DISPOSITIVO DE MANDO AUXILIAR PARA EL ROBOT PORTAENDOSCOPIO HIBOU

Cristian Alejandro Tosse Robles
Emilio Alejandro Lasso Casas

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento De Electrónica, Instrumentación y Control
Línea de Investigación: Control y Robótica
Ingeniería en Automática Industrial

Popayán, Marzo de 2012

DISPOSITIVO DE MANDO AUXILIAR PARA EL ROBOT PORTAENDOSCOPIO HIBOU

Cristian Alejandro Tosse Robles
Emilio Alejandro Lasso Casas

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:
Phd. Oscar Andrés Vivas

Popayán, Marzo de 2012

Hoja de Aprobación

_____

_____

_____

Director _____
Phd. Oscar Andrés Vivas

Jurado _____

Jurado _____

Fecha de sustentación: Popayán, Marzo 22 de 2012

A Dios y todas las personas que con su entera comprensión, paciencia y dedicación, forjaran en mí la fortaleza y tenacidad para alcanzar mis metas. A mis padres que con su incondicional apoyo y amor permitieron culminar con éxito mis estudios de pregrado, gracias a mi hermana por su entrega y ayuda en los momentos de tristeza y alegría, a mi novia y amigos por su inmenso descernimiento y a mi compañero de tesis Emilio, por su valiosa amistad e inmensa colaboración para la consecución de este proyecto.

Cristian.

A Dios por estar siempre presente en cada etapa de mi vida. A mis padres por ser la fuente de mi energía que me ilumino para enfrontar las dificultades que día a día se me presentaron y que al recordarlas me llena de orgullo saber que gracias a sus enseñanzas, las pude sobrepasar, a mis compañeros por su amistad y alegría con la que me acompañaron en estos años de estudio y A Cristian por ser mi compañero en la obtención de este gran logro, y a todas las personas que de una u otra manera me ayudaron a llegar hasta lograr este objetivo.

Emilio.

Agradecimientos

A nuestro director de tesis Andrés Vivas que nos brindo su colaboración en el ámbito intelectual y personal, para la realización de este proyecto.
A los profesores que contribuyeron con sus enseñanzas y experiencias, aportadas en el transcurso nuestros estudios.
A nuestros compañeros de estudio por sus consejos, apoyo y su invaluable amistad.
A los evaluadores de este proyecto que con sus críticas constructivas fomentan el crecimiento y la mejora continua.
A los miembros del Departamento de Electrónica, Instrumentación y Control por su entereza y responsabilidad en el cumplimiento de sus labores.
A la Universidad del Cauca por su contribución administrativa y económica.

Resumen

El proyecto “Dispositivo de mando auxiliar para el robot porta endoscopio Hibou” es elaborado en la Universidad del Cauca (Colombia), por el grupo de investigación en Automática Industrial, representado por los autores y el director del mismo. En la actualidad se han desarrollado diferentes robots quirúrgicos, para mejorar los procedimientos médicos, de tal forma que estos sean más seguros y confiables, siendo una herramienta bastante útil para los cirujanos. En particular en la cirugía mini-invasiva (laparoscopia), se han construido robots como el Aesop, Endoassist, Lapman, LER, etc, los cuales presentan dispositivos de mando y control para el posicionamiento y orientación del endoscopio, con ventajas y desventajas particulares. Una de sus principales desventajas es que este tipo de tecnologías son muy costosas y sofisticadas lo cual dificultad la adquisición de estos equipos.

En el caso del robot Hibou desarrollado en la Universidad del Cauca, no se cuenta con un dispositivo externo que controle la posición y orientación de la cámara sin la ayuda de las manos, debido a que en un procedimiento laparoscópico el cirujano las utiliza para la manipulación de los instrumentos quirúrgicos. Por esta razón en este proyecto se construyó un hardware que permite manipular el órgano terminal del robot porta endoscopio Hibou virtual, mediante los movimientos de la cabeza del usuario. Este dispositivo basa su funcionamiento en la utilización de acelerómetros, además de una tarjeta de adquisición de datos desarrollada mediante un microcontrolador.

Para verificar que el cirujano pueda posicionar y orientar la cámara del robot Hibou virtual, se implementó un software utilizando un programa de representación gráfica por computador llamado Ogre3D, en donde se ambienta el quirófano con el paciente y el robot, además de que posee dos ventanas, las cuales presentan una visión del quirófano en general y la otra permite ver al interior del abdomen del paciente. De forma paralela, otro grupo trabajó sobre un proyecto complementario donde se implementa la parte matemática del robot Hibou, así como algunos elementos de la ventana virtual general.

Palabras Clave: robots quirúrgicos, posicionamiento y orientación del endoscopio, acelerómetro.

Abstract

The project "Auxiliary control device for the robot endoscope holder Hibou" is developed at the University of Cauca (Colombia) by the research group Industrial Automation, At present different surgical robots has been developed to improve medical procedures, so that there are more secure and reliable, being a very useful tool for surgeons. Particularly in minimally invasive surgery (laparoscopy) have built robots as, Aesop, Endoassist, Lapman, LER, etc. which show command and control devices for positioning and orientation of the endoscope, with advantages and disadvantages. One of its main disadvantages is that these technologies are very expensive and sophisticated which difficulty acquiring this equipment.

In the case of Hibou robot developed at the University of Cauca, there is not an external device to control the position and orientation of the camera without the use of hands, because in a laparoscopy procedure the surgeon use it for the manipulation of the surgical instruments. For this reason this project was built hardware that allows the manipulation of the Hibou's robot terminal organ. The device operation is based on the use accelerometer sensor, and a data acquisition card developed by a microcontroller.

To verify that the surgeon can position and orient the Hibou's robot virtual camera, was implemented using a software program for computer graphic called Ogre3D, where is moodeled the operating room with the patient and the robot also has two windows that, which present a vision of operation room in general and other allow see inside the abdomen of the patient. in parallel, another group worked on a complementary project which implements the mathematical part of the robot Hibou, and some elements of the general virtual window.

Keywords: surgical robots, position and orientation of endoscope, acelerometer.

Contenido

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Abreviaturas

FDA:	(*Food and Drug Administration*) Agencia de Drogas y Alimentos.
USB:	(*Universal Serial Bus*) Bus Universal en Serie.
OGRE:	(*Object-Oriented Graphics Rendering Engine*) Librería Orientada a Objetos de Representación Gráfica.
3D:	Tridimensional.
API:	(*Application Programming Interface*) Interfaz de Programación de Aplicaciones.
CEGUI:	(*Crazy Eddie's GUI System)* Sistema para crear GUI con Ogre.
(CMI):	Cirugía Mini-Invasiva.
CO_2:	Dióxido de Carbono.
LER:	(*Light Endoscopio Robot*) Robot Portaendoscopio de luz.
RAM:	(*Ramdom Acces Memory*) Memoria de Acceso Aleatorio.
OpenGL:	(*Open Graphics Library*) Librería Gráfica de Libre Uso.
MGD:	Modelo Geométrico Directo.
MGI:	Modelo Geométrico Inverso.
CAD:	(*Computer Aided Desing*) Diseño Asistido por Computadora.
CCD:	(*Charge Coupled Device*) Dispositivo de Carga Acoplada.

Introducción

La historia de la robótica antigua y moderna es importante conocerla para saber cómo funcionan los nuevos robots. La fabricación de máquinas que imitan al ser humano en sus tareas o labores se ha mantenido desde hace más de 4000 años. En la robótica clásica se destacaron inventores como Arquitas de Tarento, Heron de Alejandría, Hsieh-Fec, Al-Jazari, Bacon, Turriano, Leonardo da Vinci, Vaucanson o Von Kempelen, entre otros [1].

Con el desarrollo de la mecánica, la electrónica y la informática en el siglo XX, se logró desarrollar robots capaces de realizar de forma autónoma tareas de gran complejidad.

Por otra parte la cirugía laparoscópica, es el resultado de los esfuerzos repetidos, durante muchas generaciones, de cirujanos visionarios que querían ofrecer curación quirúrgica de los padecimientos, sin causar el daño involuntario que se ocasiona al abrir la pared abdominal. Los registros históricos de maniobras laparoscópicas datan desde el siglo X, con Abulcasis, quién realizó una inspección endoscópica del cuello uterino. En 1805 Bozzini en Frankfurt implementó el primer endoscopio con una cánula de doble luz, una vela y un espejo reflejante para observar cálculos y tumores de la vejiga, pero fue hasta 1960 que el Dr Kurt Semm desarrolló un aparato de insuflación con regulación de presión y flujo de gas, con lo cual impulsó a las nuevas generaciones para seguir evolucionando y convertir a este procedimiento quirúrgico en una cirugía de alta aceptación en el mundo [2].

En la cirugía mini-invasiva, los instrumentos quirúrgicos y el sistema de visión son introducidos a través de pequeñas incisiones, esto implica que el paciente se recupera más rápido después de la cirugía, con menos traumas y menores cicatrices respecto a una cirugía convencional [3]. Sin embargo, tiene algunas dificultades tales como: el control de la cámara de video, la dificultad ergonómica en el procedimiento quirúrgico, el entrenamiento del cirujano, la pérdida de sensación táctil sobre el paciente, la precisión, la pérdida de visión tridimensional que se tiene respecto a las cirugías abiertas y la falta de rotación de la articulación de la muñeca del médico [4].

Por esta razón a finales del siglo XX se llega a la utilización de robots, siendo el PUMA 560 el primero que fue utilizado en cirugía, en este caso para tomar una biopsia del cerebro. Fue a partir de esta invención que se desarrollaron otros robots quirúrgicos, Robodoc, Gaspar o Acrobot, Zeus, Aesop, etc [5].

Con la llegada del año 2000 la FDA (*Food and Drug Administration*) aprueba el Da Vinci Surgical System (*Intuitive Surgical Inc, Sunnyvale, CA, USA*), el cual es un sofisticado robot asistente del cirujano, utilizado para procedimientos urológicos como la prostatectomía, cistectomía o la nefrectomía [1].

Cabe resaltar que las principales investigaciones, desarrollos y aplicaciones sobre robótica quirúrgica se han realizado en países como Estados Unidos, Japón, Francia, Alemania y Europa en general, y que para países en desarrollo como Colombia son tecnologías muy difíciles de adquirir y aplicar, principalmente por sus altos costos, ya que el precio comercial de estos robots puede oscilar entre los $200.000 y $1’000.000 de dólares.

Debido a esto en Latinoamérica se han desarrollado diferentes proyectos como el Tonatiuh [6], creado por el Laboratorio de Bioelectrónica del Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional en México, Kirubot [7], diseñado por el Grupo de Investigación en Bioingeniería de la Universidad Pontificia Bolivariana de Colombia que es controlado desde LabView, y un manipulador desarrollado por el Grupo de Automática y Robótica de la Pontificia Universidad Javeriana de Cali para cirugías de columna vertebral [8]; hechos que demuestran el esfuerzo de los investigadores por incursionar en el campo de la robótica quirúrgica y estar a la vanguardia mundial.

En la Universidad del Cauca se han diseñado robots quirúrgicos como el Lapbot y el robot Hibou. Éste último robot es un robot porta endoscopio que se ha elaborado a partir de estructuras cinemáticas conocidas, construido en un ambiente virtual bajo Matlab®, y tomando como señal de entrada la proveniente de un joystick con conexión por puerto USB, para la orientación y posicionamiento del órgano terminal, método inusual en los robots actuales.

Es por ello que con éste proyecto se pretende realizar un dispositivo de mando externo para el robot porta endoscopio Hibou Virtual, que oriente y posicione la cámara mediante movimientos de la cabeza, utilizando tecnologías que no han sido exploradas anteriormente en la robótica quirúrgica como la utilización de acelerómetros. Para esto se requiere diseñar e implementar un dispositivo

electrónico tridimensional externo al PC, construido a partir de sensores de posición, que le permita al cirujano el control del robot, además de un software que simule el movimiento de una imagen captada por una cámara virtual dentro del abdomen del paciente, y que ésta sea dirigida por medio del módulo electrónico que mueve el endoscopio virtual. Conjuntamente se aspira a desarrollar un proyecto Macro que incorpore los modelos matemáticos que rigen al robot, implementados sobre un software libre, que permita la ejecución de éste en tiempo real pudiéndose integrar con el primer prototipo del robot Hibou que se realizará en el futuro.

Este proyecto fue desarrollado en paralelo con otro complementario, llamado "Software manipulador del robot porta endoscopio Hibou para cirugía laparoscópica", en el cual se implementaron los modelos matemáticos del robot, el sistema de control y algunos elementos de la interfaz virtual general tales como la camilla y el quirófano. En la Figura I1 se muestra los componentes de cada uno de los proyectos así como la interacción existente entre ellos. Los bloques enmarcados en rojo corresponden a este proyecto, mientras que los verdes corresponden al proyecto complementario.

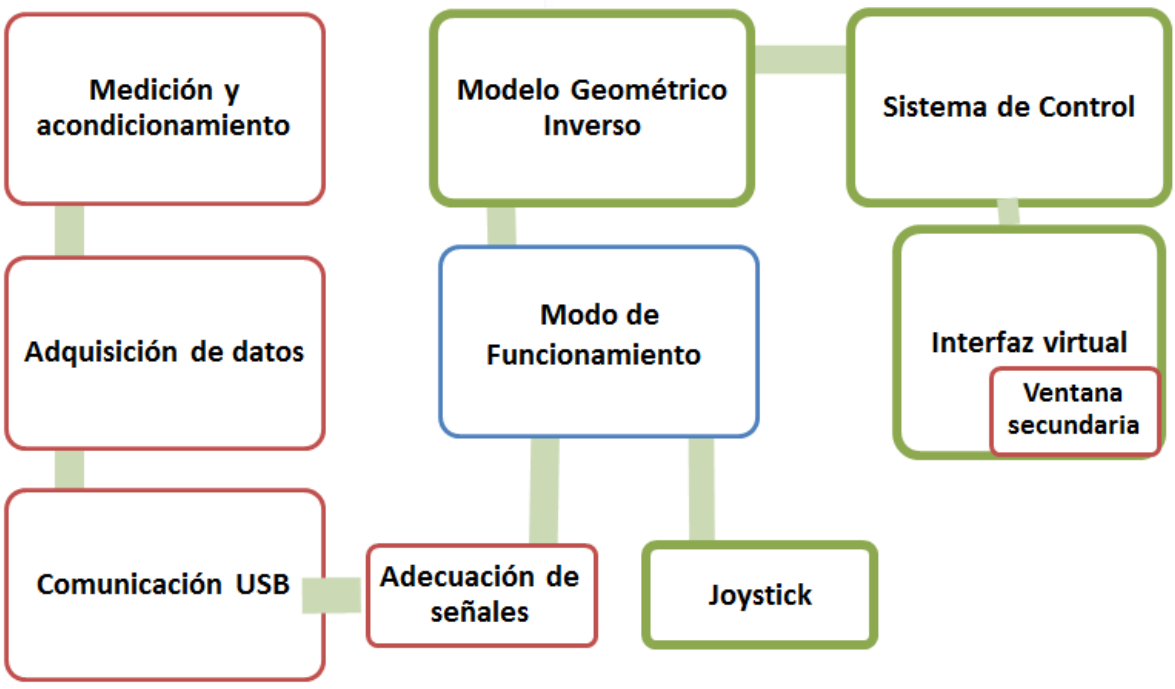


Figura I1. Esquema general de los proyectos involucrados.

El primer capítulo presenta conceptos básicos sobre robótica médica, cirugía laparoscópica, robótica quirúrgica, robots porta endoscopios y robots comerciales para cirugía de laparoscopia**,** estos temas pretenden dar un conocimiento general, que es primordial para el entendimiento de los siguientes capítulos. En el segundo capítulo se investigan los métodos para posicionar el endoscopio, sensores para el posicionamiento del porta endoscopio y la selección del acelerómetro para la manipulación del órgano terminal del robot, además se describe como se elabora el hardware básico de prueba. El tercer capítulo muestra el motor gráfico utilizado para el proyecto, diferentes software utilizados para el ambiente virtual y el código implementado en Ogre 3D. El cuarto capítulo describe el diseño del dispositivo electrónico y en el quinto capitulo quedan consignadas las conclusiones del proyecto.

1. Robótica empleada en procedimientos quirúrgicos

1.1. Robótica médica

En la actualidad la robótica es una gran ayuda para la cirugía, debido a que es una herramienta que facilita el desarrollo de cirugías de alta complejidad en donde el ser humano no puede acceder con facilidad. En los últimos años la robótica ha generado diferentes equipos como simuladores quirúrgicos que sirven para el entrenamiento de estudiantes o profesionales, y de robots quirúrgicos, que permiten efectuar procedimientos de una forma más segura y eficiente.

Estos aportes tecnológicos contribuyen a que en el área de la salud se puedan realizar operaciones delicadas, las cuales pueden disminuir la abertura en la zona de operación a solo unos pocos milímetros, y por esta abertura realizar la operación.

La cirugía mini-invasiva (CMI) específicamente la laparoscopia, es el procedimiento que se ha escogido para trabajar en este proyecto ya que es una de las más desarrolladas a nivel mundial y nacional [1] y [1][9].

1.2. Cirugía laparoscópica

Es una técnica quirúrgica que se practica a través de pequeñas incisiones, usando la asistencia de una cámara de video que permite al equipo médico ver el campo quirúrgico dentro del paciente y accionar en el mismo. Se llama a estas técnicas mínimamente-invasivas, ya que evitan los grandes cortes de bisturí requeridos por la cirugía abierta o convencional y posibilitan, por lo tanto, un periodo post-operatorio mucho más rápido y confortable.

La cirugía se realiza gracias a una video-cámara que se introduce en el cuerpo a través de una incisión, esta cámara de pequeño tamaño cuenta con una fuente de luz fría que ilumina el campo quirúrgico dentro del abdomen.

En la laparoscopia es necesario separar la pared abdominal de los órganos, elevando el abdomen para que el cirujano tenga acceso al interior del paciente y pueda manipular los instrumentos con relativa facilidad. Actualmente se emplea CO_2 para este fin [10].

Los equipos e instrumentos utilizados para realizar la cirugía son:

- Herramientas largas llamadas trocares, por las cuales se introducen en el cuerpo del paciente los instrumentos quirúrgicos. Estos pueden ser de 5 y 10 mm de diámetro.
- Instrumentos quirúrgicos como pinzas, tijeras, cauterio, láser, engrapadoras, ligaduras, etc.
- Cámara de video, básica para obtener la imagen con la que trabajan los cirujanos.
- Monitor de alta resolución conectado al laparoscopio que proyecta la imagen obtenida por la cámara.
- Endoscopio.
- Equipo de anestesia automatizada.
- Aparato automático de insuflación que introduce aire al cuerpo para facilitar la separación de los órganos del paciente.
- Aguja de Veress [11].

Los instrumentos anteriormente mencionados se pueden observar en la Figura 1.1.

Figura 1.1 Instrumentos y equipos para la cirugía laparoscópica [11]

La aguja de Veress es el instrumento que se utiliza para puncionar la cavidad abdominal e insuflar el abdomen. Después de obtener acceso al abdomen se realizan incisiones de 5 a 10 mm, para introducir los trocares que son instrumentos con un punzón cortante. Primero se practica una pequeña incisión generalmente de 10 mm de diámetro, cercana al ombligo, por la cual se introduce un trocar que permite el ingreso de un sistema de iluminación con fuente de luz fría, y una cámara para visualización y grabación de imágenes [12]. Es importante destacar que la laparoscopia, antes de ser una técnica quirúrgica, es un método de exploración, que consta de un componente visual que permite magnificar (de 10 a 20 veces) la cavidad abdominal, y un componente táctil que no es una característica inmediata, pero que permite identificar el campo de trabajo [12] y [13].

En la actualidad, los procedimientos quirúrgicos laparoscópicos más comunes son:

- **Colecistectomía:** intervención que se realiza para quitar una vesícula biliar enferma. Cosiste en extirpar por completo la vesícula biliar a través de 3 incisiones de 5 a 10mm por donde se introducen finos instrumentos y una cámara que lleva la imagen a un monitor de alta resolución [14].

- **Funduplicatura:** La cirugía está indicada en algunos casos de hernia hiatal y enfermedad por reflujo gastroesofágico. Consiste en restituir la presión de la válvula llamada “esfínter esofágico inferior” mediante un pliegue del estomago sobre el esófago a manera de envoltura [8].

- **Apendicectomía:** Consiste en extirpar el apéndice cecal. Por vía laparoscópica se realiza generalmente a través de 2 a 3 incisiones de 5 a 10mm bajo anestesia general, y puede realizarse tanto en apendicitis no complicadas como en muchos de los casos complicados [15].

- **Colectomía asistida:** Extirpación o resección de una parte enferma del intestino grueso (colon). La resección del intestino delgado se lleva a cabo mientras el paciente se encuentra bajo anestesia general. Se realiza una incisión en el abdomen, se extirpa la parte enferma del intestino grueso (colon) y finalmente se suturan los dos extremos sanos y se cierra la incisión [10].

Las principales ventajas de la cirugía laparoscópica son: menor dolor, recuperación precoz, estancia hospitalaria e incapacidad cortas, morbimortalidad

baja, resultado estético excelente, ayuda diagnóstica, recuperación temprana de la actividad intestinal, menor formación de adherencias y de hernias incisionales, menor trauma postoperatorio, etcétera. Las desventajas, además de la morbilidad propia del acceso a la cavidad abdominal, se derivan de la tecnología utilizada: pérdida de la sensación táctil y limitación en el control del sangrado.

1.3. Robótica quirúrgica

La robótica quirúrgica ha sido desarrolla desde hace algunas años, pero se dice que ésta fue inspirada en su mayor parte por proyectos como el robot controlador de James Watt construido en 1758, el brazo robótico desarrollado por el inventor estadounidense George Devol, entre otros. Sobre la década de los 80`s se implementó el brazo robótico PUMA, con fines industriales, pero debido a su excelente desempeño, en el año de 1985 se optó por utilizarlo por primera vez en una biopsia neuroquirúrgica. A partir de ahí se desarrollaron robots cirujanos como Gaspar o Acrobot. Estos avances se han creado por las constantes mejoras de procedimientos médicos complejos y del perfeccionamiento de las capacidades que el cirujano pueda proveer en el quirófano. Por esta razón cabe aclarar que la invención de robots destinados a la cirugía o diferentes prácticas médicas, no tienen la finalidad de excluir al profesional de la salud sino por el contrario se desea hacer de estas tecnologías un complemento ideal para él [2].

Es el caso de robots destinados a diferentes tipos de cirugías como Robodoc, Da Vinci, Zeus, Padyc, LER, Endoassist, etc. A continuación se hará una breve descripción de alguno de los robots más representativos en la actualidad.

1.3.1 Robodoc

Éste robot es desarrollado por *Integrated Surgical System Inc.* en California Estados Unidos, está compuesto por un robot SCARA de cinco grados de libertad, equipado con fresa de alta velocidad, un monitor de sala y el armario de control. Este sistema robotizado sirve para realizar operaciones como el vaciado de hueso en prótesis de cadera y rodilla. Sus principales ventajas son: precisión del 98%, no necesita cementación, minimiza micro-movimientos entre el implante y hueso, éste robot se puede ver en la Figura 1.2 [16].

Figura 1.2 Robodoc [16].

1.3.2 Da Vinci

El sistema quirúrgico Da Vinci es un sistema desarrollado por Intuitive Surgical, que consiste en un robot quirúrgico diseñado para posibilitar cirugías complejas con invasiones mínimas al cuerpo humano [17]. Este sistema está conformado por diferentes elementos entre los cuales se encuentra: la consola maestra, robot esclavo, instrumentos, interfaz gráfica de usuario y sistema de obtención de imagen.

Consola maestra y robot esclavo

Es la mesa de control en donde el cirujano ejecuta los movimientos que habrá de simular el robot esclavo, el cual está constituido por tres brazos. Uno de ellos contiene el manipulador para la cámara y los otros dos, los manipuladores de instrumentos que reproducen los movimientos de las manos del cirujano realizados desde la consola (ver Figura 1.3).

Figura 1.3 Consola maestra (Imagen izquierda) y robot esclavo Da Vinci (Imagen derecha) [18].

Instrumentos

Los instrumentos que utiliza el robot son: tijera, bisturí, diferentes tipos de pinzas, ganchos, disectores y porta-agujas; todos ellos están dotados de retroalimentación táctil electrónica que transmite las sensaciones de presión, resistencia, flexibilidad, etc. (ver Figura 1.4).

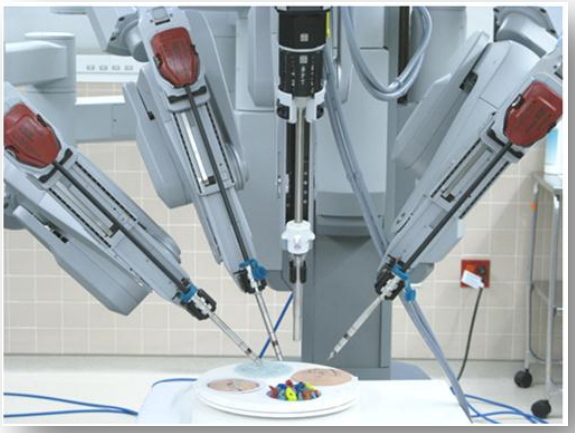


Figura 1.4 Instrumentos quirúrgicos Da Vinci [18].

Interfaz gráfica de usuario y sistema de obtención de imagen

Computadora con procesador Pentium de 200 MegaHertz y 64 megabytes de memoria RAM, y 20 procesadores Sharc en el controlador constituyen el sistema completo de este robot. La interfaz gráfica ayuda al cirujano a ampliar o disminuir sus movimientos en escalas de (1:1, 1:3, 1:5), reposicionando la cámara. El software implícito en esta interface garantiza la seguridad del paciente. Para la obtención de la Imagen el sistema contiene una cámara doble que le permite obtener dos señales de video, que al integrarse componen una señal de video estereoscópica, la cual se proyecta mediante dos monitores de alta definición, en un sistema llamado “caja de espejos” el cual crea una imagen en tres dimensiones [18].

1.3.3 Zeus

El Proyecto Zeus nace en Goleta, California en Estados Unidos, desarrollado por Computer Motion en 1997. Este es un sistema *master-slave*, que consiste en una unidad de control, dos brazos mecánicos y el robot Aesop, todo en conjunto forman el robot Zeus. Los modelos estándar poseen 2 grados de libertad, pero en la actualidad existen robots con 5 grados de libertad. El robot también contiene una computadora que se encuentra dentro de la consola del cirujano, la cual controla los tres brazos, ésta es la que lleva el rastro de la posición tridimensional de la punta de cada instrumento y de la cámara. En Zeus el cirujano observa la operación con un sistema de imágenes tridimensionales, el brazo robótico que controla cámara es accionado mediante comandos de voz por el cirujano (ver Figura 1.5) [5].

Figura 1.5 Robot Zeus [5].

1.4 Robots porta endoscopios

Los robots porta endoscopios tienen como tarea específica asistir al médico o cirujano cuando está practicando una cirugía por la técnica de laparoscopia, contribuyendo con la fijación y el control de los movimientos del laparoscopio.

La función en si del robot es mantener y posicionar el laparoscopio que se introduce en el abdomen del paciente a través de un orificio o trocar, con lo cual se busca explorar el interior del paciente mediante las imágenes grabadas por una cámara. El profesional de la salud guía al robot mediante instrucciones, para que éste se desplace y se ubique en la posición deseada.

En la actualidad existen diferentes opciones para esta aplicación y el tipo de control varía entre sistemas instruidos por comandos de voz, sensores infrarrojos ubicados en la cabeza del cirujano, pedales, etc.

La tarea de sostener y posicionar el laparoscopio originalmente le correspondía al asistente del cirujano, sin embargo se observó que por la duración prolongada de las cirugías, el asistente se cansaba y tendía a moverse mucho. Esto da como resultado una imagen distorsionada en el monitor y por consiguiente una deficiencia en el proceso operatorio. De aquí parte la justificación de la implementación de la robótica en este tipo de tareas; ahora el robot laparoscópico hará la tarea del asistente pero sin la preocupación del agotamiento y de perturbaciones por temblor o fatiga muscular [19].

