete Tao Framework incluida en Visual C#, se verifico el comportamiento de la esfera frente a la posición del HAPTIC 3R y se comprobó la detección de colisión y la retroalimentación de fuerza.

REFERENCIAS BIBLIOGRAFICAS

[1] P. Jaime. *Diseño Háptico de los Dispositivos de Trabajo y Accidentabilidad*. Magister de Ergonomía de la Universidad de Concepción, 2008.

[2] J. Gutiérrez. *Ergonomía Háptica, Boletín de Factores Humanos.* www.tid.es/documentos/boletin/numero15_5 [Consultado Enero 2010]

[3] J.J Gibson. *The Senses Considered as Perceptual Systems*. Boston: Houghton Mifflin, 1966.

[4] S. Ballesteros. *Percepción Háptica de Objetos y Patrones Realzados: Una Revisión*. Departamento de Psicología Básica, Universidad Nacional de Educación a Distancia, 1993.

[5] G. de la Torre. *The Importance of the Sence of Touch in Virtual and Real Environments*. International Society for Haptics, Mexico, 2006.

[6] NOVINT. *NovintGame.* http://home.novint.com [Consultado Julio, 2009]

[7] SCHLUMBERGER. *Realidad Virtual en la Industria del Gas y del Petroleo.* http://199.6.131.12/es/scictr/watch/vr/haptics.htm [Consultado Febrero, 2009]

[8] ANALISIS MADRI+D. *Telecomunicaciones en Telematica y Tele Tutoria: la Importancia de Captar y Transmitir la Sensacion del "Tacto" Mediante Redes.* http://www.madrimasd.org/informacionIdi/analisis/analisis/analisis.asp?id=38702&tipo=g [Consultado Febrero, 2009]

[9] SABERSINFIN. *La Robótica Pedagógica, una Experiencia de la Enseñanza-Aprendizaje Basada en Proyectos*.

http://www.sabersinfin.com/index.php?option=com_content&task=view&id=1098&Itemid=89
[Consultado Febrero, 2009]

[10] C. Smith. *Factores Humanos en Interfaces Hápticas, 1999.* http://www1.acm.org/crossroads/espanol/xrds3-3/haptic.html
[Consultado Mayo, 2009]

[11] F. Monasterio Y H. Maciá. *Dispositivos Hápticos y Cirugía Robótica* http://robolabo.etsit.upm.es/haptico/
[Consultado Mayo, 2009]

[12] G. Burdea and P. Coiffet. *Virtual Reality Technology, The Ultimate Display, in Proceedings of IFIPS Congress*, New York city, 1994.

[13] M. Ishii and S. Makoto. *A 3D Interface Device with Force Feedback: A Virtual Work a Space for Pick-and-Place Tasks.* IEEE Annual Virtual Reality International Symposium, pp. 331-335, 1993.

[14] NEOTEO. *Haptics Tocando lo Virtual.* http://www.neoteo.com/haptics-tocando-lo-virtual.neo
[Consultado Mayo, 2009]

[15] NEOTEO. *NovintFalcon en Juegos de Estrategia.* http://www.neoteo.com/DesktopModules/FBlogArticles/SiteMapArticles.aspx?rs=0&re=14999
[Consultado Mayo, 2009]

[16] *SINCRONA SPRING 2008. Aportaciones del Arte Háptico: Sociología de lo Invisible.* http://sincronia.cucsh.udg.mx/esparzaspring08.htm#_ft
[Consultado Mayo, 2009]

[17] SENSABLE TECHNOLOGIES. *Products&Services.* http://www.sensable.com/haptic-phantom-desktop.htm
[Consultado Mayo, 2009]

[18] NOVINT. *What is 3D Touch?* http://home.novint.com/whatis3dtouch/what_is_3d_touch.php

[Consultado Mayo, 2009]

[19] INMERSION. *Engine Inmersion.*
http://www.immersion.com/index.phpresearch/research.html
[Consultado Mayo, 2009]

[20] VIRTALIS*. Moog, FCS-Haptic MASTER*.
http://www.virtalis.com/content/view/197/682/
[Consultado Mayo, 2009]

[21] FREEDOM. *Force Feedback Hand Controller*.
http://www.mpbtechnologies.ca/mpbt/haptics/hand_controllers/freedom/desription.html
[Consultado Abril, 2009]