1.5 Robots comerciales porta endoscopios

1.5.1 Aesop

Robot creado por Computer Motion Incorporated en Santa Bárbara, California, este sistema está compuesto por un brazo mecánico acoplado a un ordenador que reconoce comandos de voz específicos, que se captan a través de un micrófono. La computadora registra estas órdenes verbales y las transforma en movimientos que guían la cámara hacia la posición deseada por el cirujano.Lo particular del sistema es que solo obedece a las voces de los cirujanos registrados en la base de datos de la computadora, con lo cual se evita problemas de interferencia con otras voces, Aesop puede configurarse para tres diferentes posiciones, con lo

cual garantiza una gran versatilidad. Este robot cuenta con una serie de sistemas de seguridad que protegen al paciente. Al inicio de la cirugía, el cirujano fija un margen inferior límite para el brazo robótico de Aesop. Esto previene al brazo robótico de herir al paciente durante la operación, de la misma manera evita comandos inadvertidos por parte del cirujano al conducir el telescopio a través de estructuras anatómicas, como el hígado [5].

Con el uso de este robot se disminuye considerablemente el número de personas requeridas para efectuar una intervención quirúrgica, con ello se mejora considerablemente la estabilidad y calidad de la imagen debido a que por errores humanos estas pueden verse afectadas [20] y [21]. Este brazo robótico se observa en la Figura 1.6.

Figura 1.6 Aesop [20].

1.5.2 Endoassist

Sistema creado por Armstrong Healthcare Ltd. en Inglaterra. El Endoassist es un dispositivo independiente que sostiene el laparoscopio, controlado por el cirujano en lugar del asistente. Se activa mediante un pedal y es controlado por los movimientos de la cabeza del cirujano. Está diseñado para facilitar la cirugía laparoscópica mediante la mejora de la imagen sobre la que el cirujano trabaja.
Este robot permite fácilmente insertar el sistema de video dentro de la cavidad pélvica-abdominal; la naturaleza de su interfaz humano-computador permite lograr el dominio del robot en poco tiempo [22]. El Endoassist se muestra en la Figura 1.7.

Figura 1.7 Endoassist [22].

1.5.3 LER

Light Endoscopio Robot, robot creado por TIMC, Grenoble, Francia. Brazo mecánico articulado unido a un computador, el cual se encarga de interpretar órdenes verbales simples que el robot convierte en movimientos de la cámara laparoscópica. Este sistema consta de tres partes: una base móvil, un laparoscopio de flexión, y un manipulador externo. La particularidad de este robot es su reducido tamaño por tanto es muy útil, ya que se puede acoplar en cualquier quirófano sin necesitar de grandes espacios (ver Figura 1.8) [23].

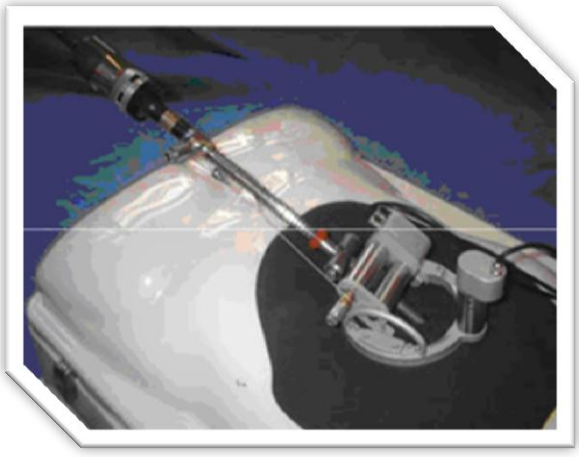


Figura 1.8 LER [23].

1.5.4 Lapman

Dieñado por *Medsys,* Bélgica. Es un robot que puede ser fácilmente trasladado desde y hacia la mesa de operaciones. El robot Lapman consta de una base móvil para un fácil manejo del sistema y un brazo que ejecuta los movimientos reales, mientras sostiene el endoscopio. Su eje tiene un laparoscopio que se mueve mediante órdenes a distancia, estas órdenes son enviadas a través del mando de control RF LapStick®. Posee un sistema que se ubica sobre el mango del instrumento quirúrgico, el cual es activado por el cirujano, el mecanismo asegura la estabilidad de la imagen, con lo cual el control de los movimientos de la cámara recae únicamente en el cirujano y libera al asistente de cirugía para ampliar sus funciones de apoyo [24]. Este robot se observa en la Figura 1.9.

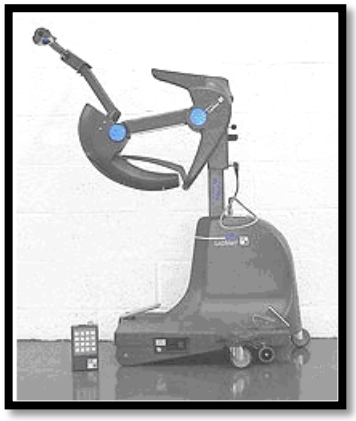


Figura 1.9 Lapman [24].

1.5.5 Tonatiuh

Diseñado en el Laboratorio de Bioelectrónica del Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, México**.** Es el brazo robótico Tonatiuh, el primero en su tipo en América Latina, diseñado por el doctor en ingeniería eléctrica Arturo Minor**.** Robot constituido por un brazo mecánico de cinco grados de libertad con tres articulaciones y un efector final. El grado de libertad rotatorio ubicado en la base se usa para la ubicación del manipulador sobre el puerto de inserción del laparoscopio y las articulaciones restantes se utilizan para la navegación. Este sistema puede ser operado por el cirujano mediante: movimientos cefálicos a través de luz infrarroja, reconocimiento de voz o por un control manual físico [6]. El Tonatiuh se aprecia en la Figura 1.10.

Figura 1.10 Brazo Tonatiuh [6].

1.5.6 Adelantos tecnológicos en Colombia sobre robots quirúrgicos

En nuestro país los avances en robótica son muy escasos, por lo que no existen robots quirúrgicos avalados para realizar procedimientos quirúrgicos, aunque se pueden destacar algunos proyectos de investigación y desarrollo de aplicaciones robóticas como:

- **Kirubot:** Prototipo diseñado y construido en la Universidad Pontificia Bolivariana de Medellín (ver Figura 1.11). El robot está conformado por un brazo robótico controlado por instrumentación virtual para la realización de procedimientos quirúrgicos. Este brazo robótico es controlado mediante una interfaz tridimensional que permite la ubicación de las coordenadas de un punto en el espacio [25].

Figura 1.11 Robot Kirubot [25].

1.5.7 Adelantos tecnológicos en la Universidad del Cauca sobre robots quirúrgicos

- **Lapbot:** Robot diseñado por el Grupo de Investigación en Automática Industrial de la Universidad del Cauca. Este robot creado para cirugía laparoscópica, fue desarrollado en un ambiente de simulación tridimensional que se creó bajo el software Ogre 3D. Ésta aplicación sirvió para probar el seguimiento de trayectorias de instrumentos quirúrgicos que se llevan a cabo en una operación de colecistectomía [26].En la Figura 1.12 se aprecia el robot Lapbot en un ambiente virtual.

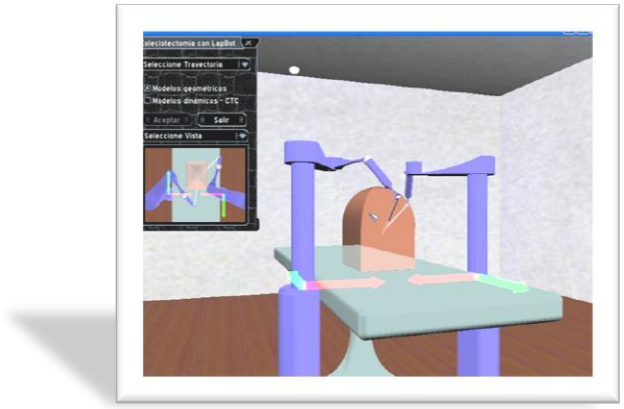


Figura 1.12 Entorno virtual de Lapbot [26].

- **Hibou:** Robot diseñado en la Universidad del Cauca por estudiantes de pregrado como proyecto de tesis, consiste en un porta endoscopio planteado a partir del estudio de las estructuras cinemáticas de robots existentes. En este análisis se lograron establecer las articulaciones que proveen el posicionamiento y las que suministran la orientación de la cámara, obteniendo una solución que plantea una estructura que pueda posicionar y orientar correctamente el efector final del robot en cualquier punto del espacio cartesiano. Para esto se hacen necesarias siete articulaciones, de las cuales cinco de ellas son motorizadas y dos son no motorizadas (articulaciones pasivas). Se decidió utilizar la técnica de articulaciones pasivas para garantizar el paso por el trocar u orificio abdominal. La estructura cinemática de este robot se puede observar en la Figura 1.13 [27].

En el Hibou simboliza el trocar o punto de incisión abdominal con un anillo entre las articulaciones 5 y 6, con lo cual las articulaciones 6 y 7 quedarían dentro del abdomen del paciente. Por otra parte, debido a la disposición física de las articulaciones 6 y 7, la distancia D7 se hace igual a cero.

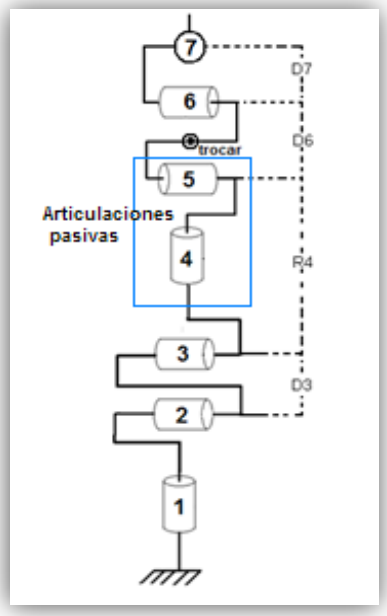


Figura 1.13 Estructura cinemática del robot Hibou [27].

Al terminar la estructura cinemática del robot se tuvieron en cuenta los siguientes aspectos:

- **Modelo geométrico directo del robot Hibou**

Una vez diseñada la estructura cinemática se procedió a obtener la tabla de parámetros geométricos, mediante el método de Khalil-Kleinfinger [28]. A partir de esta tabla se puede encontrar el Modelo Geométrico Directo (MGD) que permite calcular la posición y orientación del efector final del robot en el espacio cartesiano; así como el Modelo Geométrico Inverso (MGI), que proporciona los valores de las variables articulares que satisfacen una posición cartesiana deseada. El primer modelo desarrollado no brinda ningún inconveniente en

cálculo, solo basta con multiplicar las matrices de transformación de cada una de las articulaciones hasta obtener la matriz ${}^{0}\boldsymbol{T}_{7}$ [27].

- **Estudio del modelo geométrico inverso del robot porta endoscopio**

Para hallar el MGI del robot se utiliza como primera instancia el método de Paul [29], integrando para este método la restricción del paso por el trocar o punto de incisión, la cual radica en acotar los movimientos de posicionamiento de tal manera que el cuerpo del robot que se introduce dentro del paciente siempre tenga un punto de rotación fijo, que será el mismo punto por donde se hace la incisión en el abdomen (trocar).

La restricción del trocar se planteó de la siguiente manera:

Las articulaciones a las que afecta la restricción son exactamente las que se encuentran justo antes y después de la incisión en el abdomen del paciente. Para el robot Hibou son las articulaciones 5 y 6, y se representan con las posiciones cartesianas *P5*(*x5*, *y5*, *z5*) y *P6*(*x6*, *y6*, *z6*) respectivamente. Además el punto de incisión, el trocar, es identificado como *Pt* (*xt*, *yt*, *zt*) (ver Figura 1.14).

La restricción consiste en expresar el vector ($\overrightarrow{P5P6}$), como un producto cruz entre los vectores ($\overrightarrow{P5Pt}$) y ($\overrightarrow{PtP6}$.), es decir que el vector ($\overrightarrow{P5P6}$.) sea colineal con los vectores ($\overrightarrow{P5Pt}$) y ($\overrightarrow{PtP6}$) [26].

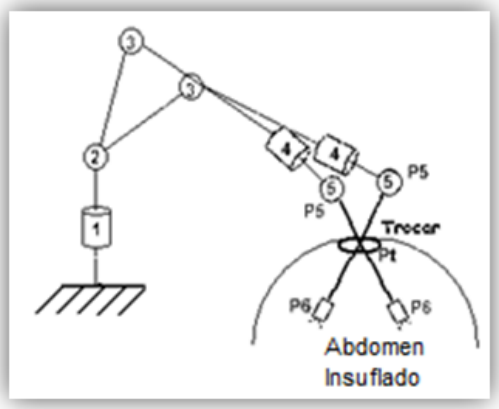


Figura1.14 Movimiento de las articulaciones alrededor del punto de incisión (trocar) [26].

Una vez planteadas las ecuaciones, se procede a calcular las soluciones para las variables articulares *θ1* a *θ7*, mediante una combinación del método de Paul y un método numérico [29]. Éste último se utilizó debido a la dificultad para encontrar las soluciones de ciertas variables articulares del robot. El método consiste en probar una serie de valores para cada variable a partir de unas condiciones iniciales. Cuando el resultado de reemplazar los valores de cada variable en las ecuaciones dadas para la solución del modelo es cercano a cero, (aproximadamente $1e^{-15}$), entonces el proceso se detiene y entrega los valores de cada variable para los cuales se cumplió el mínimo resultado posible.

Con esta información el grupo del proyecto macro encargado de la tesis “Software manipulador del robot porta endoscopio Hibou para cirugía laparoscópica”, realizó un algoritmo basado en el método de *Nelder – Mead* para ser utilizado sobre el lenguaje C++, que pretende resolver de manera numérica iterativa, las ecuaciones del modelo geométrico inverso. Este corresponde a un método heurístico de búsqueda de mínimos de cualquier función N-dimensional. Para ello a partir de un punto inicial estimativo, el método busca en el hiperespacio paramétrico aquellos valores de éstos que minimizan la función objetivo. Se basa en fundamentos geométricos; a partir de la estimación inicial y conocidos el número de parámetros a optimizar, *N*, el algoritmo construye el poliedro más sencillo en ese hiperespacio paramétrico: el poliedro posee *N*+1 vértices donde se evalúa la función objetivo y se decide que nuevos valores de los parámetros se ajustarán mejor al objetivo prefijado.

2. Investigación, diseño e implementación para el dispositivo electrónico de prueba

2.1. Métodos para posicionar el porta endoscopio

Para lograr posicionar el órgano terminal del robot Hibou se requiere de dispositivos que permitan transformar los movimientos del cirujano en órdenes de control sobre los motores que mueven las articulaciones, las cuales generan el movimiento del endoscopio asociado a la cámara del robot. Para esto es necesario investigar sobre la forma cómo los robots porta endoscopios sitúan el efector final en una ubicación en el espacio. Estos métodos se enuncian a continuación:

2.1.1 Posicionamiento por sensores infrarrojos

La utilización de sensores infrarrojos ha sido ampliamente utilizadas en la robótica, en la actualidad existen diferentes robots porta endoscopios como el Endoassist (Figura 2.1) y Tonatiuh que basan el funcionamiento de la posición de la cámara en láser ubicados sobre la cabeza del cirujano. Estos sensores son capaces de medir la radiación electromagnética infrarroja de las paredes del quirófano, con lo cual una tarjeta interpreta esta medición para convertir los movimientos de la cabeza del cirujano en mandos que el robot será capaz de interpretar con el fin de posicionar el órgano terminal [5].

Figura 2.1 Robot Endoassist- Control de la cámara por casco con infrarrojos [5].

2.1.2 Posicionamiento por comandos de voz

Los sistemas orientados por comandos de voz son muy usados por varios robots como el Aesop, LER y Lapman los cuales emplean órdenes verbales específicas, que son grabadas en una base de datos de una computadora a través de un micrófono. El sistema de reconocimiento de voz utilizado para orientar la cámara, realiza un reconocimiento de patrones de palabras empleadas por el cirujano, con el fin de evitar que el ruido emitido por personas que se encuentren en el quirófano, afecte el control de la cámara asociada al endoscopio. Esto permite que la cámara siga los requerimientos del profesional de la salud (ver Figura 2.2) [23].

Figura 2.2.Robot Aesop- Control de la cámara por comandos de voz [23].

2.1.3 Posicionamiento por sensores de ultrasonido

En la robótica se manejan mucho los sensores por ultrasonido debido a su tamaño y fácil implementación, aunque pueden ser muy poco precisos y poco confiables debido a las interferencias o ruido que puede existir en el entorno. Por estas razones en la actualidad no existen robots porta endoscopios que utilicen este tipo de dispositivos para la orientación del efector final.

2.2 Selección del sensor para el posicionamiento del porta endoscopio

Para la selección de un sensor óptimo para posicionar el endoscopio se indagó sobre los dispositivos que se utilizan actualmente en los endoscopios mencionados anteriormente, pero también se buscaron algunas opciones que difieran en las soluciones encontradas para la orientación de la cámara. Se encontraron diferentes sensores los cuales se explicarán a continuación.

2.2.1 Dispositivos para el posicionamiento del endoscopio

2.2.1.1 Sensores por ultrasonido

La mayoría de sensores por ultrasonido dedicados especialmente a la medición de distancias, basan su funcionamiento en la emisión de un pulso de ultrasonido cuyo lóbulo, o campo de acción es de forma cónica. Midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco se puede establecer la distancia a la que se encuentra el obstáculo que ha producido la reflexión de la onda sonora. Para entender mejor se muestra de una manera más clara en el siguiente esquema (Figura 2.3), donde se tiene un emisor que emite un pulso de ultrasonido que rebota sobre un determinado objeto y la reflexión de ese pulso es detectada por un receptor de ultrasonido [30].

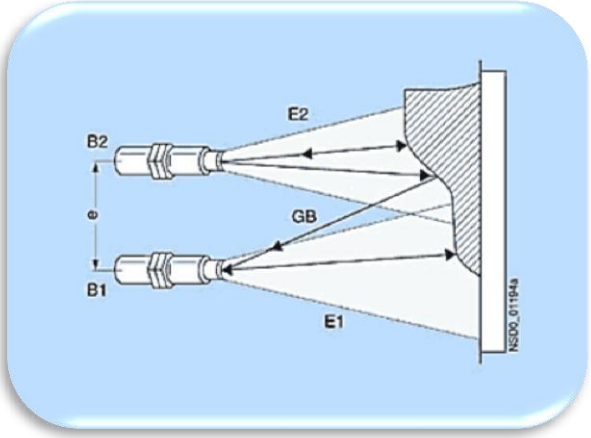


Figura 2.3 Sensor por ultrasonido [30].

2.2.1.2 Laser de triangulación

Los láseres de triangulación determinan la posición de un cuerpo por la medición reflejada en la superficie del objeto. Estos dispositivos constan de un laser, un detector (CCD) y un objetivo. Para realizar la medición de la distancia del objeto, el láser emite una luz en la superficie. Este rayo es reflejado por el área y es recibido por el detector a través de una lente. Dependiendo de la posición de la viga de la matriz de detectores o CCD, el ángulo (α) es calculado y por consiguiente la distancia desde el sensor hasta el plano es deducido. Este ángulo es inversamente proporcional a la distancia del objetivo, por tanto a medida que aumenta la distancia el ángulo disminuye y viceversa (ver Figura 2.4) [31].

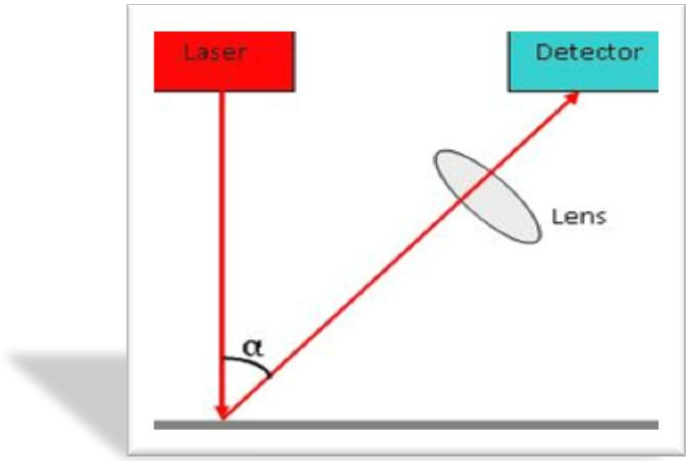


Figura 2.4 Laser de triangulación [31].

2.2.1.3 Laser de larga distancia

El laser envía un haz de luz con diferentes frecuencias y compara la señal que se ha reflejado en el objeto a medir, con el valor de referencia interno. Las características básicas son: resolución de 1mm, distancia hasta 30 m sin reflector y hasta 100m con reflector. Este dispositivo se puede observar en la Figura 2.5 [32].

Figura 2.5 Láser de larga distancia [32].

2.2.1.4 Acelerómetro

El funcionamiento básico del acelerómetro depende de un sistema de masas y resortes. Aunque en la actualidad se utiliza el principio de capacitancia para medir el desplazamiento de un sistema elástico que consiste en una barra de silicio sujetada por cuatro hilos. A esta configuración se adicionan tres placas metálicas que forman dos condensadores. La placa de la mitad cambia la distancia entre las placas de los extremos. Esto significa que en un acelerómetro (por ejemplo el ADXL50), los cambios en capacitancia están dados por la diferencia (o movimiento) entre las placas de los extremos y la placa del centro. El movimiento de la placa central depende de la barra de silicio anclada en los cuatro extremos, para finalmente entregar una variación de voltaje proporcional a los cambios de gravedad que experimenta (ver Figura 2.6) [33].

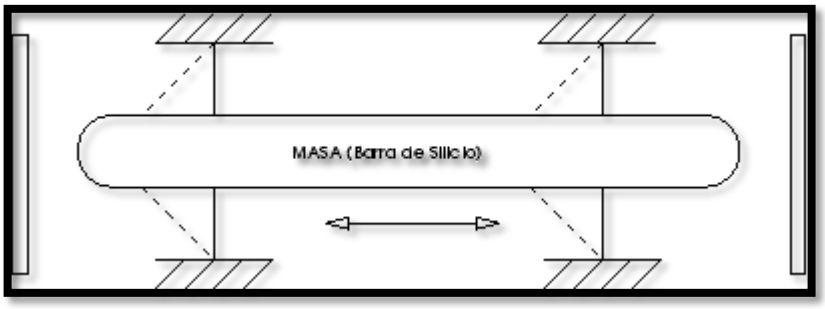


Figura 2.6 Modelo de un acelerómetro con capacitancia [33].

2.2.2 Alternativas de sensores para el control de posición

Después de investigar diferentes tecnologías para posicionar la cámara se procedió a proponer algunas alternativas de los métodos propuestos anteriormente. En la Tabla 2.1 se muestran los sensores con su respectiva etiqueta:

Tabla 2.1 Alternativas de sensores para orientación de la cámara

ALTERNATIVA	ETIQUETA
ADXL320 (acelerómetro 2 axis)	P1
ADXL345 (acelerómetro 3 axis)	P2
MMA7361L (acelerómetro 3 axis)	P3
EZ4 (Sensor ultrasónico)	P4
EZ1 (Sensor ultrasónico)	P5
SRF08 (Sensor ultrasónico)	P6
GP2Y0A02YK0F (sensor laser distancia 15-150cm)	P7
GP2D12 (Sensor laser detección 10-80cm)	P8
GP2Y0A21YK0F (sensor laser de distancia 10-80cm)	P9

Con las propuestas anteriores se establecieron diferentes criterios de selección con cada una de sus ponderaciones, con el fin de escoger la mejor alternativa. En la Tabla 2.2 se evidencia el criterio, la descripción de éste y su correspondiente prioridad.

Tabla 2.2 Criterios de selección y ponderación.

CRITERIO	DESCRPCION	PRIORIDAD
Precio	El costo total del dispositivo en el catalogo	1
Montaje	Facilidad de adecuación para instalar el dispositivo	2
Tamaño del sensor	Dimensión del sensor en instalación	3
Rango de medición	Rango en el cual el sensor opera	4
Flexibilidad al entorno	Adaptabilidad a diferentes entornos	5

Para seleccionar el sensor óptimo para el proyecto se evaluaron las posibilidades expuestas mediante la matriz de selección, en la cual se valoran cada uno de los criterios frente a las alternativas, dependiendo de las ponderaciones y prioridades colocadas a cada una de las disyuntivas. Con estas valoraciones obtenidas sobre las diferentes casillas, se procede a realizar la suma de cada una de ellas por columnas, para poder determinar el mayor valor y de esta manera encontrar la mejor opción. Esto se puede evidenciar en la Tabla 2.3

Tabla 2.3 Matriz de selección

		ALTERNATIVAS								
Criterios	**PRIO.**	P1	P2	P3	P4	P5	P6	P7	P8	P9
Precio	1	7 7	7 7	8 8	5 5	5 5	3 3	7 7	6 6	6 6
Montaje	2	5 10	5 10	5 10	6 12	6 12	6 12	7 14	7 14	7 14
Tamaño del sensor	3	9 27	9 27	10 30	7 21	7 21	5 15	3 9	3 9	3 9
Rango de medición	4	6 24	8 32	8 32	6 24	6 24	5 20	4 16	3 12	3 12
Precio Flexibilidad al entorno	5	10 50	10 50	10 50	5 25	5 25	5 25	6 30	6 30	6 30
Total		118	126	130	87	87	75	76	71	71

A partir de la matriz de selección se logró evidenciar que la alternativa etiquetada con P3, que corresponde al acelerómetro MMA7361L, es la mejor opción. Otro elemento importante que permitió inclinarse sobre algún método fue el hecho de que no existe porta endoscopios que basen el sistema de posición y orientación de la cámara en este tipo de sensores, lo cual podría convertirse en un aporte científico importante de este proyecto.

2.3 Diseño de la tarjeta para acondicionamiento de señales del acelerómetro

Para el diseño de la tarjeta de acondicionamiento de señales, se determinó dividirlo en diferentes ítems, dependiendo de las funciones que desempeñan. Estas se presentan a continuación:

2.3.1 Sensado

El acelerómetro escogido para determinar la posición y orientación del órgano terminal del robot es un MMA7361L de *Freescale Semiconductor* (Figura 2.7). Este sensor requiere muy poca energía y puede ser configurado para realizar mediciones entre ±1.5g y ±6g. Otras características que posee son modo *sleep*, acondicionamiento de señal, filtro pasabajas de 1 polo, compensación de temperatura, autoprueba, y detección de 0g para caída libre.

Figura 2.7 Sensor de aceleración MMA7361L [34].

2.3.1.1 Especificaciones

- Dos rangos de medición seleccionables (±1.5g, ±6g).
- Bajo consumo de corriente: 400 µA.
- *Sleep mode*: 3 µA.
- Bajo voltaje de operación: 2.2 V - 3.6 V.
- Alta sensibilidad (800 mV/g a 1.5g).
- Rápido tiempo de encendido (0.5 ms de tiempo de respuesta).
- Autoprueba para detección de caída libre.
- Acondicionamiento de señal con filtro pasobajo.
- Diseño robusto, alta resistencia al impacto.
- Dimensiones: 0.90 x 0.50".
- Costo: US $4,78.

2.3.1.2 Características especiales

El MMA7361L posee unas características particulares las cuales se describen a continuación:

a. ***0g-Detect*:** El sensor ofrece una función de detección de 0g, la cual proporciona una señal lógica alta cuando los tres ejes se encuentran en gravedad cero. Esta característica permite proteger el sensor cuando éste se encuentra en una caída libre.

b. ***Self Test*:** El sensor tiene una función de auto prueba, ésta le permite la verificación de la integridad mecánica y eléctrica del acelerómetro en cualquier momento, ya sea antes o después de la instalación. Esta función es crítica en aplicaciones tales como la protección del disco duro en donde se debe garantizar la integridad del sistema durante la vida útil del producto. Para utilizar esta función, el acelerómetro debe ser puesto boca abajo para que el eje Z se encuentre en -1g. Cuando la función de auto prueba es iniciada, una fuerza electrostática se aplica a cada eje, lo cual hace que cada uno de ellos se desvíe. Los ejes X e Y se desvían ligeramente mientras que el eje Z se recorta para desviar 1g. Este procedimiento asegura que tanto la mecánica y las secciones electrónicas del acelerómetro estén funcionando correctamente.

c. ***G-Select*:** La función de selección de gravedad permite elegir entre dos sensibilidades. Dependiendo de la entrada lógica que se alimente sobre el pin 10 (ver Figura 2.8), la ganancia interna del dispositivo se puede cambiar lo que le permite funcionar con una sensibilidad de 1,5 g o 6 g. Esta característica es ideal cuando un producto tiene diferentes aplicaciones que requieren dos sensibilidades para un rendimiento óptimo. La sensibilidad se puede cambiar en cualquier momento durante el funcionamiento del producto. El pin de selección de gravedad se puede dejar sin conexión para las aplicaciones que sólo requieren una sensibilidad de 1.5g.

d. ***Sleep Mode*:** El acelerómetro de 3 ejes ofrece un modo de reposo que es ideal para conservar la vida útil de la batería o para reducir el consumo de energía. Cuando el modo de reposo se activa, las salidas del dispositivo se apagan, proporcionando una importante reducción en la corriente de funcionamiento.

e. ***Filtering*:** El acelerómetro posee diferentes condensadores internos para realizar un filtraje tanto a las señales de entrada como las de salida [34].

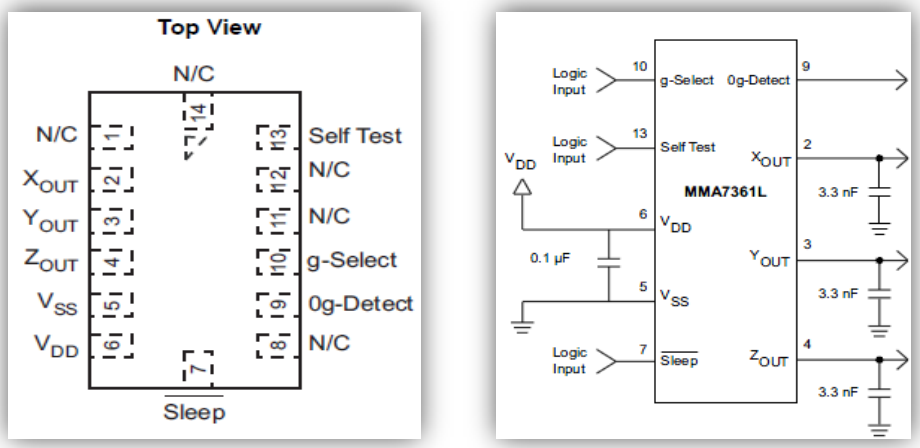


Figura 2.8 Distribución de pines acelerómetro MMA7361L *Freescale* [34].

2.3.1.3 Ventajas y desventajas del acelerómetro

En la Tabla 2.4 se muestran algunas de las múltiples y diversas ventajas que posee el acelerómetro MMA7361L y las pocas desventajas que presenta, por lo cual afianza a este sensor como un dispositivo de alta fidelidad y fiabilidad para las tareas que se desean ejecutar en el proyecto.

Tabla 2.4 Comparación entre ventajas y desventajas del acelerómetro.

VENTAJAS	DESVENTAJAS
Adaptable a diferentes entornos	No apto para ambientes agrestes
Bajo costo	Requiere tarjeta de acondicionamiento de señales
Fácil implementación	Susceptible a temperaturas mayores a 85 ºC

Bajo consumo de energía	
Alta sensibilidad	
Robustez y resistencia a impactos	

2.3.2 Regulador de voltaje

Para normalizar el voltaje de la entrada del acelerómetro, se escogió el dispositivo LD1117D33 de STMicroelectronics. Este es un regulador de voltaje de baja potencia, capaz de proporcionar hasta 800 mA de corriente de salida, con un voltaje de referencia ajustable de 1,25 V. Este integrado ofrece las siguientes tensiones de salida: 1.2V, 1.8V, 2.5V, 2.85V, 3.0V, 3.3V, 5.0V. Para el desarrollo de la tarjeta es necesario escoger como voltaje de salida 3.3 voltios, debido a que esta es la tensión necesaria para excitar el acelerómetro.

2.3.2.1 Especificaciones

- Voltaje mínimo de tensión 1V.
- Voltaje máximo de entrada 15V.
- Potencia máxima de disipación 12W.
- Corriente de salida de 800mA.
- Tensión de salida fija de: 1.2V, 1.8V, 2.85V, 3.0V, 3.3V, 5.0V.
- Voltaje de referencia de 1.25V
- Límite de corriente interna y térmica, variable entre ±1%(a 25ºC) y el 2% en su máximo rango de temperatura [35].

2.3.2.2 Componentes adicionales

Para el desarrollo final de la tarjeta se empleó una resistencia para limitar la corriente en la entrada del pin *Sleep* del sensor. También se utilizó capacitores de 3,3nF, 0,1µF,1 µF y 10 µF, sobre el regulador de voltaje y sobre las salidas de los

ejes X,Y y Z del acelerómetro, esto con el fin de limpiar las señales de entrada y de salida respectivamente. De igual manera se uso puntos de salida para cada uno de los pines del sensor, para facilitar el acople con otros dispositivos. El plano se diseñó en Eagle [36] (Figura 2.9), para posteriormente realizar la PCB con elementos superficiales.

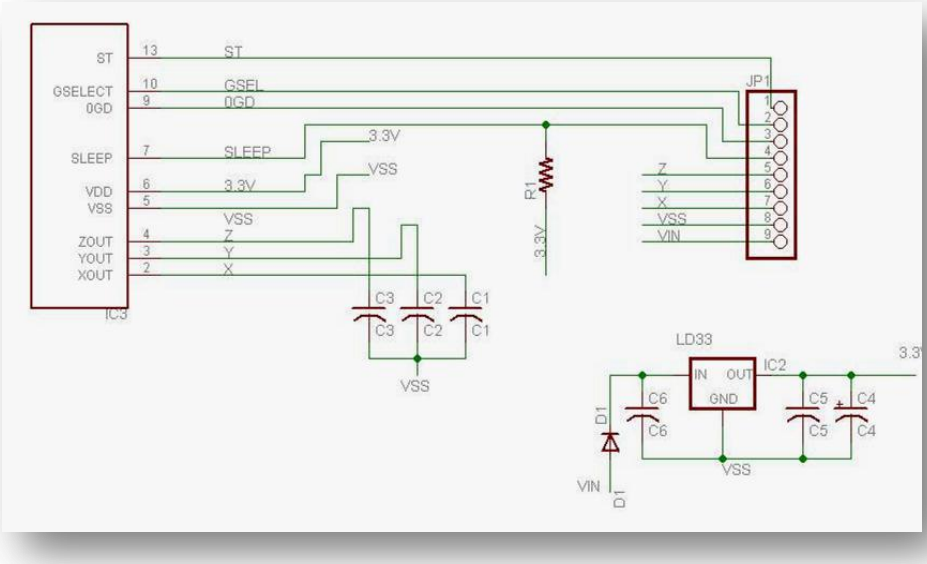


Figura 2.9 Planos en Eagle 5.11 para tarjeta de acondicionamiento de señales.

2.4 Software y hardware básico

2.4.1 Implementación en LabView

Labview es un entorno de programación gráfica usado comúnmente por ingenieros e investigadores para desarrollar sistemas sofisticados de medida, pruebas y control usando íconos gráficos e intuitivos, además de cables los cuales se asemejan mucho a los diagramas de flujo. Ofrece una integración incomparable con miles de dispositivos de hardware y brinda cientos de bibliotecas integradas para análisis avanzado y visualización de datos, todo para crear instrumentación virtual. La plataforma LabView es escalable a través de múltiples objetivos y sistemas operativos, desde su introducción en 1986 se ha vuelto uno de los lideres en la industria

Dentro de las características más relevantes de este software encontramos: programación más rápida, la posibilidad de integración con hardware, análisis avanzado de flujos de información, visualización de datos mediante múltiples interfaces de usuario, almacenamiento de datos y generación de reportes.

Los programas que se desarrollan sobre esta herramienta se denominan VI (Instrumento Virtual), lo que da una idea de la funcionalidad y los alcances que tiene. Algunos de los principales usos de LabView son: adquisición de datos, automatización industrial, control de instrumentos, diseño de controladores [37].

Para esta aplicación particular se uso una de las últimas versiones disponibles (LabView 2009).

2.4.2 Entorno de Labview

El tipo de programación que se utiliza es el tipo G que significa programación grafica, la cual en este caso tiene dos componentes, un panel frontal que es donde se encuentran los elementos con los cuales interactúa el usuario, es decir donde se diseña la interfaz de usuario y por consiguiente donde se ubican los controles e indicadores, y el panel de código que como su nombre lo indica es sobre el cual se estructura la lógica que permite el funcionamiento de la aplicación.

LabView posee VIs prediseñados que facilitan mucho algunas tareas específicas tal y como son: la adquisición de datos e imágenes, la comunicación con dispositivos a través de puertos del ordenador, el procesamiento digital de señales, funciones matemáticas simples, etc.

2.4.3 Rutina de adquisición de datos con instrumentos virtuales

El programa desarrollado se encuentra encerrado dentro de un bucle (*while*), donde la condición de finalización se asocia con un control que el usuario puede encontrar en el panel frontal, lo cual permite que el programa se ejecute de manera continua, a menos de que dicho control se active. Dentro de este ciclo encontramos un bloque de *frames*, que permite que cada uno de los cuadros se vaya ejecutando en orden de izquierda a derecha. En el primer *frame* se encuentra un VI prediseñado que nos permite establecer comunicación con la tarjeta de adquisición de datos de National Instruments (NI 6008), y generar adquisición de

datos por tres de los ocho canales analógicos disponibles (ya que se configuró la tarjeta para trabajar en modo no diferencial). Por estos canales se leerá el valor de voltaje que entrega el sensor asociado con el valor de aceleración al cual se encuentra sometido para cada uno de los tres ejes coordenados. Estos valores de voltaje adquiridos son enviados a indicadores numéricos para que puedan ser observados por el usuario en el panel frontal cuando la aplicación se esté ejecutando y se guarden en variables X, Y, Z, para que puedan ser utilizados más adelante (ver Figura 2.10).

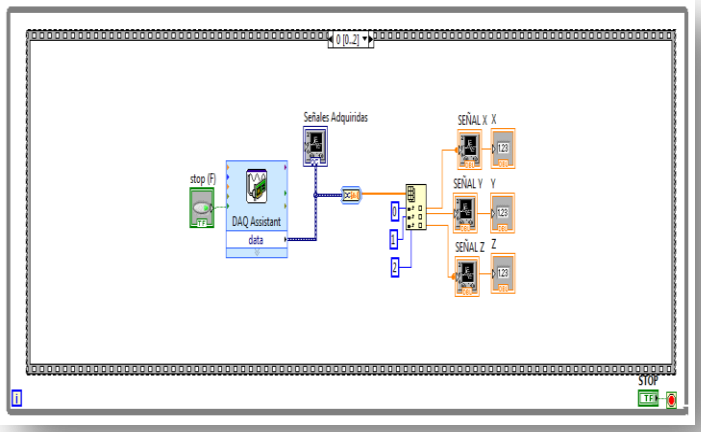


Figura 2.10 *Frame* para adquisición de datos.

Para poder generar la rotación de un objeto tridimensional creado sobre esta plataforma, se requiere convertir los valores de voltaje a valores en radianes, por lo cual el siguiente *frame*, recibe los valores de las variables X, Y, Z, y les resta un voltaje de *off-set* que entrega el sensor para cada uno de los ejes. Los resultados se pasan a nuevas variables llamadas Xrad, Yrad y Zrad, de tal forma que cuando se genera la rotación del sensor en cierto sentido, se generen valores positivos, y en sentido contrario valores negativos (ver Figura 2.11).

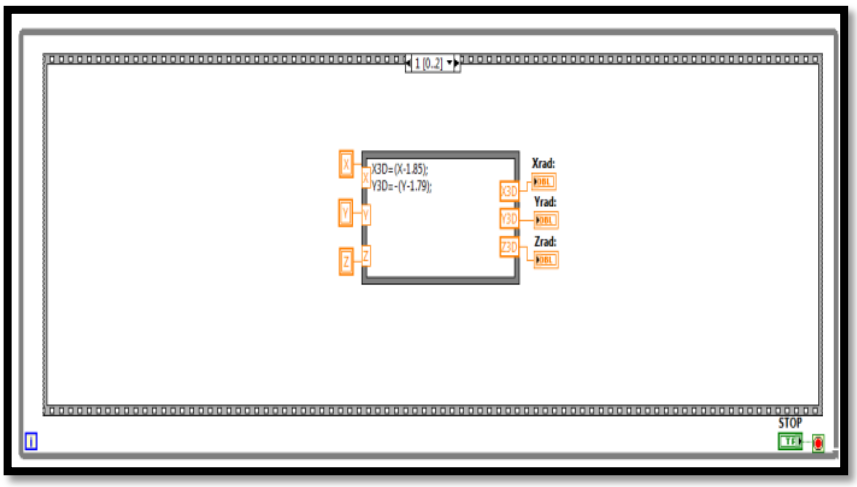


Figura 2.11 *Frame* para rotación del objeto en 3D.

En el último *frame*, se crea una escena tridimensional y un objeto en tres dimensiones (en este caso un cubo), se asignan valores a ciertos parámetros de estos elementos, como por ejemplo la textura, y se genera la rotación del objeto dentro de la escena, de acuerdo a los valores que toman las variables que se obtuvieron anteriormente (ver Figura 2.12).

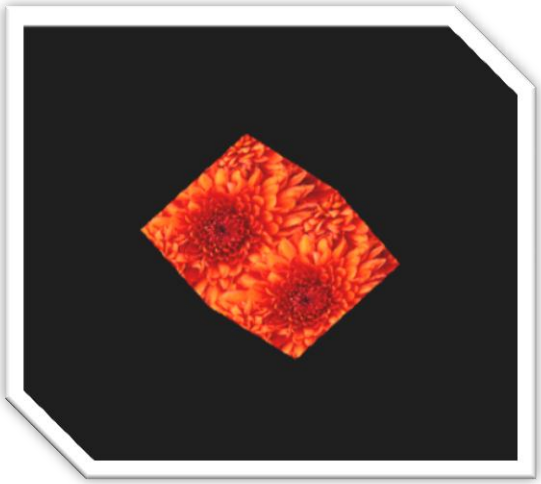


Figura 2.12 Escena del movimiento del cubo en 3D.

2.4.4 Implementación del hardware del dispositivo electrónico

Para el diseño del dispositivo electrónico se decidió utilizar diferentes componentes para resolver el problema de posición de un objeto en tres dimensiones, los elementos utilizados en el diseño del hardware son:

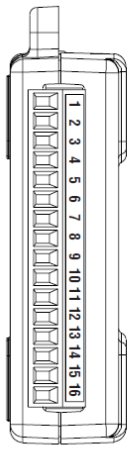
a. Acelerómetro MMA7361L

Sensor de 3 ejes acoplado a una tarjeta para acondicionamiento de señales de voltaje. Su descripción se encuentra en la sección anteriormente vista (Capítulo 2.3).

b. Tarjeta de adquisición de datos NI6008

La NI6008 es una tarjeta que permite adquirir datos ya sea analógicos o digitales, cuenta con 32 pines entre los cuales se encuentran 8 entradas analógicas, 2 salidas analógicas, 3 entradas digitales, 8 salidas digitales, 1 voltaje de referencia externa, voltaje de polarización y pin de configuración externa [38]. En la Tabla 2.5 se puede ver la distribución de pines del modulo analógico.

Tabla 2.5 Distribución de pines de la NI6008.

Module	Terminal	Signal, Single-Ended Mode	Signal, Differential Mode
	1	GND	GND
	2	AI 0	AI 0+
	3	AI 4	AI 0–
	4	GND	GND
	5	AI 1	AI 1+
	6	AI 5	AI 1–
	7	GND	GND
	8	AI 2	AI 2+
	9	AI 6	AI 2–
	10	GND	GND
	11	AI 3	AI 3+
	12	AI 7	AI 3–
	13	GND	GND
	14	AO 0	AO 0
	15	AO 1	AO 1
	16	GND	GND

La NI6008 es utilizada en el proyecto para leer las señales que provienen del sensor (acelerómetro) las cuales son de tipo analógico por tanto, se utilizaron tres canales analógicos para adquirir las señales de los tres ejes del dispositivo.

- **Amplificador operacional LM324**

El LM324 consiste en una tarjeta que contiene 4 operacionales independientes de alta impedancia de entrada, diseñados para operar en diferentes rangos de voltaje que oscilan entre -15V y +15 V.

Especificaciones técnicas:

- Rango de voltaje de alimentación: $3V_{DC}$ a 30 V_{DC} o fuente de alimentación dual: ±1.5 V a ± 15V.
- Entrada baja de corriente: $45nA_{DC}$
- Entrada baja de voltaje de *offset*: $2mV_{DC}$ y corriente de *offset* $5nA_{DC}$.
- Rango de entrada de voltaje diferencial igual al voltaje de alimentación.
- Salida de voltaje: $0V_{Dc}$ a V_{CC} – 1.5 V_{DC} [39].

Distribución de pines:

El integrado LM324 cuenta con 14 pines entre los que se encuentran 4 salidas, 4 pines de entrada positiva, 4 pines de entrada negativa, voltaje de alimentación y el último pin corresponde a la tierra [39]. En la Figura 2.13 se puede observar cada uno los pines con su respectiva numeración:

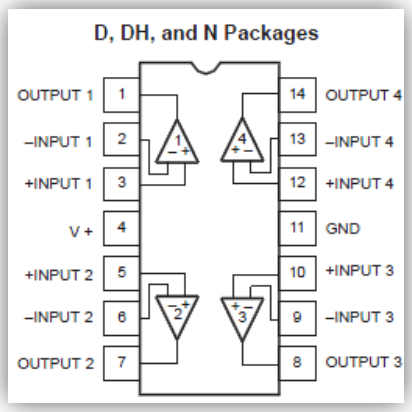


Figura 2.13 Amplificador operacional LM324 [39][39].

Mediante estos dispositivos y la utilización de una protoboard para el montaje de estos elementos, se realizó un circuito constituido por tres etapas. La primera etapa es la de sensado; en la cual se lee información del entorno mediante el uso del acelerómetro. Este se encarga de medir cambios en la aceleración de la gravedad que este puede sufrir mientras se efectúa un movimiento, estos cambios son interpretados y enviados a la etapa de amplificación, por medio de señales de voltaje las cuales son del orden de los milivoltios por cada gravedad ejercida. Después de esta etapa se llega a la fase de amplificación, en donde se lee las señales provenientes del sensor y se procede a amplificar cada una de estas señales que caracterizan los 3 ejes (X, Y, Z). El proceso de amplificación se realiza mediante un amplificador operacional, el cual se configuró para que a su salida se obtenga el doble de la señal de entrada. Por último se encuentra la etapa de adquisición de datos, ésta permite ingresar los datos o la información obtenida en la etapa de amplificación y de sensado hacia el ordenador, con el fin de plasmar toda esta información en un software, que nos ayude a interpretarla de una manera gráfica.

3. Sistema virtual

3.1 Motor gráfico Ogre 3D

Para el desarrollo del sistema virtual se utilizó **Ogre 3D** (*Object-Oriented Graphics Rendering Engine*). Este es un motor gráfico de código abierto, orientado a objetos e implementado sobre lenguaje C++.Este software tiene la particularidad de tener una licencia GNU o Licencia Publica General Menor (LGPL), por tanto se puede utilizar sin el costeo de licencias, aunque con algunas restricciones [40].

Ogre fue creado en el 2001 como una plataforma de renderizado para gráficos en tres dimensiones, el cual puede ser usado para creación de juegos, simuladores, arte interactivo, entornos virtuales, visualización científica, etc. [41].

Este motor gráfico utiliza diferentes librerías gráficas como OpenGL y Direct3D de Microsoft, lo que permite convertirlo en un software multiplataforma que puede ser acoplado a diversos sistemas operativos como, Linux, Windows, Mac OS. Por otra parte Ogre basa su funcionamiento en el uso de clases y subclases. Para lograr entenderlo mejor se presenta el siguiente diagrama, en donde solo se ilustran los componentes más importantes de Ogre (Figura 3.1) [40].

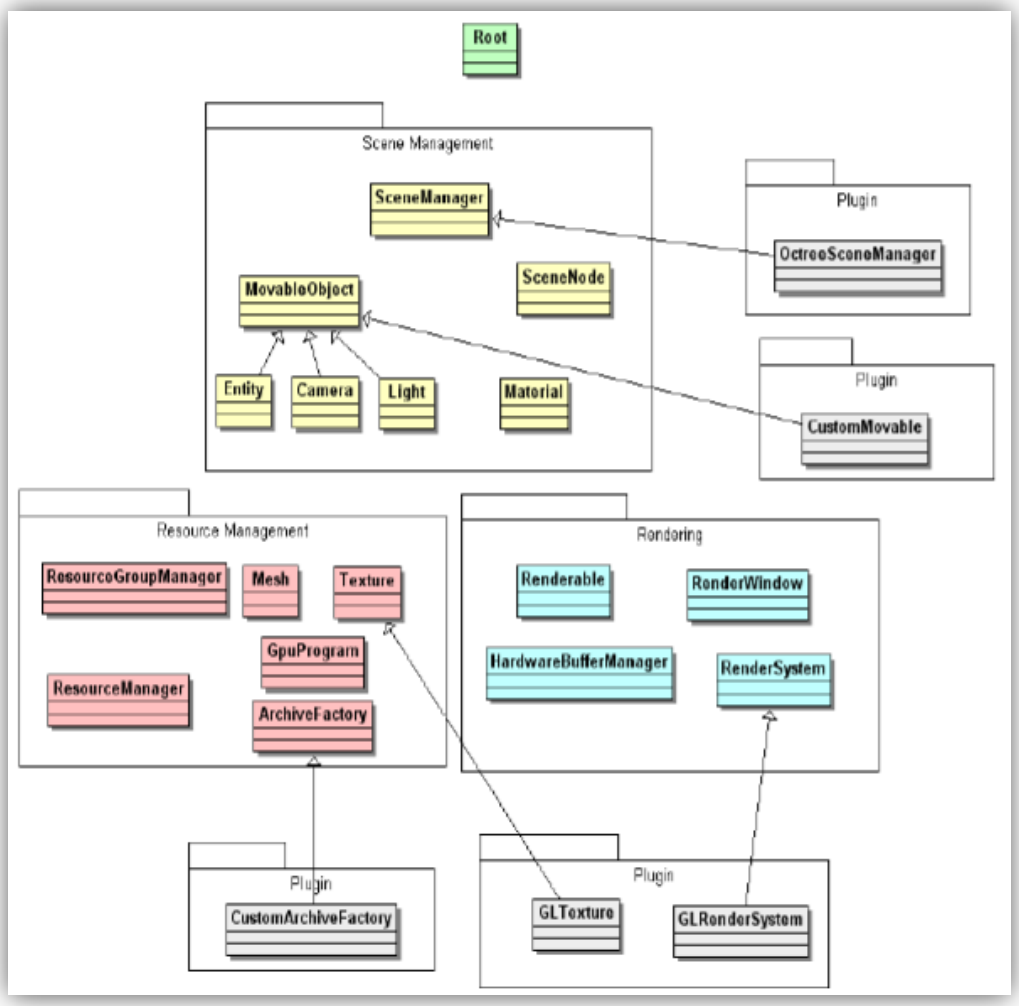


Figura 3.1. Esquema general de componentes de Ogre [41].

En la parte superior del diagrama se encuentra el objeto raíz (*Root*). Esta es la manera que se representa en el sistema de Ogre, y el lugar en donde se crean los objetos del nivel más alto, que se tienen que desarrollar, tales como los directores de escena (*scene managers*), sistemas de renderizado y ventanas de renderización, cargar *plugins* y todas las demás funciones elementales. La mayoría del resto de clases que tiene Ogre se pueden evidenciar en el Anexo A

3.1 Diferentes paquetes utilizados para el ambiente virtual

Dentro del desarrollo del ambiente virtual se utilizaron además de Ogre 3D otros programas tales como Visual Studio 2005, Blender y Solid Edge ®. El primer software es un entorno de desarrollo integrado (IDE), para sistemas operativos Windows, el cual soporta varios lenguajes de programación como Visual C++, Visual C#, Visual J#, ASP.NET Y Visual Basic.NET. Visual Studio permite desarrollar aplicaciones, sitios web, etc. En el proyecto sirvió para compilar el código desarrollado bajo el lenguaje Visual C++, con lo cual se creó la aplicación del ambiente virtual para posteriormente renderizarlo con el motor gráfico Ogre 3D [42].

Por otra parte se usaron programas como Solid Edge ®, Blender y Make Human para la implementación de las Piezas del Robot, el modelado del cuerpo del paciente y los respectivos órganos. Solid Edge ®(ver Figura 3.2) es un programa parametrizado de diseño asistido por computadora de piezas tridimensionales. Permite el modelado de piezas de distintos materiales, doblado de chapas, ensamblaje de conjuntos, soldadura, funciones de dibujo en plano para ingenieros, etc. [43].

Blender (ver Figura 3.3) es un programa informático multiplataforma, dedicado especialmente al modelado, animación y creación de gráficos tridimensionales. Actualmente es compatible con todas las versiones de Windows, Mac OS X, Linux, Solaris, FreeBSD e IRIX [44]. Este software se ve más en detalle en la siguiente sección.

MakeHuman es un programa libre con licencia GPL muy útil para crear cuerpos humanos en tres dimensiones, permitiendo modificar detalladamente cada parte del rostro y del cuerpo generando cualquier expresión y movimiento.

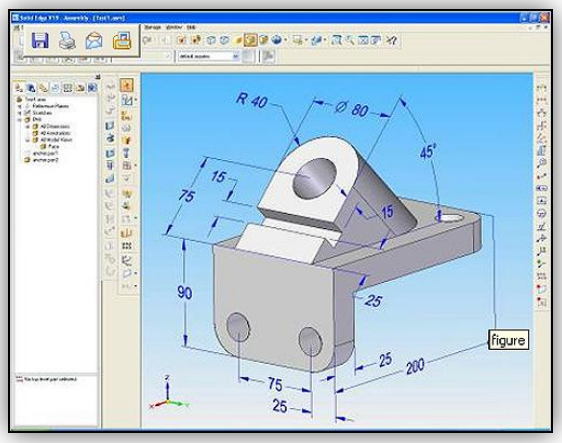


Figura 3.2 Interfaz gráfica de Solid Edge ® [43].

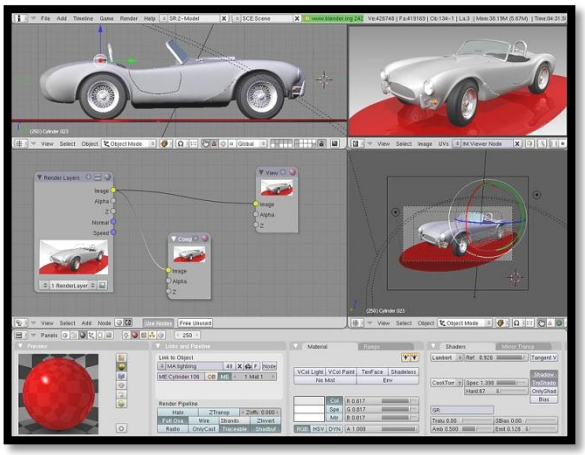


Figura 3.3 Interfaz gráfica de Blender [44].

3.2.1 Modelado en tres dimensiones mediante Blender

Blender es un software de código abierto, que integra una serie de herramientas para la creación de un amplio rango de contenidos 3D, con los beneficios añadidos de ser multiplataforma y tener un tamaño de unos 5 MB.