[22] IMAN BROUWER*. Haptic Hardware.*
http://www.bracina.com/haptichardware.html
[Consultado Abril, 2009]

[23] VRLOGIC. *CyberTouch.*
http://www.vrlogic.com/html/immersion/cybertouch.html
[Consultado Mayo, 2010]

[24] HAPTION. *Virtuose Deskopt 6D.*
http://www.haption.com/site/eng/html/materiel.php?item=0
[Consultado Mayo, 2010]

[25] HAPTION. *Virtuose 6D35-45.*
http://www.haption.com/site/eng/html/materiel.php?item=1
[Consultado Mayo, 2010]

[26] HAPTION. Virtuose 6D40-40.
http://www.haption.com/site/eng/html/materiel.php?item=3
[Consultado Mayo, 2010]

[27] ENACTIVE. *EnactiveInterfaces.*
http://www.bracina.com/haptichardware.html
[Consultado Mayo, 2009]

[28] TENDENCIAS SOCIALES. *La realidad Virtual ya Permite el Aprendizaje en Tres Dimensiones.* http://www.tendencias21.net/La-Realidad-Virtual-ya-permite-el-aprendizaje-en-tres-dimensiones_a2047.html [Consultado Marzo, 2009]

[29] HAPTEX. *Information Society Technologies.* http://haptex.miralab.unige.ch/ [Consultado Junio, 2009]

[30] E. Meza y J. Ramirez. *Planeación Neuroquirúrgica y Navegación Estereostática*. Tesis de la Universidad Nacional de Colombia, 2009.

[31] M. Pinto. *Análisis e Implementación de una Interfaz Háptica en Entornos Virtuales*. Tesis de la Universidad Nacional de Colombia, 2009.

[32] W. Khalil y E. Dombre, *Modeling, Identification and Control of Robots.* London: Kogan Page Science, 2002. [E-Book]. Disponible: google-books.

[33] R. Paul, *Robot Manipulators: Mathematics, Programming and Control.* Cambrigde: MIT Press, 1982.

[34] W. Khalil y D. Creusot, *"Symoro+: A System for the Symbolic Modelling of Robots"*, *Robotica*, vol. 15, pp. 153-161, 1997.

[35] L. Sciavicco, y B. Siciliano, *Modeling and control of robot manipulators 6th ed*. Londres: Springer, 2005. [E-Book]. Disponible: google-books.

[36] C. Smith, *"Fundamentals of Control Theory"*, *Chemical Engineering,* vol. 86, pp. 11-39. 1979.

[37] G. ALMEIDA. Unidad 1. *Fundamentos Generales de la Robótica.* Universidad Técnica de Ambato, 2009.

[38] D. MARULANDA. *Aplicación Multicuerpo para Robot Angular o Antropomórfico*. MATLAB®CENTRAL, 2008.

[39] USB. Que es USB?
http://www.monografias.com/trabajos11/usbmem/usbmem.shtml#QUEES
[Consultado Julio, 2010]

[40] PICMANIA. *Controlando servos con una sola interrupción.*
http://picmania.garcia-cuervo.net/picc_servos_x_8.php
[Consultado Enero, 2010]

[41] PICMANIA. *Timer Calculator.*
http://picmania.garcia-cuervo.net/recursos.php
[Consultado Enero, 2009]

[42] WIKIPEDIA. *Microsoft Visual Studio Express.*
http://es.wikipedia.org/wiki/Visual_studio_express
[Consultado Julio, 2010]

[43] WIKIPEDIA. *Opengl.*
http://es.wikipedia.org/wiki/OpenGL
[Consultado julio, 2010]

[44] MICROSOFT. *Inicio de Visual C#.*
http://msdn.microsoft.com/es-co/vcsharp/dd919145.aspx
[Consultado Julio, 2010]

[45] dCases.com. *Tao Framework.*
http://www.dcases.com/?p=176
[Consultado Julio, 2010]

[46] M. LIN. *Collision Detection*. Listen-up web site. 2007.

[47] Pendiente1. *Haptic Feedback Model for Simulation of Cylindrical Cutting Tools Interacting with Arbitrary 3d Objects. PhD thesis of Master of Engineering in Information and Communications Technologies*. Asian Institute of Technology. School of Engineering and Technology, Thailand, 2008.