Destinado a artistas y profesionales de multimedia, Blender puede ser usado para crear visualizaciones en tres dimensiones, tanto imágenes estáticas como vídeos de alta calidad, mientras que la incorporación de un motor de 3D en tiempo real permite la creación de contenido interactivo que puede ser reproducido

independientemente, para ver más en detalle las características y funciones de Blender ir al Anexo B.

Después de tener claro el funcionamiento de este software se procede a realizar la instalación, la cual se especifica en el ítem Instalación de Blender y su utilización del Anexo C. Unas vez completo el proceso se comienza elaborar los diferentes órganos que componen el abdomen del paciente y el cuerpo humano, en donde se albergan los mismos. Para ello en la siguiente sección se explicará con un ejemplo cómo se elaboró cada uno de estos elementos [44].

3.2.1.1 Modelado del cuerpo humano en 3D

Para ejemplificar como se modela en Blender, se tomó como ejemplo el cuerpo humano, debido a su alto grado de dificultad al momento de crear y diseñar, con lo cual se logra explicar gran parte de las funciones que posee este software.

3.2.1.1.1 Cuerpo básico

Para modelar el cuerpo del paciente que se utilizó en el entorno virtual, se debe tener como base un objeto en tres dimensiones, que para nuestro caso será un cubo. Con éste se pretende realizar un contorno de la estructura básica del cuerpo de una persona. En primera instancia se deselecciona el cubo (tecla A), se muestran los vértices (tecla Tab), se escoge el cursor de selección (tecla B dos veces) y con él se eligen los vértices que componen la cara superior del cubo. Con ello se hace un cubo contiguo al que se tiene por defecto mediante la función extruir (tecla E), obteniendo así la forma de las piernas, tronco y brazos del cuerpo humano, como se muestra en las Figuras 3.4 y 3.5.

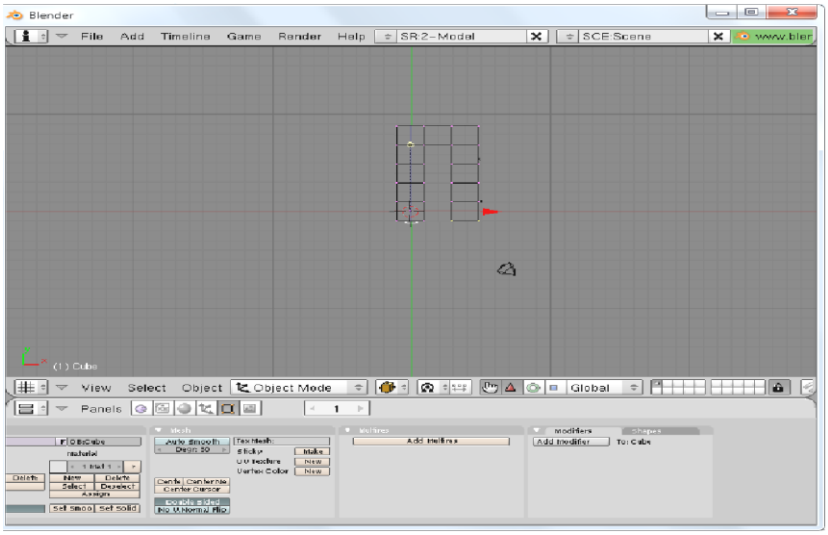


Figura 3.4 Extremidades inferiores del cuerpo humano.

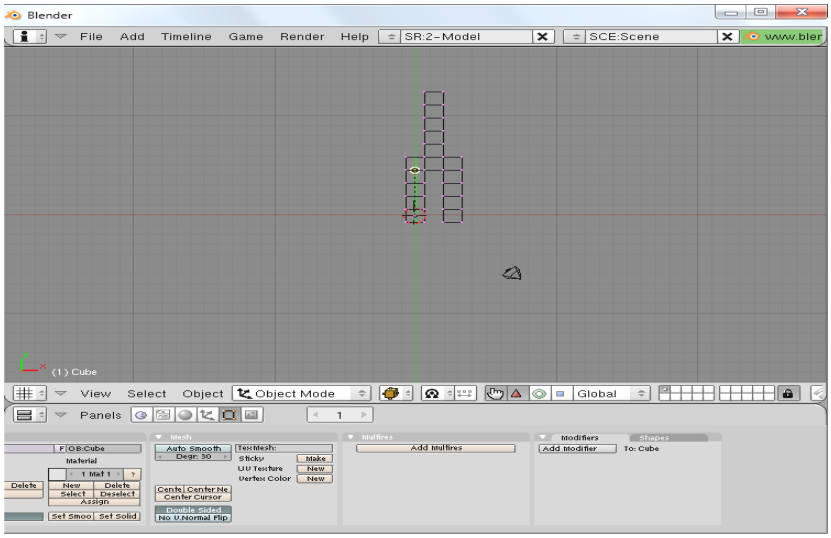


Figura 3.5 Tronco del cuerpo humano.

Al finalizar con el tronco y extremidades superiores e inferiores del cuerpo humano, se realiza la cabeza, ubicando el cursor sobre el centro del cubo superior (Cuello) y se añade una esfera. Para hacerlo en Blender se utiliza la tecla espacio, escoger "*Icosphere*" dentro de las opciones que se despliegan y determinar el tamaño adecuado ingresando el valor del radio de la esfera, con ello se obtiene la cabeza del cuerpo humano (ver Figura 3.6).

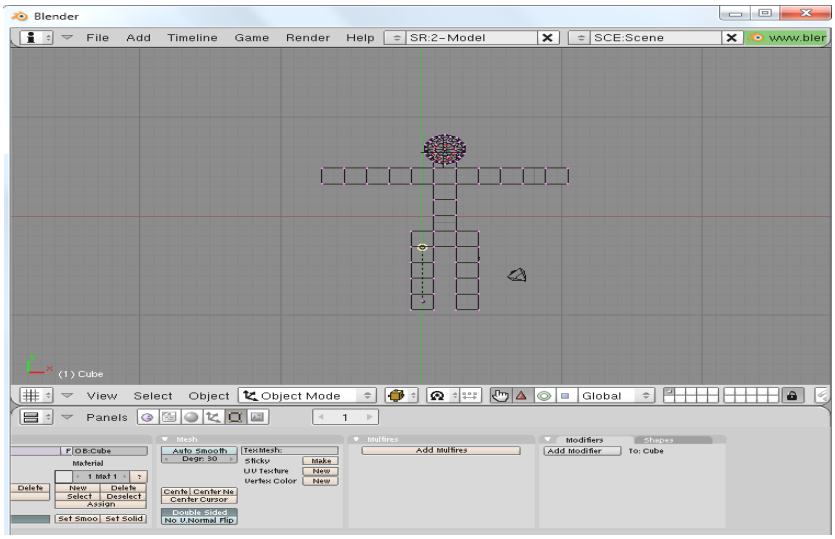


Figura 3.6 Cuerpo humano básico

3.2.1.1.2 Detallado del cuerpo humano

Para hacer que nuestro modelo sea más realista se deben redondear las líneas afiladas de las caras que se topan unas con otras. Esto se genera seleccionando la totalidad de nuestro objeto, y sobre el panel inferior, en el modo de edición (tecla F9) se encuentra una pestaña llamada "*Modifiers*", se elige y se da click sobre "*Add Modifiers*". Éste extiende un menú en donde se encuentra *Subsurfaces*, se da click sobre esta viñeta y con ello termina la operación.

Otro de los detalles que se debe hacer es el suavizado de toda la unidad. Se consigue mediante un botón ubicado sobre el panel inferior de la interfaz de Blender. Esta función nos permite suavizar todos los bordes y unir todos los cubos que se habían hecho de forma individual (ver Figura 3.7).

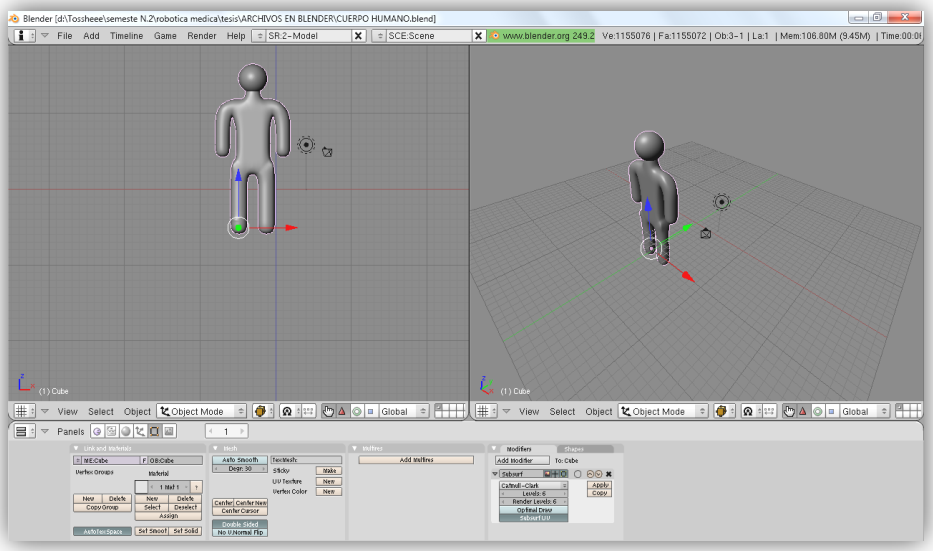


Figura 3.7 Vistas del modelo suavizado.

Teniendo el modelo integrado y suavizado, se empezó a darle color y forma a detalles como el pecho, brazos y piernas, mediante funciones localizadas sobre el panel de diseño. En primera medida se escogió un color base, el cual se genera en Blender utilizando la barra de herramientas, establecida en el panel inferior de la interfaz. En ella se encuentra una opción llamada "*Material buttons*", se elige y dentro de esta pestaña ubicamos "*Materials*". Aquí se deben visualizar tres barras modificables que nos dan la opción de cambiar el color del objeto en el que estamos trabajando. Cada una de ellas representan el modelo de color RGB (del inglés *Red, Green, Blue*; "rojo, verde, azul"), éste hace referencia a la composición del color en términos de la intensidad de los colores primarios con que se forma. Por consiguiente se debe variar cada uno de los rangos (el valor oscila entre 0 y 1) de las barras para obtener el color, que para nuestro caso es el rosado o color piel (en la barras de color; R: 1.00, G: 0.637, B: 0.461).

Con el color escogido se debe empezar a formar la estructura muscular del cuerpo humano, utilizando las funciones de expandir y contraer. En este diseño se modelaron los músculos del pecho (pectoral mayor y menor), brazo (bíceps, tríceps y antebrazo), hombros (deltoides), cuello (trapecio), espalda (dorsal y lumbar) y piernas (cuádriceps, abductores, aductores y pantorrillas). Para formarlos se debe deseleccionar el cuerpo (tecla A) y activar los vértices de las mallas que componen la unidad (tecla Tab), elegir los puntos en los que vamos a

trabajar (tecla B dos veces) y escalar (tecla S), según la forma del músculo en la que se esté trabajando. De esta manera se le da forma al cuerpo humano (ver Figura 3.8)

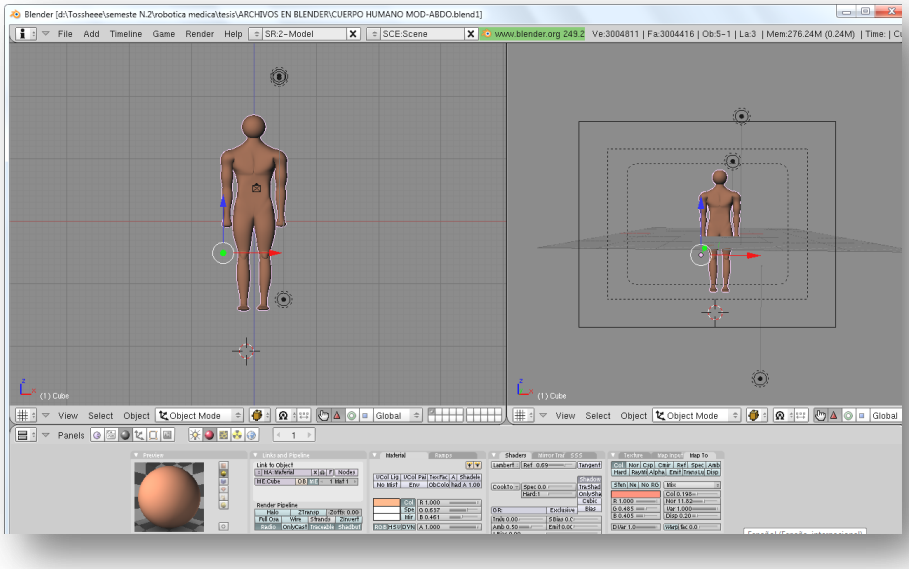


Figura 3.8 Vista del modelo con detalles de tronco y extremidades.

Por otra parte el modelo debe contar con detalles que caracterizan a una persona, tales como la forma del rostro, manos y pies. Es por ello que se diseñaron adicionando objetos predeterminados por el software (*Icosphere, UVsphere, Cilynder,* etc.) y dándole su forma particular mediante las funciones anteriormente vistas, con lo que se logró darle realismo a la figura que se está representando (ver Figura 3.9).

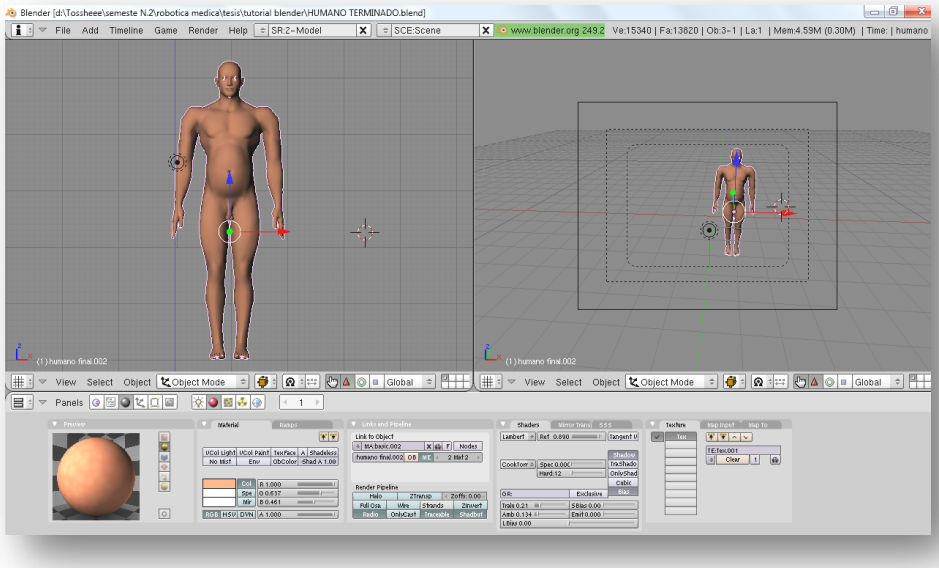


Figura 3.9 Modelo del cuerpo humano con detalles en el rostro manos y pies.

Por último se creó el abdomen y se le dio texturas al color base, debido a que los requerimientos del proyecto exigen que el paciente debe tener la cavidad abdominal insuflada. Esto se efectuó escogiendo los vértices de las mallas que componen la zona muscular del estómago y se escaló tanto en ancho como en alto (ver figura 3.10). Finalizando el modelo se utilizaron diferentes texturas para mejorar el aspecto del cuerpo humano. Esto se hizo con la ayuda de la barra de herramientas, escogiendo la viñeta llamada "*Texture*", añadiendo una nueva textura (*add new*) y seleccionando "Voronoi", que es un ítem que difumina el color base del modelo. Esto con el fin de darle mejor aspecto al modelo, como se puede observar en la Figura 3.10.

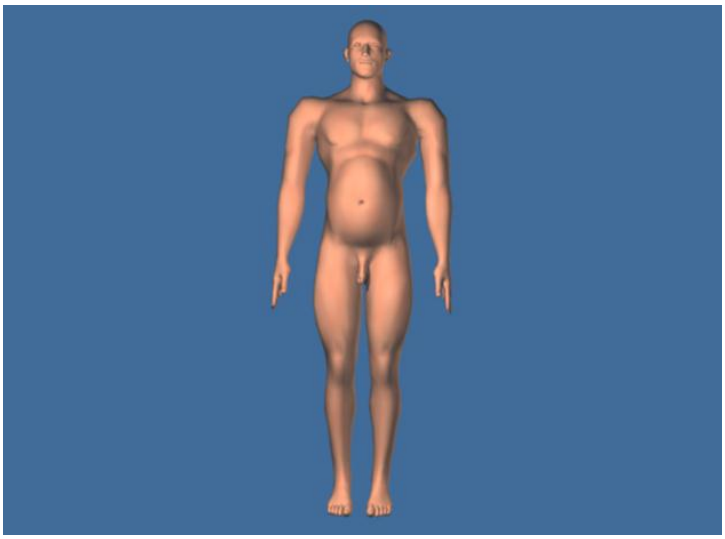


Figura 3.10 Modelo terminado del cuerpo humano.

3.2.1.2 Modelado del hígado en 3D

Para ejemplificar como se modela los órganos del cuerpo humano en Blender, se escogió el hígado, ya que este tiene una forma muy particular, lo cual hace que su elaboración sea compleja.

Para modelar el órgano se tuvo como objeto base una esfera. Ésta se adiciona sobre el espacio de trabajo (tecla Espacio), y se escoge "*Icosphere*" dentro de las alternativas que se despliegan y se determina el tamaño adecuado ingresando el valor del radio de la esfera (ver Figura 3.11). Posteriormente se seleccionan (tecla B dos veces) los vértices de la parte Inferior de la esfera, mediante el cursor y se orientan hacia abajo, con el fin de formar un ovalo. Al termino de esta ejecución se procede a seleccionar (tecla B dos veces) los vértices de la parte superior derecha y se extienden sobre el eje X, buscando formar un cono. Enseguida se deselecciona el objeto (tecla A) y se presiona la tecla X, para cambiar el modo de selección de vértices a bordes (*Edges*). Esto se hace para poder escoger (tecla B dos veces) los bordes de la parte central del objeto y reducirlo a la mitad (tecla S), de esta manera se pretende realizar la estructura básica del hígado (ver Figura 3.12).

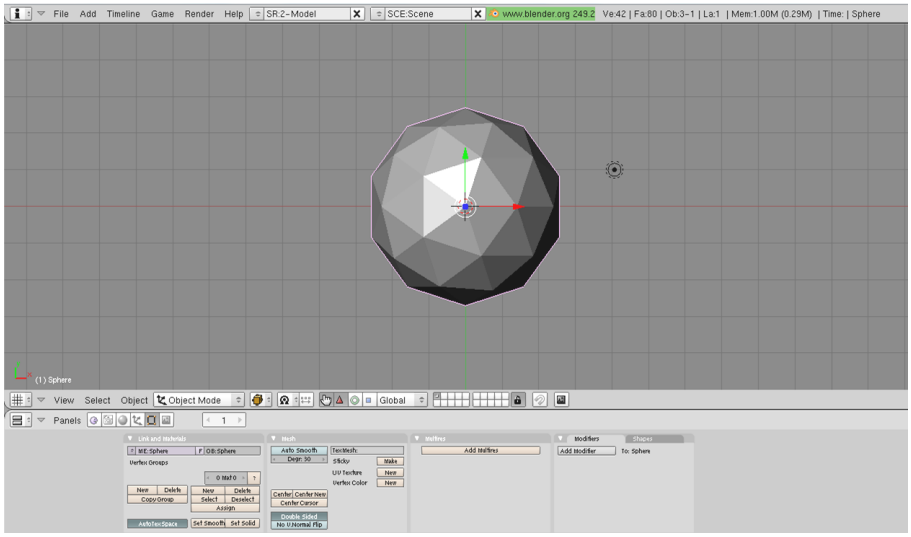


Figura 3.11 Esfera básica para el modelamiento del hígado.

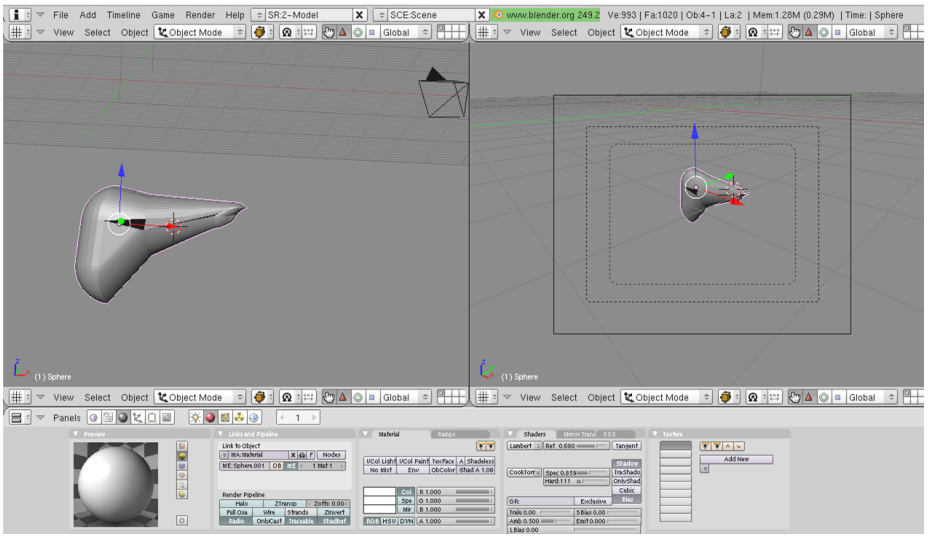


Figura 3.12. Estructura básica del hígado.

Para finalizar con el modelamiento se deben suavizar los bordes y caras, tal y como se hizo con el modelo del cuerpo humano, además de darle el color y la textura correspondiente, que para este caso se escogió el vino tinto (en la barras de color; R: 1.00, G: 0.116, B: 0.113), y "*Clouds y Musgrave*" como textura. Esto se puede observar la Figura 3.13.

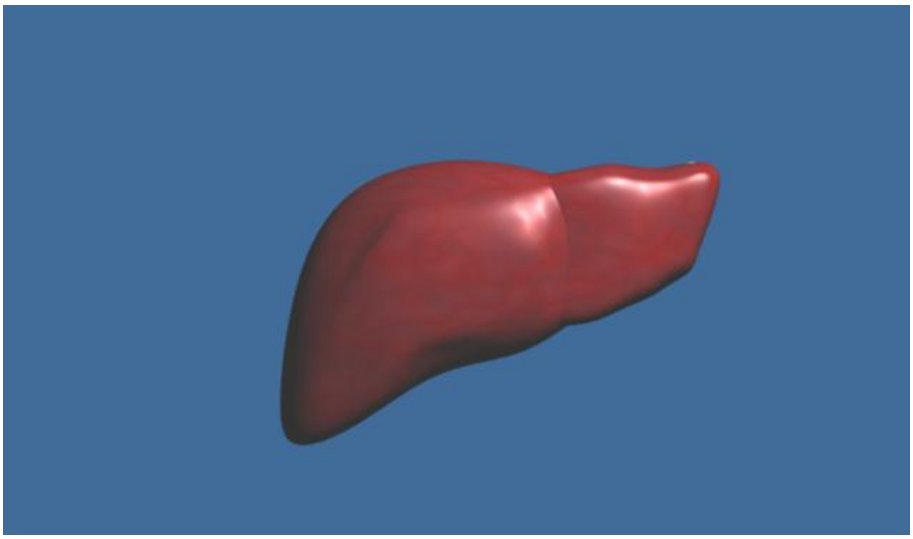


Figura 3.13. Hígado terminado.

3.2.1.3 Instalación de herramientas adicionales

Para realizar tareas especificas en Blender como exportar o importar objetos desde otro *software,* se debe realizar la instalación de dos herramientas:

a. Python 2.71:

Python es un lenguaje de programación dinámica muy potente que se utiliza en una amplia variedad de dominios de aplicación, que permite trabajar sobre sistemas operativos Windows, Linux / Unix, Mac OS X. Para realizar la instalación se debe seguir con los siguientes pasos:

1. Descargar el instalador Python 2.7.1 para Windows dirigiéndose a http://www.python.org/ftp/python/

2. Hacer doble clic en el instalador, Python-2.7.1.exe.

3. Se debe seguir con los pasos que indique el instalador.

4. Al finalizar la instalación se debe reiniciar el equipo.

b. OgreXMLConverter

El OgreXmlConverter es una herramienta de línea de comandos que puede convertir archivos binarios '*.mesh*' y '*.skeleton*' .a XML y viceversa. Para la instalación se debe seguir con los pasos que se muestran a continuación:

1. Descargar el instalador OgreXMLConverter desde http://www.ogre3d.org/ tikiwiki/ Tools%3A+Blender.

2. Dar click sobre OgreXMLconverter.exe, e instalarlo en el directorio "C:\Archivos de Programa\Blender Foundation\Blender\.blender\scripts".

3. Seguir con los pasos que sugiere el instalador.

4. Al finalizar con la instalación reiniciar el equipo.

3.2.1.4 Exportación de modelos de Blender a Ogre 3D

El primer paso a desarrollar es verificar si Python 2.71 está vinculado a Blender. Una vez verificado que Blender funciona de manera correcta, se ingresa a Blender al menú *File*, y se busca la opción *Export,* de esta manera tendremos a nuestra disposición la opción Ogre Mesh, con la cual se realiza el proceso de exportación de cada uno de los objetos. Para ver esto más en detalle dirigirse al ítem "Trabajando con el script OgreMeshes" del Anexo C.

3.2 Desarrollo del Sistema Virtual

Dentro de esta sección se explican segmentos de código importantes en el desarrollo del proyecto, que le pueden brindar una comprensión clara al lector del funcionamiento del dispositivo y de la forma en que se adquieren y se utilizan los datos para controlar el robot Hibou. Primero se explica en un bloque de texto lo que se pretende hacer y entre paréntesis la línea dentro del programa a la cual corresponde.

Se trabajó con Visual Studio 2005 actualizado con SP1 y con la versión de Ogre SDK correspondiente (Ogre 1.7.2). También se instaló el Ogre Wizard que permite

tener a disposición la plantilla para proyectos tipo Ogre sobre la cual se trabajó y donde se encuentran las configuraciones más importantes y comunes en lo referente a aplicaciones gráficas.

Dentro del desarrollo lo primero que se implementó fue la parte visual de la aplicación. Para esto se modificó la subfunción *createScene(void)*, dentro de la cual se deben cargar cada uno de los elementos de la escena que se pretende crear. Para que dichos elementos puedan ser apreciados en la pantalla, se crea una luz de ambiente con ciertos atributos de color (línea 1), y se aplica una técnica de sombras para aumentar el realismo (línea 2). Así mismo se establecen tres puntos de luz ubicados en diferentes lugares del ambiente virtual y se les asigna características de orientación, color, reflexión especular y difusa (líneas 3 a 7), de tal forma que los objetos dentro del entorno respondan antes estos estímulos ópticos.

```
1    mSceneMgr->setAmbientLight(Ogre::ColourValue(0.5f, 0.5f, 0.5f));
2    mSceneMgr->setShadowTechnique(Ogre::SHADOWTYPE_TEXTURE_
     MODULATIVE);
3    Ogre::Light* pointLight = mSceneMgr->createLight("pointLight");
4    pointLight->setType(Ogre::Light::LT_DIRECTIONAL);
5    pointLight->setPosition(Ogre::Vector3(250, 150, 250));
6    pointLight->setDiffuseColour(Ogre::ColourValue::White);
7    pointLight->setSpecularColour(Ogre::ColourValue::White);
```

Para poder cargar los elementos tridimensionales diseñados de manera previa en Blender, es necesario copiar los archivos con la terminación *.mesh* a la carpeta \\OgreSDK_vc8_v 1-7-2\media\models y los *.material* a la carpeta \\OgreSDK_vc8_v1-7-2\media\materials. Para hacer el llamado a cada una de las mallas y materiales que conforman el ambiente virtual se deben crear entidades a las cuales se les asigna un nombre y se le pasa como parámetro el nombre de la malla para que pueda ser visualizada dentro de la escena y pueda recibir atributos (línea 8). Dicha entidad debe ser vinculada a un nodo con un nombre diferente al de la entidad y asignarle coordenadas para su ubicación dentro del entorno (línea 9). Debido a que los archivos fueron desarrollados en Blender, estos presentan cierta orientación que no coincide con la de Ogre 3D, por esto es necesario utilizar

comandos de rotación sobre el nodo, en los ejes que no concuerden (línea 12) (ver Figura 3.14).

```
8   Ogre::Entity *estomago = mSceneMgr->createEntity("Estomago",
    "EstomagoF.mesh");
9   Ogre::SceneNode *Sestomago = mSceneMgr->getRootSceneNode()-
    >createChildSceneNode("EstomagoFNode",Ogre::Vector3(50,10,25));
10  Sestomago->attachOgbject(estomago);
11  Sestomago->scale(1.5,1.5,1.5);
12  Sestomago->pitch(Ogre::Degree(90));
```

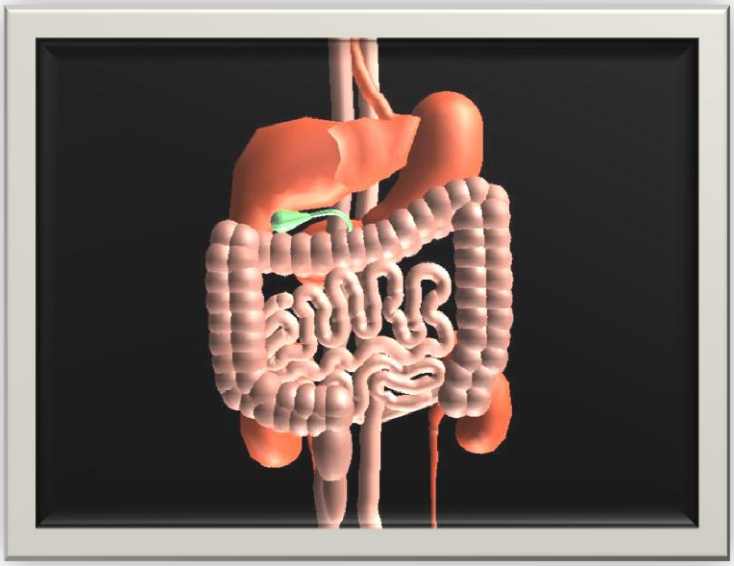


Figura 3.14 Órganos del cuerpo Humano renderizados mediante Ogre 3D.

Es necesario crear cámaras para la visualización del entorno virtual. Se trabajaron dos (líneas 13, 14), la primera es para que el usuario tenga una vista general del quirófano, paciente, robot y camilla, la otra en cambio se sitúa al final del órgano terminal del robot y permite observar el interior de la cavidad abdominal. Se configuran algunos parámetros tales como la posición de cada una de las cámaras (líneas 15, 18), la orientación (líneas 16, 19) y las distancias mínimas y máximas que se dibujan en pantalla (líneas 19.20).

```
13  mCamera = mSceneMgr->createCamera("PlayerCam");
14  m1Camera = mSceneMgr->createCamera("PlayerCam1");
15  mCamera->setPosition(Ogre::Vector3(100,50,-50));
16  mCamera->lookAt(Ogre::Vector3(0,40,0));
17  mCamera->setNearClipDistance(1);
18  m1Camera->setPosition(Ogre::Vector3(0,0,0);
19  m1Camera->lookAt(Ogre::Vector3(40,0,0));
20  m1Camera->setNearClipDistance(1);
21  m1Camera->setFarClipDistance(15);
22  mCameraMan = new OgreBites::SdkCameraMan(mCamera);
23  m1CameraMan = new OgreBites::SdkCameraMan(m1Camera);
```

3.3.1 Implementación de la ventana secundaria

Para realizar una ventana adicional sobre el ambiente virtual se hace uso del procedimiento llamado *"Viewport"*. En este se debe crear dos punteros, uno de ellos que direcciona hacia la ventana principal, y se le pasan como parámetros, el nombre de una de las cámaras anteriormente creadas, junto con valores asignados a los atributos del color de la pantalla (líneas 24 a 26).

De forma similar se asigna el segundo puntero a la ventana secundaria, junto con la cámara que le corresponde y características específicas, como el color del fondo de la ventana, el tamaño (alto y ancho) y la posición de esta sobre la pantalla principal, que para nuestro caso se ubicó sobre la esquina inferior derecha (líneas 27 a 29) (ver Figura 3.15).

```
24  Ogre::Viewport* vp = mWindow->addViewport(mCamera);
25  vp->setBackgroundColour(Ogre::ColourValue(1,1,1));
26  mCamera->setAspectRatio(Ogre::Real(vp->getActualWidth()) /
    Ogre::Real(vp->getActualHeight()));
27  Ogre::Viewport* vp1 = mWindow->addViewport(m1Camera,1,0.5,0.5,
    0.5,0.5);
28  vp1->setBackgroundColour(Ogre::ColourValue(0.3,0,0));
29  m1Camera->setAspectRatio(Ogre::Real(vp1->getActualWidth()) /
    Ogre::Real(vp1->getActualHeight()));
```

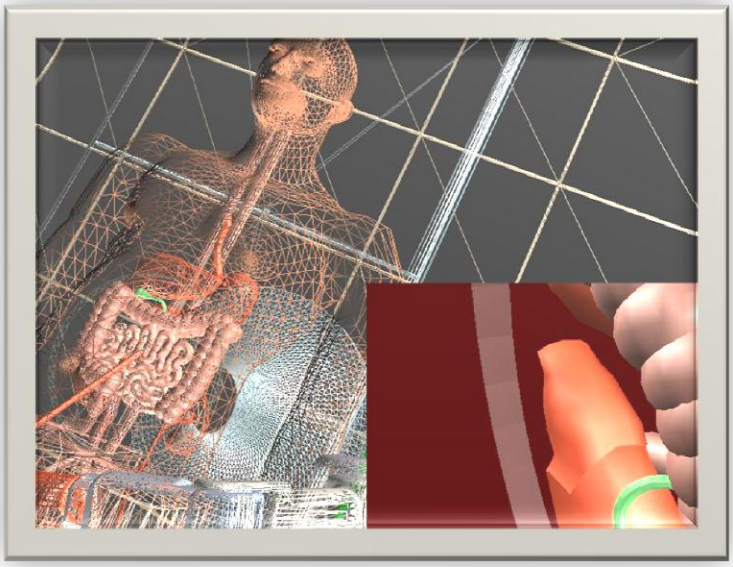


Figura 3.15 Sistema virtual con ventana secundaria.

4. Dispositivo para el control externo del robot

Para la elaboración del hardware y software del control externo del robot se determinó dividirlo en 3 partes con el fin de facilitar el proceso de elaboración del mismo. En la primera parte se elaboró la electrónica del dispositivo y el diseño del casco o compartimento en donde se alojará el sistema electrónico. En la segunda sección se implementó lo concerniente al programa que gobierna el dispositivo de control, y por último se hizo la integración de los dos sistemas.

4.1 Diseño del dispositivo electrónico

Para la implementación de la parte electrónica para el control externo del robot se tuvieron en cuenta los siguientes aspectos:

4.1.1 Sensado

Como se explicó en el capítulo anterior se utilizó un acelerómetro triaxial MMA7361L acoplado a una tarjeta acondicionadora de señales (ver capítulo 2), con el cual se miden los cambios de aceleración en cada uno de los ejes XYZ.

4.1.2 Adquisición de Datos

Para establecer una comunicación entre Visual Studio 2005 (programa ejecutado en el ordenador) y el sensor, es necesario utilizar una tarjeta de adquisición de datos que permita captar las señales entregadas por el acelerómetro, para posteriormente utilizarlas en la modificación de la posición de la cámara dentro del entorno virtual desarrollado. Dicha tarjeta se construyó a partir de los siguientes elementos:

a. Microcontrolador PIC 18F2550

Este es un dispositivo ideal para aplicaciones que demanden bajo consumo de energía, posee comunicación mediante USB 2.0 de alta velocidad (12MBit/s), tiene

28 pines de los cuales 24 son líneas de entrada y salida tanto analógicas como digitales.

Especificaciones Técnicas

- Tipo de programa memoria: Flash
- Programa de memoria : 32 KB
- CPU Speed : 12 MIPS
- Bytes de RAM: 2048
- Datos en la EEPROM: 256 bytes
- Comunicación digital periféricos: 1-A/E/USART, 1-MSSP (SPI/I2C)
- Captura / Comparación / PWM Periféricos: 2 CCP
- Temporizadores: 1 x 8-bit, 3 x 16-bit
- ADC: 10 canales, 10-bit
- Comparadores: 2
- USB: 1, a toda velocidad, USB 2.0
- Rango de temperatura : -40 A 85 ºC
- Voltaje de funcionamiento: 2 a 5,5 V
- Conteo de pines:28

Distribución de pines

Este dispositivo tiene 10 entradas analógicas, 4 salidas digitales, 3 pines para el cristal oscilador, 2 pines para salida PWM, 4 salidas discretas, 3 pines para el puerto USB (ver Figura 4.1) [45].

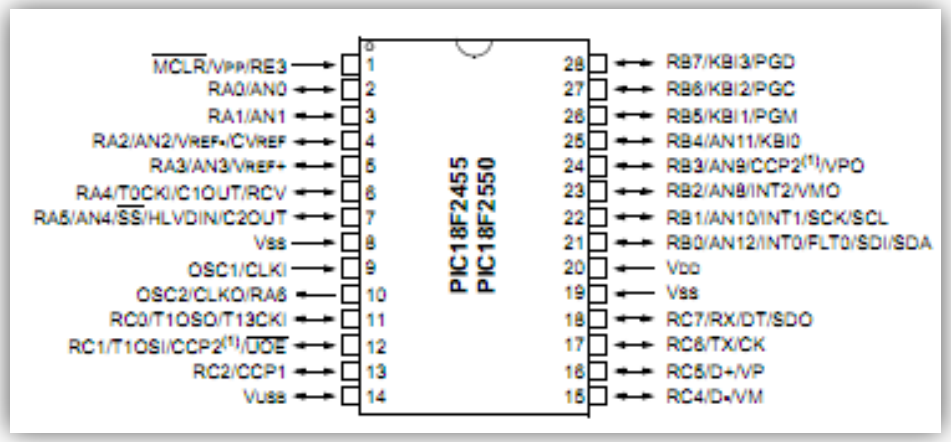


Figura 4.1 Microcontrolador PIC18F2550 [45].

b. Amplificador operacional LM324

Este dispositivo se utilizó en una configuración de seguidor de voltaje para realizar un acople de impedancias, entre el divisor de voltaje (arreglo con resistencias) y el sensor. Esto con el fin de garantizar que la alimentación del acelerómetro sea de 2.5 V. En la capitulo anterior (sección 2.4.3) se explicó con mayor detalle las especificaciones y características de este integrado.

c. Componentes adicionales

Para la implementación final de la tarjeta se usaron algunas resistencias para limitar corriente en las entradas digitales del dispositivo, leds para indicar el estado de la tarjeta (conectado o no conectado) y un divisor de voltaje (implementado también con resistencias) para de esta manera poder alimentar el sensor de aceleración utilizado (MMA7361LT). Por último se debe realizar la conexión de los diferentes dispositivos la cual se puede observar en la siguiente imagen (Figura 4.2).

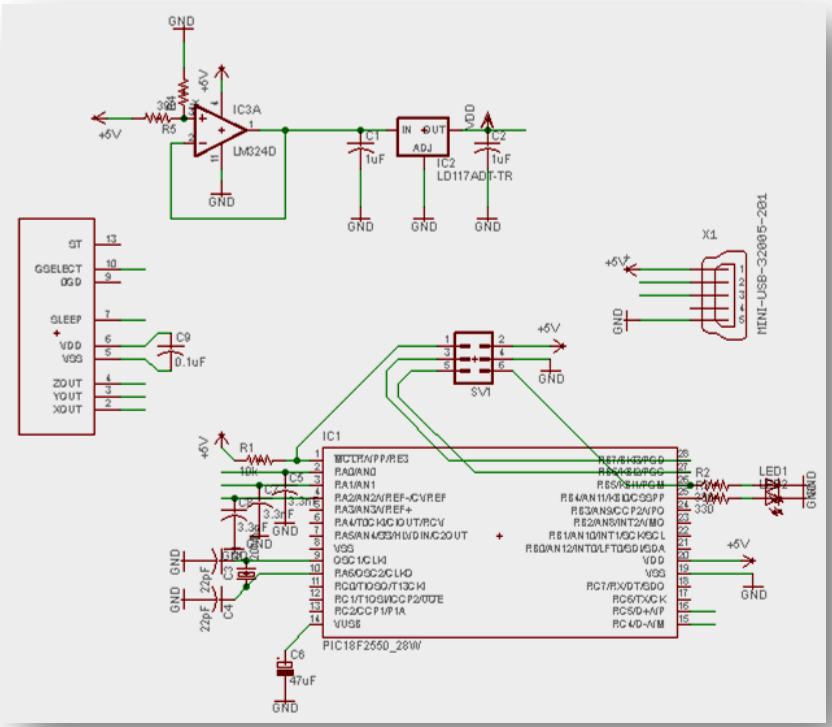


Figura 4.2 Diagrama circuital del dispositivo electrónico.

Para esta aplicación en particular se puede destacar la conexión del sensor la cual se explica a continuación:

El pin de selección de sensibilidad se encuentra conectado al positivo del circuito, para que el dispositivo opere en configuración de 2g lo que nos proporciona una mayor salida proporcional a la posición angular (alta sensibilidad). Las salidas generadas por el acelerómetro se entregan a través de los pines x, y, z, dependiendo de las alteraciones que este experimente. Dichas salidas se conectan directamente a tres entradas analógicas del microcontrolador, que se pueden identificar mediante las etiquetas de AN0, AN1, AN2 respectivamente.

La alimentación del circuito es proporcionada por la fuente del computador al que se conecte a través del puerto USB. Ya que el sensor es de bajo consumo de potencia, opera con voltajes y corrientes reducidas, por lo que se implementó un divisor de voltaje acompañado de un amplificador en configuración de seguidor.

Para efectos de prueba de la tarjeta, se realizó una PCB (Figura 4.3), la cual se creó mediante planos generados en Eagle 5.11.

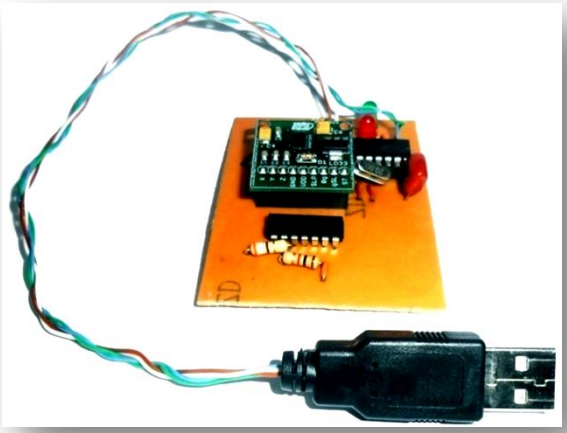


Figura 4.3 Tarjeta para el control externo.

La tarjeta final se diseñó, utilizando dispositivos superficiales, puesto que se necesita un elemento electrónico de un tamaño y peso reducido. El diseño de la PCB doble capa se elaboró con la ayuda del software Eagle 5.11, para posteriormente ser construida por la empresa PCB Tecnologías ubicada en la

ciudad de Cali la cual se dedica a la fabricación de impresos, en la imagen se muestra la tarjeta final, elaborada con dispositivos superficiales (Figura 4.4).

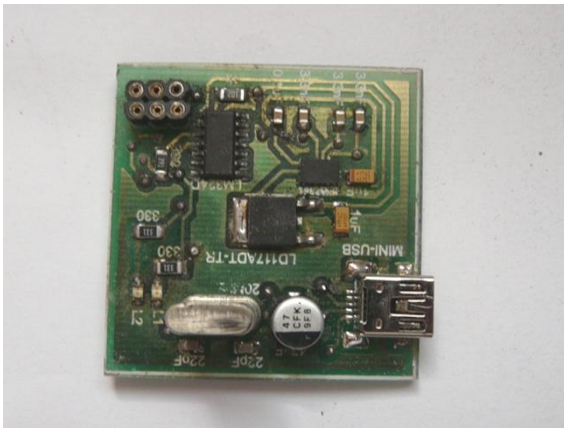


Figura 4.4 Tarjeta final con componentes superficiales.

4.1.3 Diseño del casco o diadema

En la construcción del casco para el dispositivo electrónico se tuvo en cuenta en primera instancia el tamaño de la tarjeta o PCB, debido a que esta se colocará en la parte superior de la diadema, con el propósito de poder orientar la cámara con los movimientos de la cabeza. Otros requisitos importantes que se deben tener presentes en la elaboración de este módulo son:

- Ergonomía: Casco diseñado bajo parámetros básicos que ofrezcan comodidad, seguridad y eficiencia al usuario.

- Liviano: Peso moderado que oscile entre los 250 y 500 gramos aproximadamente.

- Robusto: Casco hecho sobre un material resistente a prueba de impactos.

- Versátil: Adaptable a cualquier forma y tamaño de la cabeza del cirujano.

- Confortable: Diseño implementado con materiales que ofrezcan la mayor comodidad posible.

Para cumplir con las exigencias presentadas, se busco construir un casco con un diseño simple, pero de altas prestaciones. Para ello se utilizaron como base diferentes elementos los cuales se especifican a continuación:

4.1.3.1 Arnés

Es un accesorio que se coloca sobre la cabeza del usuario y permite aferrarse sobre cuatro puntos de la misma; la parte frontal, superior, posterior y lateral, lo cual garantiza una excelente sujeción (ver Figura 4.5). Este arnés está constituido por tres partes:

- **Estructura:** Es específicamente el cuerpo del arnés, elaborado en materiales como el policarbonato o en su defecto por el polietileno de alta resistencia.

- **Sujetador:** Está constituido por pasadores, tuercas y arandelas que permiten la sujeción de la careta o protector visual.
- **Cremallera ajustable:** Constituye el sistema para regular el tamaño del arnés con el fin de ajustarse a la forma de la cabeza del usuario. Este se ubica particularmente sobre la parte superior y posterior de la estructura del arnés.

Figura 4.5 Arnés para casco.

4.1.4 Compartimiento en resina para dispositivo electrónico

Para proteger y albergar de manera segura la PCB o tarjeta electrónica, se realizo un contenedor en resina rígida, siguiendo pautas establecidas para la elaboración y modelado de piezas [46].

En primera instancia se fabricó un modelo base en arcilla (Figura 4.6), con terminación en macilla plástica para cubrir poros e imperfecciones, con el fin obtener un mejor acabado.

Figura 4.6 Molde para PCB en arcilla.

Con el diseño de la carcasa en arcilla ya terminada, se efectuó el correspondiente molde en silicona RT, hecho en dos partes debido a que el cuerpo a reproducir tiene una forma compleja y presenta complicaciones al desmoldar.

Para finalizar con el proceso de fabricación del compartimiento se procedió a adicionar la resina rígida sobre el molde y se dejó curar por un tiempo aproximado de 24 horas, para posteriormente sacarlo, pulirlo y pintarlo.

4.1.5 Detalles finales en la elaboración del casco

Con la carcasa de la tarjeta ya terminada, se ubicó el conector hembra mini USB tipo B en la parte posterior del casco, y se usó un cable plano para poder acoplar este modulo con el ordenador.

Como detalles finales se sujetó la carcasa con tornillos planos al arnés y se colocó una tapa en acrílico para sellar el compartimento. Se utilizó un recubrimiento blanco sobre todo el arnés y se pulió para obtener una terminación óptima para la presentación final del trabajo (ver Figura 4.7).

Figura 4.7 Casco terminado.

La implementación hardware del dispositivo representó cierto costo económico. En la siguiente tabla se especifica el valor de cada uno de los elementos utilizados en la construcción del casco.

Tabla 4.1 Lista de costos.

	Elemento	**Cantidad**	**Costo ($)**
1	PIC18F2550	1	16.800
2	LM324	1	850
3	Regulador 3.3V	1	2.000
4	Acelerómetro MMA7361L	1	7.600
5	Resistencias	9	800

6	Condensadores	13	2.550
7	Leds	2	1.000
8	Terminal Mini USB tipo B	1	2.000
9	PCB	1	26.500
10	Montaje y soldadura		17.400
11	Envío de dispositivos desde Bogotá		7.000
12	Silicona RT	250 ml	50.000
13	Resina rígida	500 ml	15.000
14	Arcilla gris	500 g	1.500
15	Cobalto	50 ml	1.500
16	Macilla de poliescol	150 g	6.000
17	Pintura de poliuretano	75 ml	16.000
18	Lijas (calibre 150, 220, 400 y1200)	2	8.000
19	Arnés de polipropileno	1	18.000
			$ 200.500

4.2 Implementación software

Para el desarrollo de la aplicación embebida se utilizó el lenguaje C (con el compilador CCS), sobre el entorno de programación de Mplab. Básicamente la tarea que debe ejecutar el dispositivo es la de recibir vía USB comandos generados desde un computador y dependiendo del tipo de información recibida, realizar cierta tarea y retornar el resultado por el mismo canal de comunicación (puerto USB).

4.2.1 Protocolo USB (*Universal serial bus*)

Es necesario conocer cómo funciona el protocolo de comunicaciones USB (Figura 4.8) tanto en un ordenador, como en un dispositivo integrado como el que se pretende usar. Existen variantes dentro de este protocolo, que le brindan flexibilidad para ser usado en aplicaciones donde los requerimientos son diversos dependiendo del tipo de transferencia que se desea [47]. Dentro de los tipos de transferencia más comunes encontramos:

- Transferencias de control: se utiliza para transmitir comandos cortos y simples, donde lo que se busca es una entrega de datos garantizada y sin

errores. El tamaño de los paquetes de datos varía dependiendo de la velocidad a la cual se pretende operar

- Transferencias isócronas: son comúnmente utilizadas en aplicaciones donde lo que interesa es el envió constante de información, sin preocuparse tanto por la eventual pérdida de la misma, generalmente para la transferencia de video y audio en tiempo real. Se caracterizan por tener un ancho de banda, pero una latencia limitada.

- Transferencias de interrupción: son transferencias en las cuales los dispositivos no transfieren información de manera continua, pero que cuando lo hacen tienen requerimientos de latencia, se caracterizan por el reenvió de datos en caso de fallo.

- Transferencias Bulk: se usa en transferencias de grandes cantidades de información, donde se garantiza la disponibilidad de toda la banda, pero sin garantizar velocidad en la transmisión.

Para establecer comunicación entre un dispositivo USB y un ordenador, hay que tener en cuenta que se deben establecer canales lógicos unidireccionales a los cuales se les denomina *pipes*, por los cuales va a circular la información y que se conectan desde el controlador del *host* hasta una entidad lógica en el dispositivo llamada *endpoint*.

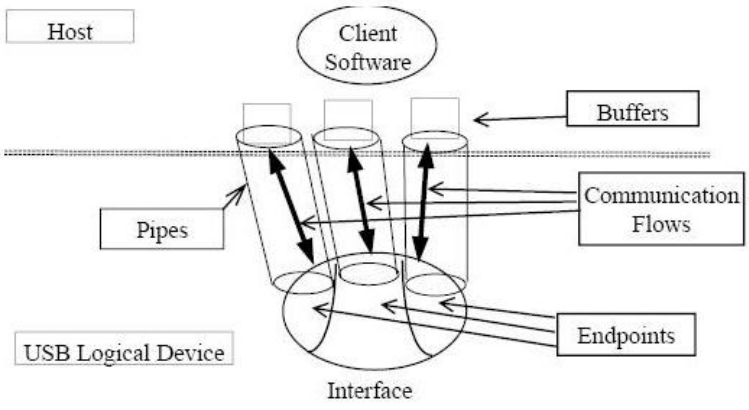


Figura 4.8 Diagrama lógico de comunicación USB [47].

Dependiendo del tipo de transferencia que se va a manejar en la comunicación, se deben definir los *pipes* correspondientes, por ejemplo si se va a establecer una transferencia isócrona, se debe crear un *pipe* de ese tipo [48].

El protocolo USB maneja una estructura semejante a la del modelo OSI pero reducido a tres niveles (Figura 4.9). La capa superior se podría decir que es la relacionada con la parte lógica, por lo tanto es donde podemos llevar a cabo una implementación software de acuerdo con las necesidades de nuestro programa. La capa intermedia se encarga de la gestión de los datos entra el nivel tres y el uno, por lo que se puede decir que se la relacionada con los *drivers* que nos proporciona Microchip y que se instalan dentro del ordenador, mientras que el nivel más bajo es el nivel físico asociado con las conexiones eléctricas.

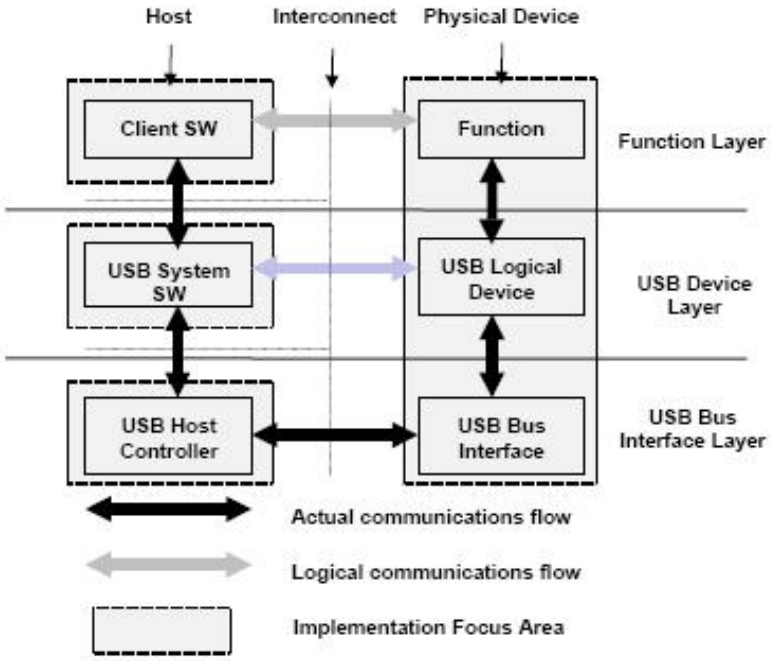


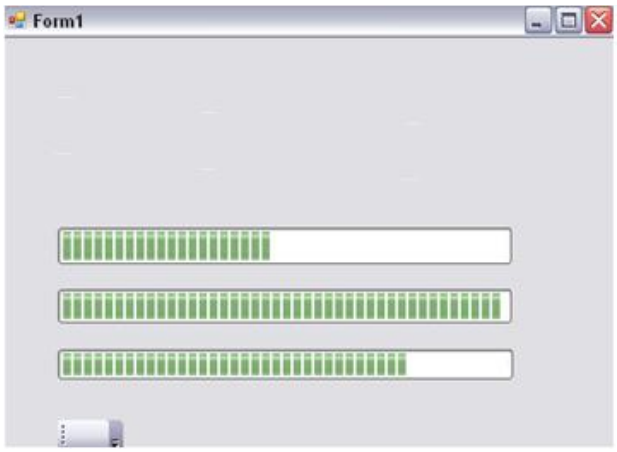
Figura 4.9 Modelo estructural USB [49].

4.2.1.1 Librerías de comunicación USB

Inicialmente se trabajó con la librería de enlace dinámico HIDCLASS.dll, que permitió establecer comunicación entre el dispositivo embebido (PIC18F2550) y un ordenador, utilizando comunicación HID, que es mediante el cual se comunican dispositivos de interfaz humana con el computador tal y como lo son mouse,

teclado , joystick, etc. El experimento consistió en adquirir voltajes por tres canales analógicos y transmitir los resultados obtenidos vía USB al computador, y en este mostrar dichos valores en barras de estado sobre un formulario de Windows (Figura 4.10) diseñado en Visual Studio 2005.

El archivo .dll mencionado anteriormente contiene funciones del lado del host que le permiten realizar configuración, lectura y escritura de datos vía USB, por lo que para el desarrollo de la aplicación fue necesario realizar una vinculación de la librería dinámica con Visual Studio. Ya que la librería está almacenada en un archivo dll único, se debe utilizar la vinculación implícita [50], que consiste en incluirlo directamente dentro de las referencias del proyecto, y guardarlo en la carpeta que contiene el ejecutable generado para que éste funcione de la manera correcta. Así es sencillo hacer el acople entre Visual Studio y la herramienta proporcionada por Microchip.



4.10 Barras de estado en *Windowsform*.

A pesar de que los resultados obtenidos fueron positivos, se presentó un inconveniente que nos impedía trabajar con esta librería bajo una aplicación básica de C++. Se hizo imposible generar el proceso de vinculación del dll mediante el método implícito, que tan solo está disponible cuando se hace uso de ciertas plantillas predeterminadas de Visual Studio 2005 tal y como lo es la de formularios para Windows, problema que haría imposible la posterior vinculación con la librería gráfica que se pretende utilizar para la creación del entorno virtual y el robot. Por esta razón se optó por cambiar la librería de comunicación por una

que presentara mayor flexibilidad con respecto a la integración con los diferentes IDE de programación.

Se encontró otra librería proporcionada por Microchip, que es la MPUSBAPI [51] (librería USB de propósito general) la cual contiene funciones tanto para el host como para el hub. Se trabajó con la última versión disponible, que puede ser obtenida de la página oficial. Esta librería se compone de tres archivos un .dll que al igual que en la anterior, es el que contiene las clases mediante las cuales se pueden llevar a cabo funciones USB de lectura, escritura, etc. Los dos archivos faltantes son un .h y un .c, en los cuales se encuentra todo el código que permite llevar a cabo la vinculación explicita de la librería. De esta forma para poder hacer uso de las funciones y procedimientos de comunicación contenidos en MPUSBAPI se deben incluir dentro del proyecto los archivos .h y .c, y generar una copia del .dll dentro de la carpeta del proyecto. Una de las grandes ventajas de trabajar con MPUSBAPI es que es tal vez una de las librerías más populares en lo referente a aplicaciones USB, por lo que MICROCHIP ofrece mucha información relativa a esta y genera soporte a los usuarios mediante la actualización continua de la misma, tanto así que la última versión que tenemos a nuestro alcance permite trabajar sobre cualquier sistema operativo, incluso el más reciente tal y como lo es Windows 7 tanto en 32 como en 64 bits.

El objetivo de la aplicación es el de adquirir datos vía USB por varios canales analógicos, donde cada uno de estos se encuentra asociado con los valores X, Y, Z entregados por el acelerómetro, y con base en estos datos modificar el valor de la posición articular de cada una de las partes de robot, para que el efector final alcance una nueva posición deseada, para ello se debe establecer una rutina de programación que permita la comunicación entre el dispositivo externo y el computador, de tal forma que se envíen instrucciones que deben ser ejecutadas por el microcontrolador para posteriormente ser retornadas al ordenador, quien se encarga de hacer uso de estos resultados para efectuar el control del robot quirúrgico.

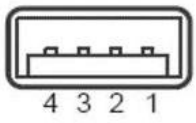
4.2.1.2 Identificadores USB

Antes que nada se debe vincular el dispositivo USB con el ordenador, de tal forma que sea reconocido como un elemento hardware plug and play adicional, y que por lo tanto aparezca dentro de la ventana del administrador de dispositivos. Es de vital importancia la asignación de ciertos identificadores para el dispositivo,

mediante los cuales se haga el proceso de autenticación e identificación por parte de Windows. Estos indicadores reciben el nombre de VID y PID, y se trata de números hexadecimales únicos, que caracterizan cada uno de los productos USB que encontramos en el mercado y que se encuentran tanto registrados como reservados. El VID es el identificador asociado con el fabricante, ya que por lo general las marcas importantes suelen manejar uno solo, mientras que el PID es el identificador del dispositivo como tal. Cuando se realiza el proceso de reconocimiento por parte del sistema operativo, lo primero que se verifica son estos códigos, para que la comunicación con el ordenador no tenga ningún tipo de problema. Si se conectan varios elementos con identificadores idénticos se va a producir un rechazo por parte del ordenador, ya que se van a producir conflictos en la comunicación. Para este proyecto particular se hará uso del VID registrado por Microchip, que puede ser usado por todos aquellos que adquieran sus productos, mientras que el PID será seleccionado por nosotros, teniendo en cuenta también los que se encuentran reservados por esta marca.

4.2.1.3 Conectores USB

Los terminales USB 2.0 se componen de cuatro pines internos a través de los cuales se transmiten tanto los datos, como alimentación proveniente de la fuente del computador. Hay que tener en cuenta que la potencia suministrada a estos tiene que estar dentro de ciertos valores, ya que una carga excesiva podría ocasionar la avería del puerto o la fuente. El modo de trasmisión de los datos es diferencial, por lo que los pines de comunicación son dos, uno positivo y el otro negativo, por lo tanto la conexión del puerto con el microcontrolador es la que se observa en el diagrama 3.5, la cual va a permitir tanto la trasferencia bidireccional de información, como la alimentación directa del dispositivo eliminando así la necesidad de una fuente digital externa (ver Figura 4.11).



1	+5 (Vbus)	Red
2	Data-	White
3	Data+	Green
4	Gnd	Black

Figura 4.11 Terminal USB Tipo A [47].

4.2.2 Desarrollo del programa para el sistema embebido

Como se mencionó anteriormente, la aplicación para el microcontrolador fue desarrollada dentro del IDE de programación MPLAB bajo el lenguaje CCS, el cual no está disponible dentro del paquete predeterminado de MPLAB, por lo cual debe ser instalado adicionalmente en conjunto con un *plugin* para que pueda ser asociado con el IDE. Debe ser configurado como lenguaje de compilación, utilizando la opción *select lenguage toolsuite* (Figura 4.12), que puede ser encontrada dentro del menú Project de MPLAB y buscando la localización donde se produjo la instalación del CCS por parte del usuario.

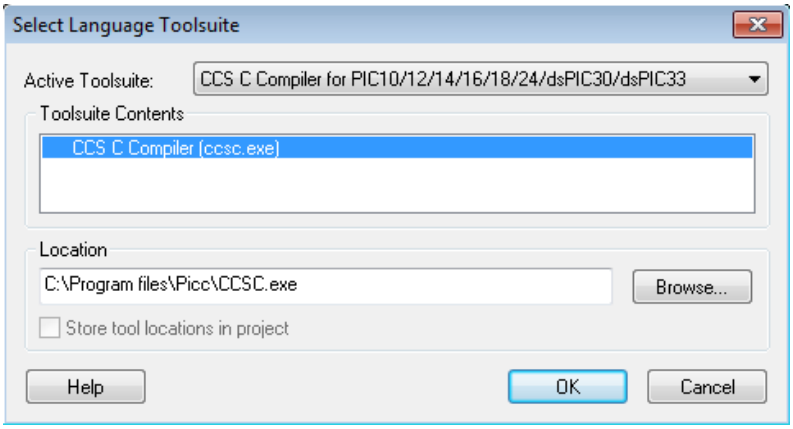


Figura 4.12 Configuración de lenguaje de compilación.

En esta etapa se deben incluir los archivos de cabecera necesarios para el desarrollo del proyecto, los cuales son 18f2550.h (línea 1) que contiene los descriptores asociados con este microcontrolador, pic18_usb.h (línea 2) que contiene los descriptores físicos de comunicación USB para PICs de la familia 18 y la usb.c (línea 4) que contiene los manejadores y los *tokens* de configuración. Todos estos archivos de cabecera son proporcionados por MPLAB y la librería de CCS por lo que basta con incluirlos en el proyecto mediante el comando #include, seguido del nombre del .h a adicionar encerrado entre comillas dobles.

4.2.2.1 Rutina de comunicación

La rutina de comunicación y transmisión de datos se implementa en el archivo fuente .c que recibe el nombre de “Principal”. Se puede observar de forma general

que el proyecto se divide en tres secciones. En la primera encontramos las configuraciones generales y la declaración de variables, la segunda es de establecimiento de comunicación USB con el ordenador y la última en el bucle infinito de comunicación y trasmisión de datos.

Una vez hemos incluido los archivos de cabecera requeridos, configuramos el conversor ADC para que trabaje a 10bits (línea 1), es decir que nos entregue un valor digital entre 0 y 1024 proporcional al valor analógico que tenga como entrada en el pin de lectura (que debe estar entre 0 y 5v). De esta manera se puede trasmitir paquetes USB pequeños a una mayor velocidad y por lo tanto obtener un tiempo de respuesta menor. También se configuran los *fuses* de microcontrolador y el reloj (línea 6), de tal forma que con un cristal externo de 20 MHertz se obtenga a la entrada del procesador del PIC una frecuencia de 4.8 MHrtz, que es la requerida para entablar comunicación USB.

```
1. #device ADC = 10
2. #fuses
   HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV
   1,VREGEN
3. #use delay(clock=4800000)
```

Luego viene la definición de los descriptores USB de acuerdo con el tipo de comunicación que se pretende implementar. Lo primero que se hace es deshabilitar el uso de directivas HID (línea 8), luego habilitamos el *end point 1* (EP1) para entrada y salida de datos de tipo *bulk*, y definimos tamaños de buffer para trasmisión (línea 9) y recepción (línea 10) asociados también a cada uno de los *end points*. Dichos tamaños están dados por paquetes de 8bits.

```
4. #define USB_HID_DEVICE     FALSE
5. #define USB_EP1_TX_ENABLE  USB_ENABLE_BULK
6. #define USB_EP1_RX_ENABLE  USB_ENABLE_BULK
7. #define USB_EP1_TX_SIZE    1
8. #define USB_EP1_RX_SIZE    1
```

Para este caso en particular el tamaño del paquete de salida es de 1, octeto que va a contener el resultado de la conversión ADC por alguno de los canales analógicos que se han definido como tal. El paquete de recepción también va a tener un tamaño de 1 y va a contener la instrucción que le indicará al microcontrolador cuál es el canal por el que debe realizar la conversión ADC el cual va a estar asociado con alguno de los ejes del acelerómetro. Una vez se han definido los parámetros de comunicación se procede a la declaración de variables (líneas 13, 14, 15, 17, 18) y a la inicialización del micro (línea 16), donde se definen como entradas analógicas los primero tres pines de este.

```
9.  #define canal      recibe[0]
10. #define conversion envia[0]
11. int16 rk;
12. void inicializar_micro()
13. int8 recibe[3];
14. int8 envia[1];
```

Dentro del circuito se incluyen leds que nos permiten determinar el estado en el cual se encuentra el dispositivo. Cuando se conecta por primera vez se genera el encendido del led blanco (líneas 19, 20), indicando que la conexión eléctrica es correcta pero que aún no se ha establecido comunicación USB, ya sea porque no se han instalado los controladores del dispositivo o porque existe algún problema con el ordenador o la tarjeta. Cuando los procesos de inicialización de USB, de habilitación de periférico e interrupciones y enumeración por parte del host se realizan de una manera satisfactoria, el led blanco se apaga y se enciende un led azul (líneas 21 a 25). Estos indicadores se encuentran asociados a los pines 4 y 5 del puerto B del microcontrolador.

```
15. output_low(PIN_B4);
16. output_high(PIN_B5);
17. usb_init();
18. usb_task();
19. usb_wait_for_enumeration();
20. output_low(PIN_B5);
21. output_high(PIN_B4);
```

Cuando se ha establecido comunicación USB de una manera correcta comienza el bucle infinito, dentro del cual se produce el proceso de intercambio de datos entre el ordenador y el microcontrolador. Inicialmente el PIC está a la espera de los datos provenientes del PC (línea 26). Una vez llegan datos al *endpoint* de salida (EP1), se realiza la lectura de estos y se almacenan en la variable “recibe” (línea 27).

```
22. if (usb_kbhit(1)) {
23. usb_get_packet(1, recibe, 1);
```

Si los datos recibidos indican que se debe producir la adquisición de datos por el canal 0, se configura el PIC para que esto ocurra (líneas 28, 29), Se produce un retardo de 20uS (línea 30) y se realiza la lectura por dicho canal (línea 31).

```
24. if (canal == 0) {
25. set_adc_channel(0);
26. delay_us(20);
27. rk = read_adc();
```

Una vez finalizada la lectura ADC (línea 32) se almacenan los datos obtenidos en la variable “envía”, y se utiliza el comando de escritura de datos en el EP1 (línea 34), para que puedan ser transferidos vía USB al ordenador.

```
28. if(ADC_done()==1){
29. conversion=rk;
30. usb_put_packet(1, envia, 1, USB_DTS_TOGGLE); }
```

Cuando el dato recibido del ordenador indica que el canal de lectura es el 1, se produce un proceso similar al ya explicado, solo que las configuraciones del

microncontrolador se modifican para generar un cambio en el canal de entrada ADC. Adicional a todo lo anterior se debe asignar tanto un VID como un PID al dispositivo, los cuales pueden ser modificados en el archivo PicUSB.h del proyecto en la sección *start device descriptors*. Ya que no se ha producido el proceso de registrar nuestro producto ante usb.org debido al costo económico, debemos hacer uso del VID que ofrece Microchip.

4.2.3 Desarrollo del programa para el *host* (ordenador)

4.2.3.1 Vinculación de MPUSBAPI.dll con Visual Studio

Del lado de Visual Studio 2005 se debe hacer uso de la librería MPUSBAPI.dll. Para esto debemos añadir los tres archivos que la componen a nuestro proyecto, copiándolos directamente al directorio del mismo, añadiendo el archivo usb2550.h a los archivos de encabezado, incluyendo el archivo usb2550.lib como dependencia adicional (Figura 4.13). Esto se hace haciendo click derecho sobre el proyecto, luego sobre la opción Propiedades donde se desplegará una ventana en la que seleccionamos Vinculador y luego entrada. De esta forma tendremos a nuestra disposición todas las funciones contenidas dentro del archivo dll para poder utilizarlas en nuestro proyecto.

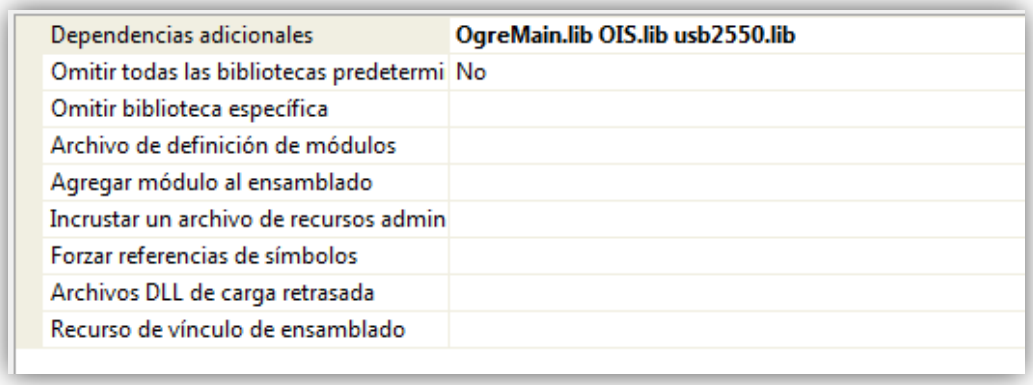


Figura 4.13 Dependencias adicionales de Ogre sobre Visual Studio.

Para que no se produzcan errores durante el proceso de compilación es necesario que se configuren los directorios de búsqueda de Visual Studio, de tal forma que la

carpeta del proyecto forme parte de ellos para que no existan problemas de enlace con los archivos de librería externos. Esto se hace dentro el menú herramientas-opciones-proyectos y soluciones-directorios de VC++ (Figura 4.14), donde se debe buscar la carpeta donde se almacenan cada uno de los archivos ya mencionados, que deben ser configurados de acuerdo con el tipo de archivo del que se trate.

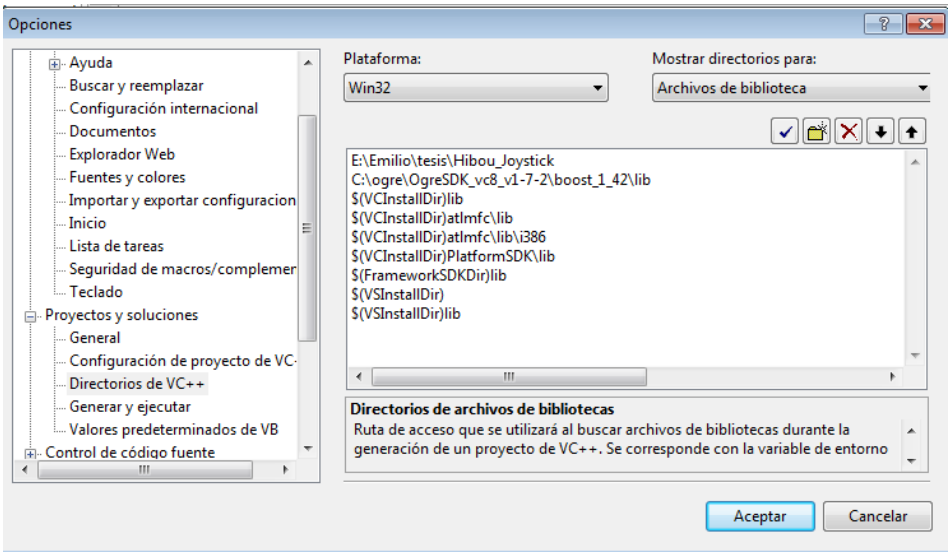


Figura 4.14 Configuración de archivos de biblioteca.

4.2.3.2 Declaraciones y configuraciones iniciales

De manera similar a la que se desarrolló el proyecto para el sistema embebido, es necesario generar la inclusión de los archivos de librería a utilizar (líneas 1 a 3), así como la declaración de las variables y configuraciones iniciales al principio del archivo .c. La librería Windows.h se requiere para poder hacer uso de retardos en la aplicación. Se definen tanto el PID como el VID del dispositivo USB externo con el que se pretende llevar a cabo el proceso de comunicación (línea 4). También se definen los nombres de los *endpoints* asociados a cada uno de los *pipes* de entrada y de salida (líneas 5, 6), así como manejadores (línea 7, 8), variables y parámetros requeridos para la configuración de las funciones del dll (líneas 9 a 18).

```
1.  #include "BasicTutorial4.h"
2.  #include "usb2550.h"
3.  #include "windows.h"

4.  PCHAR vid_pid = "vid_04d8&pid_0011";
5.  PCHAR out_pipe= "\\MCHP_EP1";
6.  PCHAR in_pipe= "\\MCHP_EP1";
7.  HANDLE myOutPipe;
8.  HANDLE myInPipe;
9.  BYTE SendData[1];
10. DWORD SendLength = 1;
11. DWORD SendDelay = 10;
12. DWORD fgSentDataLength;
13. DWORD ReceiveDelay = 10;
14. DWORD ReceiveLength;
15. DWORD ExpectedReceiveLength = 1;
16. BYTE ReceiveData[1];
17. UINT data, data2, data3;
18. int datanum,datanum2, aux=0;
```

Cuando se han definido cada uno de los parámetros requeridos, se puede generar la inicialización de las *pipes* tanto de entrada (línea 19) como de salida (línea 20). Este procedimiento se lleva a cabo solo una vez dentro del bucle ya que una vez se encuentran abiertas las vías de comunicación, no es necesario volver a configurarlas. Es por esto que se encuentran dentro de la función void Hibou_Joystick::createFrameListener(void).

```
19. myOutPipe = MPUSBOpen(0,vid_pid,out_pipe,0,0);
20. myInPipe = MPUSBOpen(0,vid_pid,in_pipe,1,0);
```

4.2.3.3 Bucle de trasmisión y adquisición

Dentro del proyecto se creó una función llamada bool Hibou_Joystick::processUnbufferedInput(const Ogre::FrameEvent& evt), mediante la cual se puede trabajar la entrada de datos sin *buffer*, es decir que se dispara

con un evento generado cuando todos los objetivos de renderizado han terminado sus comandos de renderizado, pero aún no se ha producido la solicitud por parte de la ventana de generar el intercambio de los *buffers*. Por lo tanto es una funcion que se va a ejecutar una vez cada ciclo de *render*, y va a permitir la lectura o escritura sobre los *endpoints* de entrada o salida USB para tener a disposición del programa las varibles obtenidas por la tarjeta de adquisicion de datos. Se le asigna a la posición cero del vector SendData el valor de cero, indicándole al microntrolador que debe iniciar la conversion (línea 21) ADC por el canal analógico AN0, que se encuentra asociado con los datos generados por el acelerómetro de acuerdo con la inclinacion presente con respecto al eje X. Luego se hace uso de la función MPUSBWrite (línea 22) de la librería USB que se esta utilizando, a la cual se le deben pasar los parametros del *endpoint*, un puntero al dato a enviar, una longitud y un tiempo de respuesta.

```
21. SendData[0] = 0;
22. MPUSBWrite(myOutPipe,(void*)SendData,SendLength,&SentDataLength,S
    endDelay);
```

Generamos un retardo de 2 milisegundos (línea 23) que le permitirán al PIC realizar el cambio de canal analógico de lectura y la obtención de un resultado de la conversión ADC, para poder efectuar la lectura del *endpoint* de entrada mediante la función MPUSBRead (línea 24), donde los parámetros de la función son el *endpoint* de entrada, un puntero hacia el vector donde se almacenarán los datos recibidos, el tamaño de dichos paquetes y un tiempo de respuesta. Los paquetes recibidos son almacenados en la posición cero del vector ReceiveData. Luego de esto se produce de nuevo un retardo de dos milisegundos, se asigna el valor de uno a SendData[0] para que el micro realice el cambio de canal y adquiera un nuevo valor ADC pero esta vez sea el que se relaciona con el eje Y del acelerómetro. De esta manera antes de cada renderizado de pantalla se ha obtenido un valor de lectura para cada uno de los ejes del acelerómetro.

```
23. Sleep(2);
24. MPUSBRead(myInPipe, (void*)ReceiveData, ExpectedReceiveLength,
    &ReceiveLength,ReceiveDelay);
```

Ya que el acelerómetro entrega un voltaje de *offset* es nesesasio restar cierto valor a cada uno de los ejes antes de utilizar los datos (líneas 26, 27), de tal forma que cuando la tarjeta se encucuentre en una posición cero con respecto a la inclinación en X y Y, el valor de las varibles datanum y datanum2 sea igualmente de 0.

```
25. data=ReceiveData[0];
26. datanum=(data-50);
27. datanum2=(data2-51);
```

4.2.3.4 Utilización de los datos para control del robot

Las variables obtenidas deben ser escaladas de tal forma que puedan ser utilizadas para cambiar la posición cartesiana deseada del órgano terminal del robot. Para que el programa no tenga en cuenta los datos erróneos que eventualmente pueden estar presentes, se desechan aquellos cambios que no estén dentro del rango de respuesta del sensor (línea 28). Se han programado dos posibles modos de manipulación del robot mediante el dispositivo externo (líneas 29, 35), el primero consiste en controlarlo mediante el modelo matemático del mismo, al cual le son entregadas las variables Xmouse, Ymouse y Zmouse (líneas 30, a 34). El segundo modo consiste en manipular directamente las dos ultimas articulaciones del robot, de tal forma que las otras permanezcan estáticas y se pueda lograr una mejor apreciación y exploracion del interior de la cavidad abdominal (líneas 36, 37).

```
28. if((datanum/flagUSB)<0.009 && (datanum2/flagUSB)<0.009){
29. if(flagOrientacion == 0){
30. Ymouse=Ymouse + (datanum/flagUSB);
31. Xmouse=Xmouse + (datanum2/flagUSB);
32. contI3 = contI3 + 1;
33. fprintf(usbC,"%.6lf %.6lf     %.6lf %.6lf
    %.6lf\n",datanum,datanum2,Xmouse,Ymouse,Zmouse);
34. MGI_model(Xmouse,Ymouse,Zmouse);}
35. if(flagOrientacion == 1){
36. g6 = g6+(datanum/10000);
```

```
37. g7 = g7+(datanum2/10000);
```

4.3 Integración del dispositivo electrónico y el ambiente virtual

Con el software y el hardware implementados se desarrolló la última tarea, que es la integración, en donde se une el dispositivo electrónico y el entorno virtual. Esto se hace mediante la conexión módulo electrónico, a través del puerto USB con el ordenador, el cual reconoce al dispositivo mediante el código anteriormente expuesto y genera un periodo de espera para que la adquisición de datos se pueda concebir. Después de esto el ordenador captura la información enviada por el dispositivo, se procesan los datos convirtiéndolos en positivos y negativos para que el programa pueda interpretarlos y convertirlos mediante un algoritmo en la posición deseada del órgano terminal. Con estos parámetros se hace el llamado al MGI implementado en el proyecto complementario, para que se puedan obtener las posiciones articulares del robot y así permitir su movimiento.

En las imágenes se puede evidenciar el momento en que el control externo del robot se mueve en diferentes direcciones (derecha, izquierda y al frente), tomando como referencia el centro, para con ello demostrar la posición actual del sistema virtual y la posición final que adquiere al moverse el dispositivo. En la Figura 4.15 sobre la parte izquierda, se muestra al usuario con el casco en una posición central, en la parte derecha el sistema virtual con el robot en la posición inicial y la cámara del órgano terminal dirigida hacia el intestino delgado.

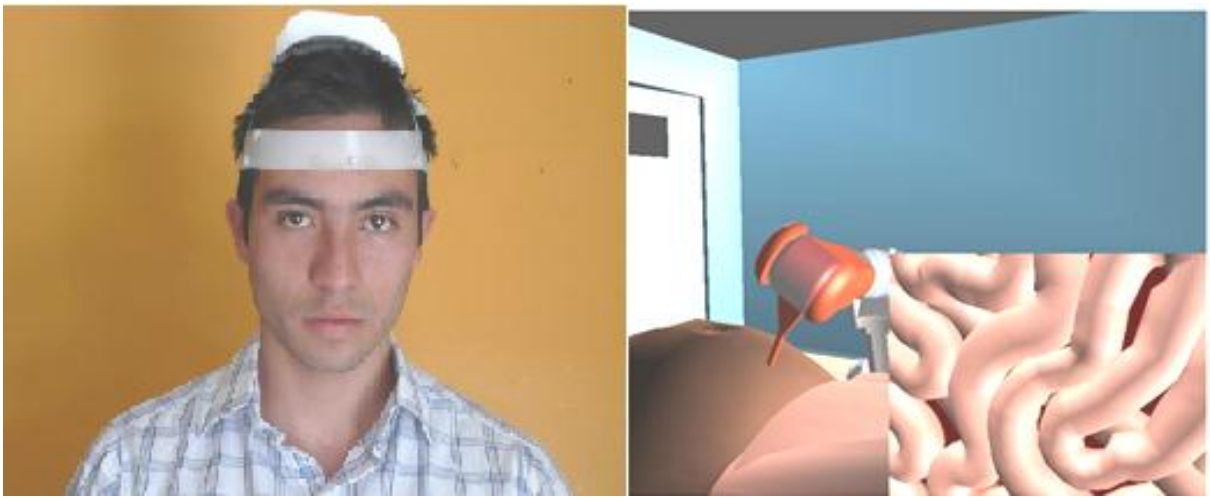


Figura 4.15 Usuario con el casco en orientación central (imagen izquierda) y ambiente virtual con robot Hibou en posición central (imagen derecha).

En la Figura 4.16 se puede evidenciar que el usuario se orienta hacia la izquierda con lo cual hace que sobre el ambiente virtual se vea al robot girado en la misma dirección, y al interior del abdomen se observe como la cámara enmarca la zona superior del intestino grueso, intestino delgado y la vesícula.

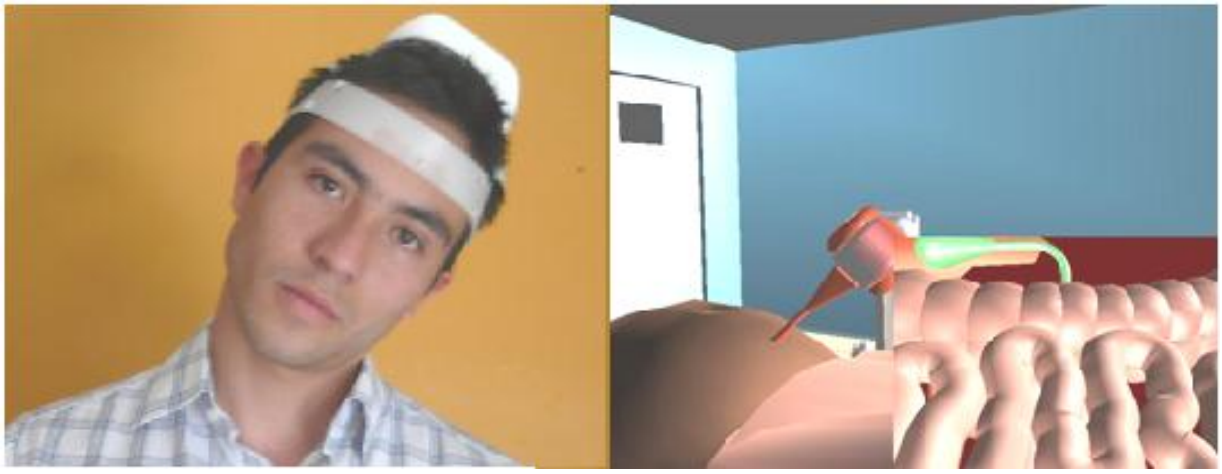


Figura 4.16 Usuario con el casco en posición hacia la izquierda (imagen izquierda) y ambiente virtual con robot Hibou en la misma orientación (imagen derecha).

En la Figura 4.17 se observa al usuario orientado hacia la derecha y en su contra parte se puede ver el sistema virtual guiado hacia la misma dirección. En el interior del abdomen se ve la parte posterior del intestino grueso y el riñón derecho.

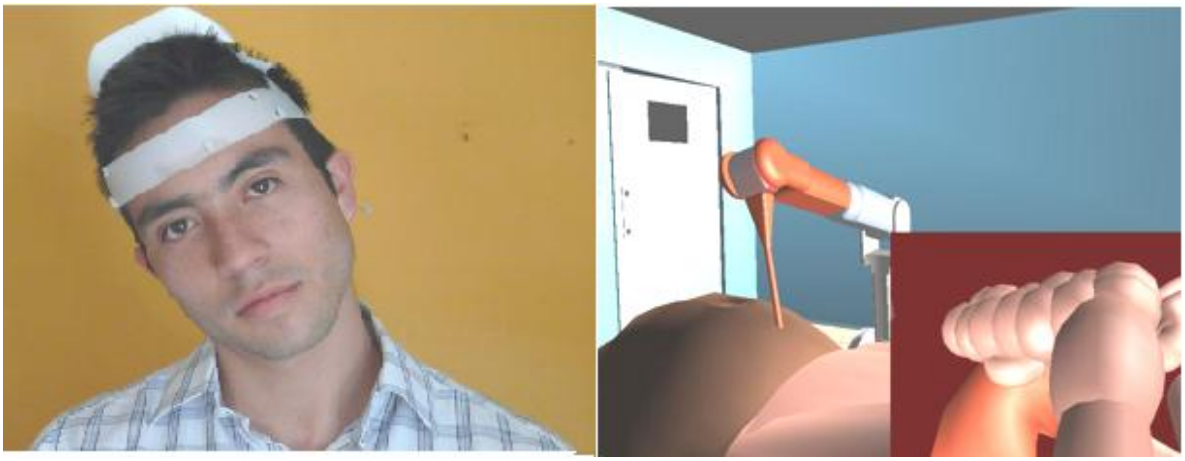


Figura 4.17 Usuario con el casco posicionado hacia la derecha (imagen izquierda) y ambiente virtual con robot Hibou en la misma orientación (imagen derecha).

En la Figura 4.18, se evidencia al usuario orientado hacia el frente. Por tanto en la imagen de la derecha, se observa al robot con la misma orientación del cirujano y la ventana que muestra el interior del abdomen enseña al intestino grueso y al riñón izquierdo.

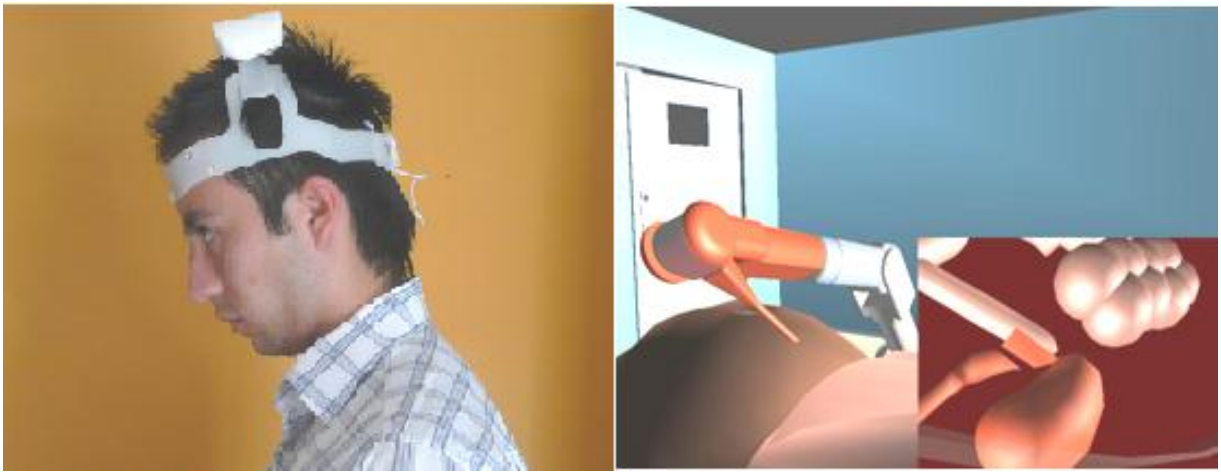


Figura 4.18 Usuario con el casco perfilado hacia el frente (imagen izquierda) y ambiente virtual con robot Hibou en la misma orientación (imagen derecha)

De la misma forma se pueden observar en las figuras 4.19 y 4.20 , los resultados obtenidos cuando se habilita el segundo modo de control donde tan solo se modifican los valores de las posiciones articulares para las dos últimas articulaciones de tal forma que el robot permanece en una posición constante pero se puede tener una mejor navegación por el interior de la cavidad abdominal.

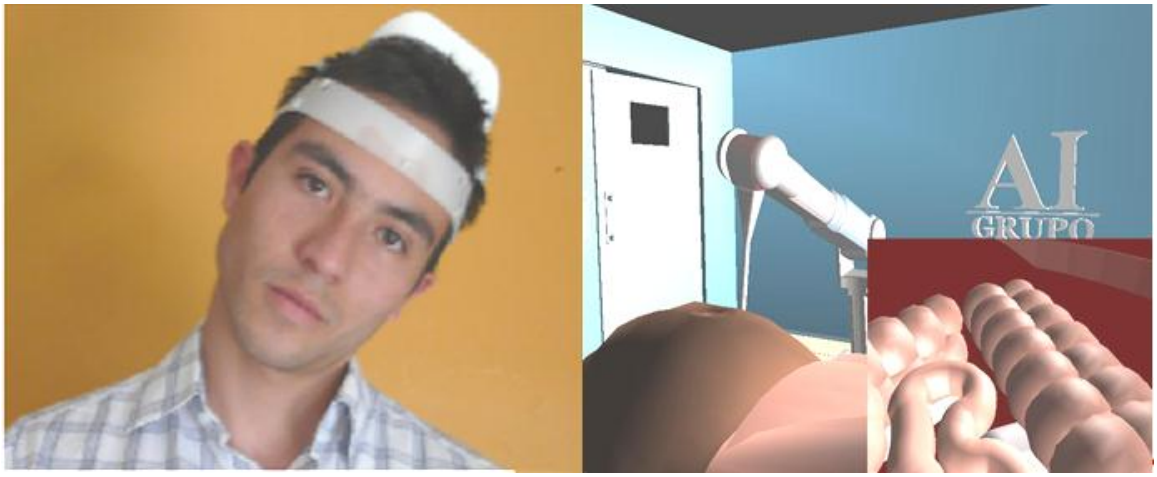


Figura 4.19 Usuario utilizando el segundo modo de operación para generar rotación lateral izquierda.

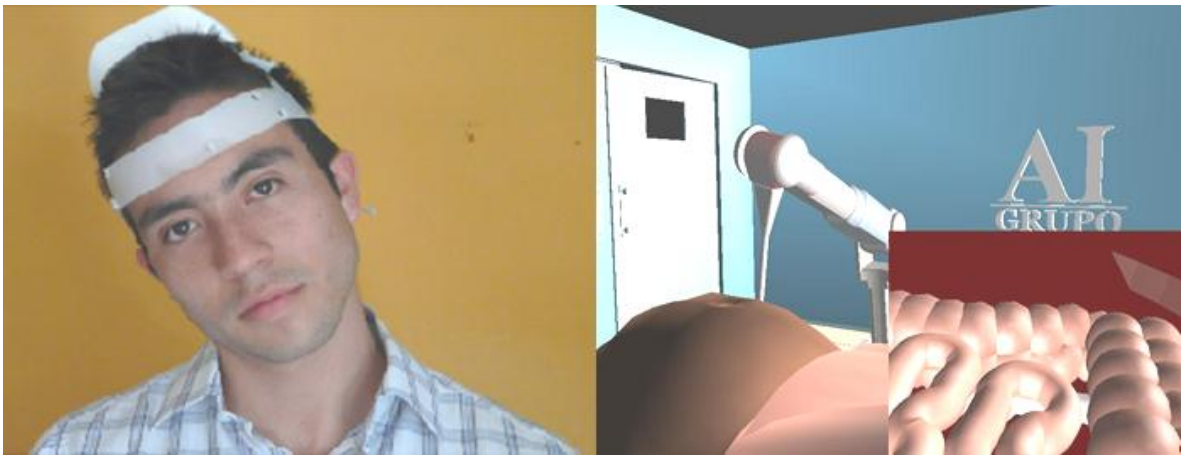


Figura 4.20 Usuario utilizando el segundo modo de operación para generar rotación lateral derecha.

5. Aspectos a mejorar

Dentro del desarrollo del proyecto, existen algunos elementos que pueden ser mejorados, para que la aplicación final tenga un nivel de rendimiento superior, tanto a nivel de comunicación, apariencia del entorno virtual y hardware.

A nivel de comunicación, existen retardos dentro de la aplicación para el host (ordenador) que permiten darle un tiempo de espera al microcontrolador, para que realice la configuración del canal de adquisición de datos y ejecute la lectura por el mismo, estos tiempos influyen de una manera leve en el número de frames por segundo del programa, generando una tendencia hacia la reducción de ellos. Es posible que se pueda explorar más a fondo las librerías USB, para que se genere una rutina de comunicación más óptima donde el PIC genere un mensaje que le indique al PC el momento en el cual ha terminado de realizar su tarea y a cargado los datos sobre los *endpoints* de salida, de tal forma que se minimice dicho tiempo. Para lograr ello, se debe realizar la modificación de los archivos de código fuente, contenidos en el CD incluido en la entrega final, específicamente en los archivos de rutina de comunicación del micro y las líneas de código contenidas en el procedimiento *"UnbufferedInput"* dentro del IDE de programación Visual Studio 2005.

Para que independientemente de la forma en que se ubique el casco en la cabeza del usuario, la aplicación tenga un comportamiento completamente adecuado y se logre una posición centrada dentro del entorno virtual cuando se inicia la aplicación, sería conveniente programar un sistema de auto calibración cuando que se ejecute mediante una tecla programada o al inicio del programa, de tal forma que la posición inicial en la que se encuentra el cirujano corresponda a un estado de quietud del robot y por lo tanto de la cámara.

A nivel de apariencia del entorno virtual, se deben explorar más a fondo las funciones de caracterización y renderizado de los objetos o entidades que se cargan dentro del entorno virtual, para permitir darles características especiales tales como mayor solides a los cuerpos y mejor nivel en cuanto a texturas y colores (esto se puede lograr investigando afondo las opciones que ofrece la herramienta OgreXmlConverter, y los procedimientos o funciones avanzadas que tiene Ogre 3D), permitiendo que se conserve la calidad en

cuanto a forma, material, texturas y color que se tiene obtiene con el software Blender(ver Figura 5.1).

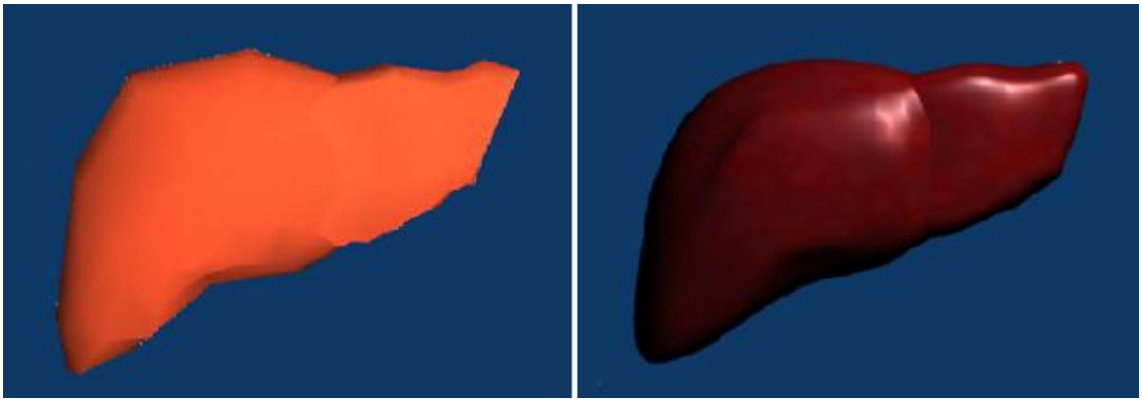


Figura 5.1 Hígado renderizado con Ogre 3D (Imagen izquierda) e Hígado renderizado con Blender (Imagen derecha).

A nivel de hardware se podrían incluir dispositivos trasmisores y receptores inalámbricos de tal forma que el usuario tenga un mayor nivel de comodidad y que al mismo tiempo permita garantizar que los datos lleguen a su destino de una manera rápida y confiable, utilizando tecnologías como *Bluetooth* o *Wireless*. De la misma forma, se requiere un pedal que permita navegar entre los diversos modos de operación que se han programado, procedimiento que actualmente se realiza mediante botones del teclado pre configurados.

6. Conclusiones

En este proyecto se realizó un estudio de las posibles alternativas para el control de un robot porta endoscopio, de tal forma que el médico cirujano pueda hacer uso de sus manos para llevar a cabo cierto procedimiento, y de manera simultánea manipular la ubicación y orientación de la cámara sin la ayuda de una persona. Se optó por hacer uso de los movimientos de la cabeza para generar datos que permitan alterar la posición articular de cada una de las articulaciones del robot Hibou, para que el efector final alcance una nueva posición deseada.

Se decidió utilizar sensores de aceleración para detectar las alteraciones en la inclinación presente, de acuerdo con los movimientos efectuados por el cirujano, debido a que se trata de elementos de medición muy confiables que ofrecen beneficios adicionales con respecto a otros tipos de tecnología, comúnmente usados en aplicaciones robóticas, pero que no han sido utilizados para llevar a cabo el control sobre robots porta endoscopio.

Para generar el proceso de comunicación entre el acelerómetro y el ordenador se desarrolló una tarjeta de adquisición de datos a partir del microcontrolador 18f2550, cuya función es la de convertir los datos analógicos entregados por el sensor, en datos digitales que son trasmitidos vía USB al computador. Por lo cual también fue necesario estudiar lo relativo al funcionamiento de este protocolo de comunicaciones, y las librerías que permiten la integración de sistemas embebidos con aplicaciones programadas bajo el lenguaje C++, tal y como lo son la MPUSBAPI.dll y la HIDCLASS.dll. De la misma forma se estudiaron los métodos de vinculación implícita y explícita que permiten tener a disposición dentro del Visual Studio 2005, las funciones y procedimientos contenidos en dichas clases.

Se desarrolló un entorno virtual del interior de la cavidad abdominal de un paciente, cuyos elementos fueron diseñados en un software para la creación de videojuegos, por lo cual cuenta con un nivel de realismo alto. Finalmente fue posible efectuar el control del robot Hibou dentro de este entorno virtual, a partir de los movimientos de la cabeza del cirujano. La implementación de la matemática relativa a los modelos dinámicos del robot, así como la camilla y el quirófano que hacen parte del entorno, fueros realizados en el proyecto complementario denominado “Software manipulador del robot porta endoscopio Hibou para cirugía laparoscópica”.

Se programaron tres posibles modos de funcionamiento que permiten una exploración más amplia por parte del robot. El primero consiste en el control de este mediante el uso de los modelos geométricos. En el segundo se realiza la manipulación directa de las dos últimas articulaciones y el último permite modificar la altura del robot. Para alternar entre dichos modos de operación, se usa una tecla en el PC, que simula un pedal que le permitiría al cirujano cambiar la forma de mando sin utilizar sus manos.

Como trabajo futuro se tiene planteada la elaboración de un primer prototipo real del robot Hibou, construido con sistemas completamente seguros que permitan garantizar la integridad física de los pacientes y que sea controlado con el dispositivo externo de mando.

Publicaciones en el marco de la tesis

1. “Ambiente virtual y control a distancia de un robot para cirugía laparoscópica” presentado en la *XI Convención de Ingeniería Eléctrica,* Facultad de Ingeniería Eléctrica de Santa Clara, Cuba, Junio 2011.

2. “Sistema virtual para el posicionamiento del robot porta endoscopio Hibou a través de un casco”, presentado en la *Revista Universitaria en Telecomunicaciones, Informática y Control,* Facultad de Ingenieria Electronica y Telecomunicaciones de la Universidad del Cauca, Vol. 1, 2012.

Referencias Bibliográficas

[1] M. Sánchez, M. Rodríguez, S. Bayarri, P Redorta, F. Rodríguez, E. Fernandez Y V. Marvrich, “Historia de la robótica: de Arquitas de Tarento al Robot da Vinci.”, Instituto de Robótica Industrial, Univ. Politécnica de Cataluña. España, 2007.

[2] R. Zamora, “Historia de la Cirugía Endoscópica”, [Online]. Disponible: http://www.anestesia.com.mx/endos.html. Consultado: Noviembre 2011.

[3] D. Camarillo, T. Krummel Y K. Salisbury, “Robotic Technology in Surgery: Past, Present, and Future”, *The American Journal of Surgery*, Vol. 188, pp. 2-14, 2004.

[4] S. Krut, “Introduction to Medical Robotics”, VI Seminario de Automática, Univesidad del Cauca, Colombia, 2006.

[5] A. Cordoba Y G. Ballantyne, “Sistemas quirúrgicos robóticos y telerobóticos para cirugía abdominal”. Revista de Gastroenterología del Perú, Vol. 23, 2003

[6] J. Mosso, A. Minor, V. Lara, Y E. Maya, “Brazo robótico para sujetar y posicionar laparoscópicos. Primer diseño y construcción en México”, *Academia Mexicana de cirugía*, Vol. 69, No 6, pp 295-299, 2001.

[7] S. Mejía, A. Arana, C. Arango, V. Pérez, H. Posada Y A. Torres, “Kirubot: Brazo Robótico Ayudante en Cirugía”, Revista IATREIA, Vol. 14, No. 4, pp. 254, 2001.

[8] A. Jaramillo, Grupo de Automática y Robótica, Pontificia Universidad Javeriana de Cali, [Online]. Disponible: http://gar.puj.edu.co/Investigacion/ Grupos/GAR/TmpGar. Consultado: Noviembre de 2011.

[9] C. Cuellar, E. Moreno, C. Casas, J. de Francisco, H. Ordoñez Y R. Riveros, “Colecistectomía Laparoscópica: primera experiencia en Colombia”, Revista Colombiana de Cirugía, vol. 6, pp. 5-12, 1991.

[10] S. Schwartz and J. Hunter**,** Principios de cirugía, 7ªEd. México: McGrawHill Interamericana, 1999**.**

[11] A. Carbajal, “Cirugía laparoscópica”, [Online]. Disponible: http://www.ciberhabitat.gob.mx/hospital/laparoscopia/index.html. Consultado: Mayo 2011.

[12] Dimeda Instruments, “Dimeda Surgical Instruments” [Online]. Disponible: http://www.dimeda.de. Consultado: Febrero 2011.

[13] M. Meinero, G. Melotti Y Ph. Mouret, “Cirugía laparoscópica”. Buenos Aires: Editorial Médica Panamericana, 1996.

[14] UT Medical Group, “Colecistectomía”, [Online]. Disponible: http: //www.utmedicalgroup.com. Consultado: Febrero 2011.

[15] J. Reyes, “Cirugía del aparato digestivo y laparoscopía: Procedimientos” [Online]. Disponible: http://www.doctorjesusreyes.com/procedimientos.html. Consultado: Junio 2011.

[16] A. Barrientos, “Robótica en medicina”, Universidad Politecnica de Madrid, 2008.

[17] J. Góngora, “Cirugía de laparoscopía avanzada” [Online]. Disponible: http://www.monografias.com/trabajos34/cirugia-robotica/cirugia-robotica.shtml. Consultado: Junio 2011.

[18] A. Carbajal, “Cirugía robótica”, 2006 [Online]. Disponible: http://www.inegi.gob.mx/inegi/contenidos/espanol/ciberhabitat/hospital/robotica/index.html. Consultado Junio 2011.

[19] Latin Salud, “Hombre vs La maquina”, *Latín Salud*, 2010, [Online]. Disponible: http://www.latinsaude.com/articulos/00625.asp. Consultado: Mayo 2010.

[20] J. Acuña, “Nueva era de la cirugía”, *Hospital Universitario del Caribe*, 2008, [Online]. Disponible: http://www.hucaribe.gov.co/novedades/julio_08/ nueva_era_cirujia.pdf. Consultado: Enero 20011.

[21] Cirugía On-line, “Navegación diagnóstica, desarrollará cirugía robótica mínimamente invasiva”, *Cirugía On-Line,* Jun. 16, 2008 [Online]. Disponible: http://medicyafac.wordpress.com**.** Consultado: Enero 2011.

[22] S. Kommu, P. Rimington, C. Anderson, y A. Rane “Initial experience with the EndoAssist camera-holding robot in laparoscopic urological surgery”, *Journal of Robotic Surgery*, Vol.1, pp.133-137, 2007.

[23] G. Morel Y P. Poignet**,** “Kinematics, position and force control issue in minimally invasive surgery”, *International conference on computerized medical imaging and computer assisted intervention,* Francia 2004.

[24] Medsys, “Technology”, *Medsys,* Enero, 2010, [Online], Disponible: http: //www.medsys.be/surgical-robots/lapman/technology.asp. Consultado: Septiembre 2009.

[25] S. Mejía, A. Arana, C. Arango, V. Pérez, H. Posada, Y A. Torres, “Kirubot: brazo robótico ayudante en cirugía”, Vol.4, No.4, 2001.

[26] S. Salinas y O. Vivas, “Simulación 3D de movimientos quirúrgicos de una colecistectomía asistida por el robot LapBot”, *Memorias VIII Congreso de la Asociación Colombiana de automática*, 2009.

[27] C. méndez y V. Torres, “Diseño y simulación en 3D de un robot porta endoscopio para cirugía laparoscópica”. Tesis pregrado Ingeniería en Automática Industrial, Universidad del Cauca, 2010.

[28] W. Khalil and E. Dombre, “Modeling, identification and control of robots”. London. Kogan Page Science. 2002.

[29] R. Paul, “Robot Manipulators: Mathematics, programming and control”. Cambrigde. MIT Press. 1982.

[30] D. Navarro, L. Ríos Y H. Parra, “Sensores de ultrasonido usados en robótica móvil para la medición de distancias” Scientia Et Technica, Universidad Tecnológica de Pereira, Vol. 10, No. 25, pp. 35-40, 2004.

[31] Blog Spot, “Sensores y actuadores”, [Online]. Disponible: http://sensors-actuators-info.blogspot.com/2009/08/laser-triangulation-sensor.html. Consultado: Agosto 2011

[32] Robótica Aplicada, “Sensores de distancia”, [Online]. Disponible: http://www.slideshare.net/diego5wh/sensores-de-distancia. Consultado: Agosto 2011.

[33] Maginvent, “Acelerometros”, [Online]. Disponible: http:// www.maginvent.org/articles/sensorartht/Acelerometros.html. Consultado: Agosto 2011.

[34] Freescale Semiconductor, “±1.5g - 6g Three Axis Low-g Micromachined Accelerometer”. ”, [Online]. Disponible: http:// www.freescale.com/files/sensors/doc/data_sheet/MMA7260QT.pdf. Consultado: Septiembre 2011.

[35] STMicroelectronics, “Low drop fixed and adjustable positive voltage regulators”, [Online]. Disponible: http://pdf1.alldatasheet.com/datasheet-pdf/view/ 94428/STMICROELECTRONICS/LD1117D33.html. Consultado: Noviembre 2011.

[36] Eagle, “Eagle PCB software”, [Online]. Disponible: http://www.cadsoftusa.com/. Consultado: Septiembre de 2011.

[37] H. Roncancio, “Tutorial de LabView”, Universidad Distrital Francisco José de Caldas, 2001.

[38] National Instruments, “NI USB-6008”, [Online]. Disponible: http://sine.ni.com/nips/cds/view/p/lang/en/nid/201986. Consultado: Septiembre 2011.

[39] National Semiconductor, “Low power quad operational amplifiers”, [Online]. Disponible: http://www.ti.com/lit/ds/snosc16b/snosc16b.pdf. Consultado: Septiembre 2011.

[40] O. Almonacid, “Simulación digital de tráfico para intersecciones señalizadas por semáforo, bajo Ambiente tridimensional”. Tesis de pregrado, Universidad del Bío-Bío, Chile. 2007.

[41] “Ogre 3D”,[Online].Disponible: http:// www.ogre3d.org. Consultado: Agosto de 2011.

[42] Visual Studio, “Introducing Visual Studio”, [Online]. Disponible: http://msdn.microsoft.com/es-es/library/fx6bk1f4.aspx Consultado: Septiembre 2011.

[43] Solid Edge Velocity Series,” Tutorial Solid Edge ST3 Modo Tradicional”, [Online]. Disponible: http://www.solid-edge.es/web/tutoriales. Consultado: Octubre 2011.

[44] Blender Wiki, “Blender Manual”, [Online]. Disponible: http:// wiki.blender.org/index.php/Doc:Manual/Introduction. Consultado: Octubre 2011.

[45] Bilbao Electronics, “Datasheet PIC18F2550 Technical Information”, [Online]. Disponible: http://www.bilbaoelectronics.com/datasheetpic18f2550 Consultado: Octubre 2011.

[46] F. González, “Como hacer moldes silicona”, [Online]. Disponible: http://es.scribd.com/doc/32829394/Como-Hacer-Moldes-de-Silicona. Consultado: Octubre 2011.

[47] E. López, “El Protocolo USB“. [Online]. Disponible: http://www.imicro. com/pdf/articulos/usb.pdf. Consultado: Octubre 2011.

[48] Soporte Técnico OEM, “Un paseo por USB 2.0“. [Online]. Disponible: http://www.fujitsu.com/downloads/EU/es/soporte/discosduros/UnpaseoporUSU-2.pdf. Consultado: Noviembre 2011.

[49] Planet Carsoft, “USB (Universal Serial Bus)”. [Online]. Disponible: http://www.forpas.us.es/aula/hardware/dia2_USB.pdf. Consultado: Octubre 2011.

[50] Microsoft, “Vincular un ejecutable a un archivo DLL”. [Online]. Disponible: http://msdn.microsoft.com/es-es/library/9yd93633.aspx. Consultado: Septiembre 2011.

[51] Slalen, “Librería del USB: MPUSBAPI.DLL”. [Online]. Disponible: http://slalen.ifastnet.com/programacion/datos/MPUSBApi.pdf. Consultado: Noviembre de 2011.

Anexo A. Componentes principales de Ogre 3D

La siguiente información está basada en el manual de Ogre, que se encuentra disponible en www.ogre3d.org.

A1. Funciones principales.

- **Gestión de escenario (*Scene Management*)**

Esta clase brinda información del contenido del escenario, como por ejemplo, la forma en que está estructurado, la posición de la cámara, luces, etc. Los objetos pertenecientes a esta área son los comprometidos con el desarrollo de una interfaz declarativa para el mundo que se desea crear, lo que quiere decir que Ogre se encarga de posicionar los objetos en el lugar deseado y el material del cual están constituidos.

- **Gestion de recursos(*Resoource Management*)**

Esta función se encarga de administrar los recursos indispensables para crear el entorno virtual que se quiera, lo que quiere decir que es aquí en donde se estable la geometría, las texturas, las fuentes necesarias para el renderizado.

- **Renderización (*Rendering*)**

Como última tarea se deben poner las imágenes en la pantalla del ordenador, esta etapa es el nivel más bajo del canal de renderizado. Los buffers son los que renderizan los estados y el producto, pero Ogre utiliza un sin número de *plugins*, lo cual hace que el programa pueda ser ampliado y gran parte de las clases se puedan heredar.

A2 Objetos principales

- **Objeto raíz (*Root*)**

Este debe ser el primer objeto en crearse y el último en destruirse. El objeto raíz ayuda a configurar el sistema, por ejemplo para el renderizado puede utilizarse *ShowConfigDialog(),* el cual genera un cuadro de dialogo para personalizar, la resolución, el color, la profundidad, etc. Así mismo se establece las opciones que el usuario seleccione para inicializar el sistema.

El *Root* también sirve como procedimiento para apuntar hacia otros objetos del sistema como: *SceneManager, RenderSystem* y otros administradores de recursos.

- **Objeto *RenderSystem***

Esta es una clase abstracta que precisa la interfaz más baja de la API 3D. Esta es la encargada de enviar las operaciones de renderizado a la API y de establecer todas las opciones de renderizado. Se denomina abstracta porque toda la implementación es específica para cada representación de la API y además existen diferentes subclases que corresponden a las diferentes funciones de la API.

- **Objeto *SceneManager***

Este es un uno de los puntos más importantes del sistema desde el punto de vista de la aplicación. El Scenemanager está a cargo de los contenidos de cada una de las escenas que se renderizan en el motor gráfico. Es el responsable de organizar el contenido de las escenas, como el crear y administrar las cámaras, luces, objetos móviles y los materiales de los mismos.

- **Objeto *ResourceGruopManager***

Esta clase es un concentrador de carga de recursos reutilizables, como las texturas y las mallas. Es el área en donde se establecen los grupos de recursos, que pueden ser cargados o descargados. El *ResourceGroupManager* puede gestionar diferentes tipos de recursos como *TextureManager* o *MeshManager,* garantizando que estos se carguen una solo vez sobre el motor de Ogre,

asimismo, optimiza la utilización de la memoria con el fin de ocupar el menor espacio posible.

- **Objeto *Mesh***

El objeto de malla (*Mesh*) representa un conjunto geométrico autónomo de tamaño reducido. Estos objetos suelen desempeñarse como cuerpos móviles de geometrías pequeñas.

Las mallas son un tipo de recurso, y son utilizados por el administrador de recursos *MeshManager*, estos son cargados regularmente desde archivos propios de Ogre con formato '*.mesh*' este tipo de archivos por lo general son creados por algún software de modelado en 3D como Blender, Solid Edge ®, etc.

A.3 Entidades

Una entidad es una instancia de un objeto móvil en la escena. Podría ser un vehiculo, una persona, una mascota, etc. Las entidades están basadas en mallas discretas, es decir, es el conjunto de mallas (*Mesh*) o colección de geometrías autónomas.

Para crear una entidad se debe llamar al método *SceneManager::createEntity*, asignándole un nombre especifico a la entidad y el nombre de la malla que se le asignará.

Las entidades no se consideran parte de la escena hasta que se adjuntan a un *SceneNode*. Adjuntando las entidades a *Scenenodes*, se puede crear complejas relaciones jerárquicas entre las posiciones y orientaciones de las entidades.

A.4 Materiales

Este objeto es el encargado de controlar como se renderízan los cuerpos de cada escena, es decir que permite darle solides a las mallas desarrollas en el entorno.

Ésta específica que propiedades de superficie base tienen los objetos tales como, la reflectancia de los colores, brillo, como están las capas de textura, que

imágenes pertenecen a la escena y la forma en que se relacionan, que efectos especiales se aplican según el medio, etc.

Los materiales se pueden configurar mediante programación, mediante el comando *SceneManager::createMaterial* y ajustar la configuración, o se puede cargar de un "*script*" cada vez que se ejecute el programa.

A5. Scripts

Los scripts son archivos de texto plano que pueden ser editados en cualquier editor de texto estándar como el block de notas, las modificaciones realizadas sobre el código tienen un efecto inmediato sobre las aplicaciones de Ogre, sin necesidad de recompilar. A continuación se presenta el elemento sobre los que se pueden hacer *scripts*:

- **Scripts de materiales**

Los *scripts* de materiales permiten definir cualquier tipo de material complejo en *scripts* que pueden ser reutilizados fácilmente. Aunque también se pueden colocar todos los materiales de una escena en el código del proyecto, utilizando los métodos de las clases Material y *TextureLayer*, sin embargo no es muy recomendable debido a que en la práctica es un poco difícil de manejar, por esta razón Ogre utiliza una forma particular de cargar los materiales, mediante la inicialización de los grupos de recursos, con lo cual se procede a buscar en todos los recursos asociados con el grupo para archivos con extensión '.*material* 'y los analiza. Un último factor importante es tener claro que el nombre de los materiales debe ser único en todos los *scripts* cargados por el sistema, ya que estos son siempre llamados por su nombre. Dentro de los *scripts* de materiales existen diferentes secciones estas son:

a. Técnicas

Técnica (*Technique*) en el *script* de material enmarca un proceso único de renderizado de un objeto. La manera más fácil de definir el material solo necesita una técnica, considerando que el hardware del ordenador varia considerablemente en sus capacidades, no obstante esto solo se debe hacer si se tiene la certeza de que todas las tarjetas que usarán la aplicación va a soportar el potencial que la técnica requiere.

Cuando un material se utiliza por primera vez, este es compilado, lo que involucra un análisis de las técnicas que se han definido para determinar si estas son soportables usando la actual API (*Application Programming Interface*) de renderizado y la correspondiente tarjeta gráfica. Si no existen técnicas sostenibles, el material se renderizará en blanco, por otra parte en la compilación se analizan diferentes series de cosas como:

1. Número de entradas *texture_unit* en cada paso.
2. Programas de vértice, geometría o fragmento.
3. Efectos de mapeo.
4. Verificar si el vendedor o el nombre del dispositivo de la tarjeta gráfica actual coincide con algunas normas específicas por el usuario.

En los *scripts* de materiales, las técnicas deben ser enumeradas en orden de preferencia, es decir que las técnicas anteriores son predilectas a las técnicas siguientes. Esto indica que se deben citar las más avanzadas y las más exigentes en el *script*, y las listas menos exigentes después.

b. Pasadas

Pasada es una renderización simple de geometría en cuestión, esto quiere decir que solo necesita una llamada de la API de renderización con un conjunto determinado de propiedades. Una técnica puede tener entre 1 y 16 pasadas, aunque se evidencia claramente que entre más pasadas se utilicen el costo computacional será mucho mayor.

Para facilitar la identificación de cada pasada, estas pueden ser nombradas aunque esto es opcional, para las que no tengan nombres en el script se llaman por el número de índice, por ejemplo la primera pasada en una técnica tiene índice 0 por lo su nombre seria “0”. El nombre debe ser único dentro de la técnica, lo cual garantiza un ejecución correcta, aunque si no es así Ogre mostrara solo un mensaje de advertencia, lo que no afectara la compilación del proyecto, el nombrar las pasadas pueden facilitar la herencia de un material y la modificación de una pasada existente.

Las pasadas tienen un conjunto de pasadas y de atributos globales, cero o mas entradas anidadas *texture-unit*, y opcionalmente, una referencia a un programa de vértice y/o fragmento.

Estos son los atributos que se pueden utilizar en una sección de “pasada” de un *script* ‘*.material*’:

1. *Ambient*
2. *Diffuse*
3. *Specular*
4. *Emissive*
5. *Scene blend*
6. *Separate scene blend*
7. *Scene blend op*
8. *Separate scene blend op*
9. *Depth check*
10. *Depth write*

c. Unidades de textura

Estos son los atributos que puede utilizar una sección *texture_unit* de un *script* ‘*.material*’:

1. *Texture_alias*
2. *Texture*
3. *Anim_texture*
4. *Cubic_texture*
5. *Tex_coord_set*
6. *Tex_address_mode*
7. *Tex_border_colour*
8. *Filtering*

d. Declaración de programas de vértices/geometrías/fragmentos

Para declarar un programa de vértices, geometría o fragmento en la unidad de materiales, es preciso definirlo en la sección de pasada de un material, con lo cual el programa se podrá utilizar para cualquier número de materiales. Este código puede ser embebido en el *script* ‘*.material*’, o si es el caso de que se quiera trabajar con múltiples archivos ‘*.material*’, se debe definir un archivo de *script* ‘*.program*’ debido a que los archivos definidos como éste, se analizan antes que todos los *scripts* ‘*.material*’.

Los programas de vértices, geometría y fragmentos tienen dos clasificaciones; de bajo nivel y de alto nivel, la primera hace referencia al lenguaje ensamblador y la segunda maneja lenguaje DirectX9 HLSL, Open GL Shader Languaje o Lenguaje Cg de nVidia. Los lenguajes de alto nivel aportan mayores ventajas que los de bajo nivel, debido a que el código es más intuitivo y da la posibilidad de usar múltiples arquitecturas en un mismo programa, es decir que un programa de nivel superior escrito bajo lenguaje Cg puede ser utilizado tanto en D3D como GL, en cambio el código de nivel inferior debe utilizar técnicas independientes, cada una de ellas para una API diferente.

Anexo B. Blender

B.1 Características principales

• Paquete de creación totalmente integrado, ofreciendo un amplio rango de herramientas esenciales para la creación de contenido 3D, incluyendo modelado, mapeado UV, texturizado, *rigging*, *weighting*, animación, simulación de partículas y otros, *scripting*, renderizado, composición, post-producción y creación de juegos.

• Multiplataforma, con una interfaz unificada para todas las plataformas basada en OpenGL, listo para ser usado en todas las versiones de Windows (98, NT, 2000, XP, Vista, Windows 7), Linux, OSX, FreeBSD, Irix y Sun, y otros sistemas operativos.

• Arquitectura 3D de alta calidad permitiendo un rápido y eficiente desarrollo.

• Tamaño pequeño del ejecutable para una fácil distribución.

B2. Interfaz

Tiene una muy peculiar interfaz gráfica de usuario, que se critica como poco intuitiva, pues no se basa en el sistema clásico de ventanas, aunque cabe destacar la configuración personalizada de la distribución de los menús y vistas de cámara. En términos generales la interfaz de usuario es el mecanismo de interacción mutua entre el usuario y el programa. El usuario se comunica con el programa mediante el teclado y el ratón, el programa responde por medio de lo que muestra en pantalla.

- **El Teclado y el ratón**

La interfaz de Blender saca provecho de los ratones de tres botones y una amplia gama de atajos de teclado. Si el ratón sólo tiene dos botones, es posible emular el botón central. Es posible usar un ratón con rueda, pero no es obligatorio, ya que también existen atajos de teclado que cumplen la misma función. En Blender se asumen las siguientes convenciones como entradas de usuario:

1. Los botones del ratón se abrevian como BIR (botón izquierdo del ratón), BMR (botón medio del ratón) y BDR (botón derecho del ratón).

2. Si el ratón tiene una rueda, BMR se refiere a hacer click con la rueda como si ésta fuera un botón, mientras que RR significa girar la rueda.

3. Las letras de los atajos se nombran añadiendo TECLA a la letra, es decir TECLAG se refiere a la letra g en el teclado. Las teclas pueden ser combinadas con los modificadores SHIFT, CTRL y/o ALT. Generalmente para las teclas modificadas el sufijo TECLA se descarta, por ejemplo CTRL-W o SHIFT-ALT-A.

4. NUM0 a NUM9, NUM+ y así sucesivamente, se refiere a las teclas ubicadas en el teclado numérico. Generalmente NumLock debería ser activada.

5. En el caso de otras teclas, se les refiere usando sus nombres, tal como ESC, TAB, F1 a F12.

6. Otras teclas especiales para tener en cuenta son las teclas de dirección, ARRIBA, ABAJO, IZQUIERDA y DERECHA.

- **Sistema de Ventanas**

La escena por defecto de Blender, muestra la vista de pantalla, con el menú principal en la parte superior, la cual contiene las pestañas *File, Add, TimeLine, Game, Render y Help,* en el medio se ubica la zona de trabajo para el modelado en 3D, esta se destaca por tener en el centro, un objeto en tres dimensiones, que por defecto es un cubo, un cursor en forma de cruz, la Luz y la cámara, sobre la zona inferior se encuentra el panel de herramientas, que se compone principalmente por pestañas y botones que sirven para editar el objeto que se está modelando, dándole color, texturas y formas .

1 Anexo C. Tutoriales.

INSTALACION DE OGRE3D PARA VISUAL STUDIO

Para instalar el motor gráfico de Ogre3D v1.7.2, se sigue de los siguientes pasos.

- **Instalar Visual Studio 2005:** Actualizar al Service Pack 1 para Microsoft® Visual Studio® 2005 Standard, Professional y Team Editions, y luego Instalar el service pack 1 para Windows Vista encontrados en los siguientes enlaces.

  - ✓ www.microsoft.com/downloads/es-es/details.aspx?FamilyID=bb4a75ab-e2d4-4c96-b39d-37baf6b5b1dc
  - ✓ www.microsoft.com/downloads/es-es/details.aspx?familyid=90e2942d-3ad1-4873-a2ee-4acc0aace5b6&displaylang=es

  Tener en cuenta que ambas versiones deben estar en el idioma del Visual Studio.

- **Descargar el paquete:** dirigirse a la página oficial de Ogre3D. Estando ahí ir al enlace *download* y luego a *SDK (Software Development Kit)*, ahí se encuentran todas las versiones de Ogre incluyendo la versión más actual (v1.7.3). Descargar la v1.7.2 para Visual C++ 2005 de 32 bits.

- **Descomprimir el paquete en una dirección que no sea muy difícil de encontrar en el equipo**, es recomendable que se ubique en el DISCO C.

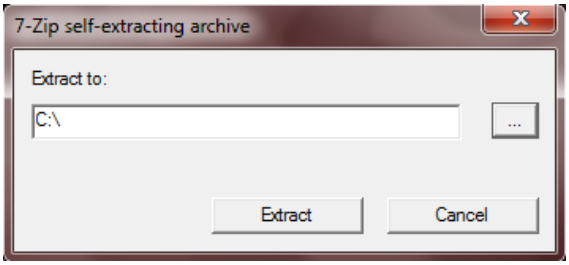


Figura C.C1. Ventana para extraer el paquete de Ogre

- **Descargar el asistente de instalación de Ogre:** Las versiones más actualizadas de Ogre tienen asistentes que direccionan a Visual Studio hacia los dll´s necesarios para la compilación ya sea en *Debug* o en *Release.* A continuación se presenta el link de descarga.

- ✓ http://code.google.com/p/ogreappwizards/

Escoger el archivo Ogre_VC8_AppWizard_1.7.2_2.exe.

- **Ejecutar el asistente de instalación:**

Figura C.C.2. Ventana de la instalación del AppWizard.

El wizard configura las propiedades para que se adicionen los archivos de inclusión y de biblioteca alrededor de una variable de entorno, además de que crea una plantilla para iniciar de forma fácil con Ogre.

- **Variable de entorno:** para agregar la variable, se abre una ventana de comandos y se escribe el comando *setx* y luego la dirección de la carpeta donde se extrajo el paquete de Ogre, como lo vemos a continuación.

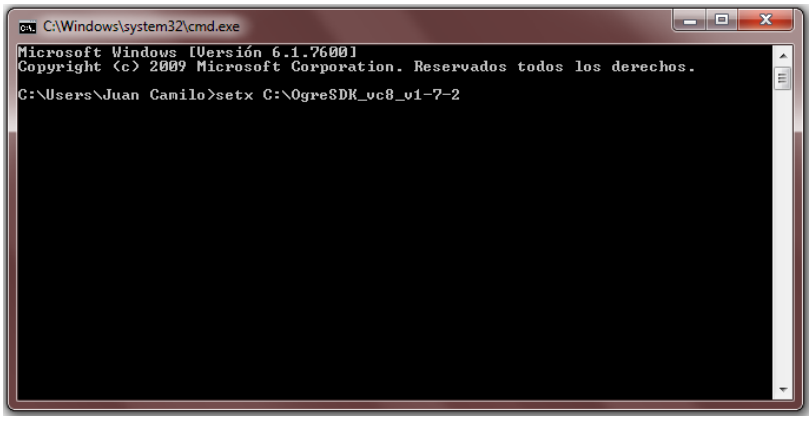


Figura C.3. Comandos para agregar la variable de entorno.

- Instalar las librerías DirectX agosto 2009 del siguiente enlace e instalar:

  ✓ www.microsoft.com/download/en/details.aspx?id=6883

- **Crear un nuevo proyecto en Visual Studio:** Nos dirigimos a archivo – Nuevo – Proyecto.

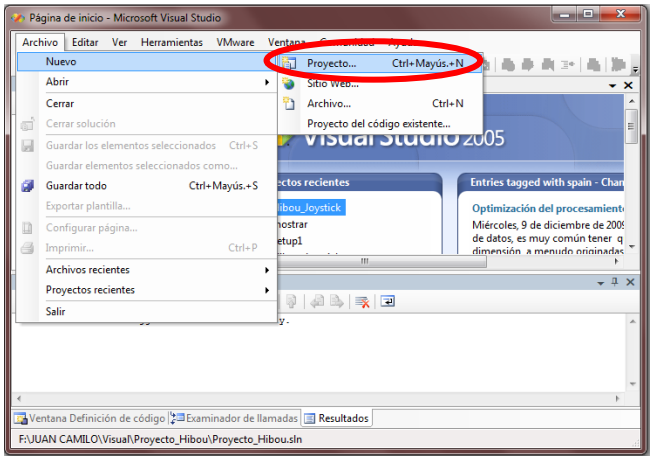
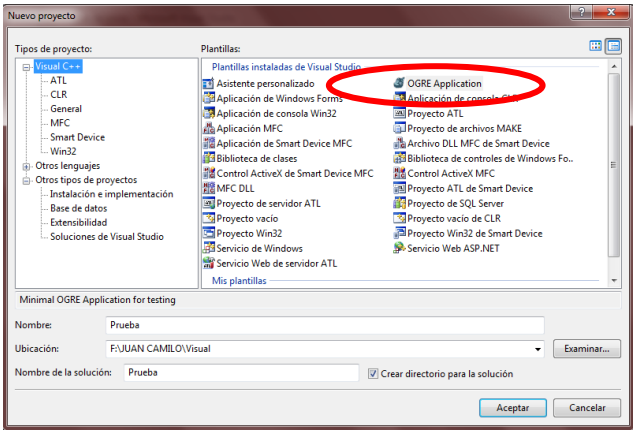


Figura C.4. Abrir nuevo proyecto.

Seleccionamos el tipo de proyecto *Visual C++,* la plantilla *OGRE Application* y luego le damos un nombre a nuestro proyecto. Aceptamos y finalizamos el asistente de aplicaciones.



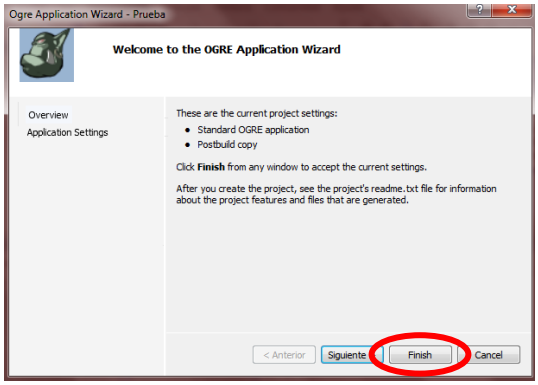


Figura C.5. Seleccionar tipo de proyecto (Izq.), Asistente de aplicaciones Ogre (der.)

Visual Studio automáticamente abre una plantilla sobre la cual se puede probar si en verdad se configuro de la forma correcta el motor gráfico, además de seguir trabajando cobre la misma plantilla.

Esta configuración tiene un pequeño problema que encontramos al instalar esta versión de Ogre3D y que se explicara y corregirá a continuación.

En la carpeta *OgreSDK_vc8_v1-7-2* se encuentra un carpeta con el nombre *boost_1_42*. El *AppWizzard* se encuentra configurado para re direccionarse a una carpeta con una versión de *boost* más reciente, haciendo referencia a la *boost_1_44.* Para solucionar este inconveniente hacemos lo siguiente.

Vamos al explorador de soluciones, damos click derecho sobre el nombre del proyecto y seleccionamos *propiedades*.

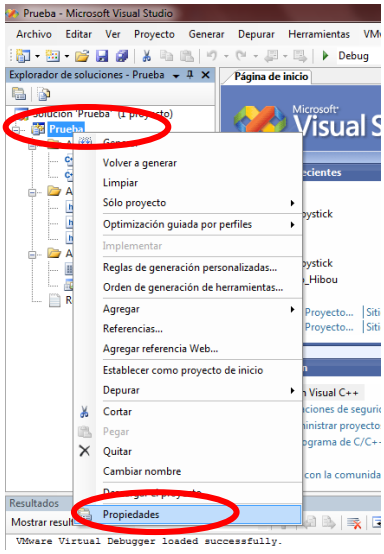


Figura C.6. Explorador de soluciones

Se abre la ventana de *páginas de propiedades.* Es importante tener en cuenta cómo vamos a correr el proyecto si en *Debug* o en *Release,* para el proyecto se compilara en *Release,* así que vamos a Administrador de Configuración y cambiamos a *Release.* Luego vamos al árbol de propiedades de configuración – C/C++ - General – Directorios de inclusión adicionales y cambiamos la siguiente línea:

$(OGRE_HOME)\boost_1_44

Por:

$(OGRE_HOME)\boost_1_42

Hacemos lo mismo en Vinculador – General – Directorios de bibliotecas adicionales

$(OGRE_HOME)\boost_1_44\lib

Por:

$(OGRE_HOME)\boost_1_42\lib

Ya hecho esto generamos la solucion y buscamos en la carpeta de ogre la carpeta bin y luego release, donde se encuentra el ejecutable de este archivo. Ejecutamos la aplicación que inicia con la siguiente ventana.

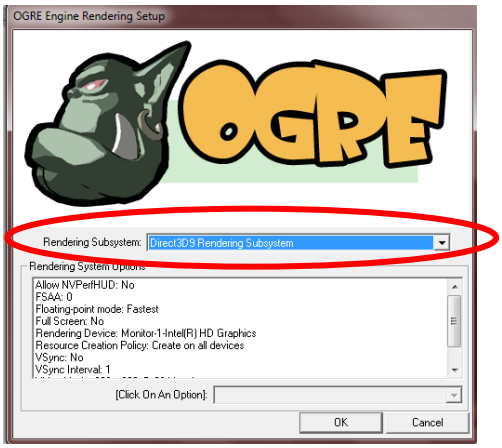


Figura C.7. Ventana de inicio de aplicación de Ogre

Seleccionamos el rendering subsystem y luego OK.

Figura C.8. Aplicación prueba de Ogre3D.

C.2. INSTALACION DE BLENDER Y SU UTILIZACION.

- **DESCARGA DE ARCHIVOS E INSTALACION.**

Blender es software libre y se puede obtener del siguiente enlace de descarga:

- ✓ *www.blender.org/download/get-blender/*

Además de este se necesita la versión apropiada de Python, que es el lenguaje sobre el cual Blender es ejecutado. En ese mismo link de descarga se puede verificar la versión compatible con el Blender y luego dirigirse al siguiente enlace para realizar su descarga.

- ✓ *python.org/download/*

Ogre3D tiene un script que trabaja junto a Blender que se puede descargar de la siguiente dirección.

- ✓ *www.xullum.net/lefthand/downloads/temp/BlenderExport.zip*

Una vez que ya se tengan los archivos de Blender, Python, realizamos su instalación. Ya hecho esto descomprimimos el archivo *.zip* llamado BlenderExport y copiamos esos archivos en la siguiente dirección.

- ✓ *C:\Users\<name>\AppData\Roaming\BlenderFoundation\Blender\.blender\scripts*

- **TRABAJANDO CON EL SCRIPT OGREMESHES.**

Antes de exportar una malla se necesita importarla, es decir traerla a Blender. Para ello nos dirigimos a File – Import, y buscamos la extensión de nuestro interés.

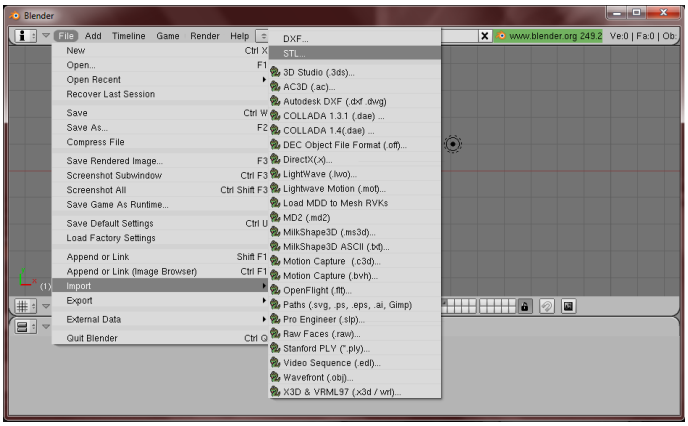


Figura 1.9. Interfaz de Blender para importar un archivo.

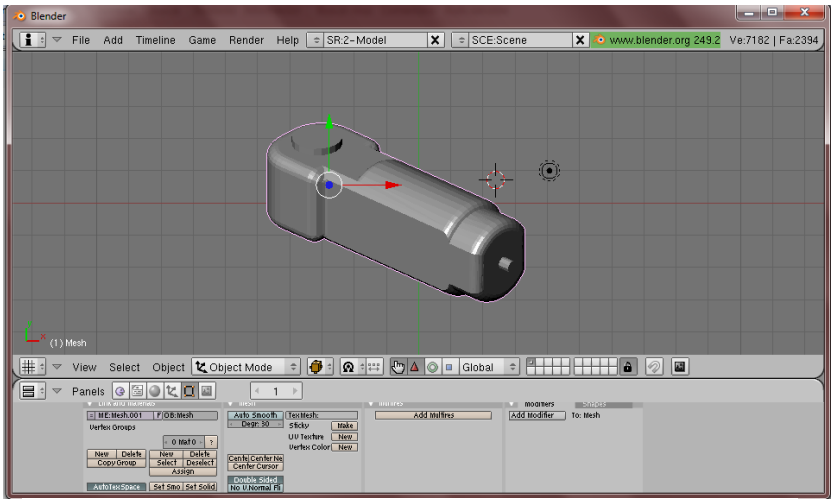


Figura 1.10. Pieza importada a Blender.

Ahora podemos agregarle un material. Presionamos F5 y se activan las propiedades de materiales para la malla.

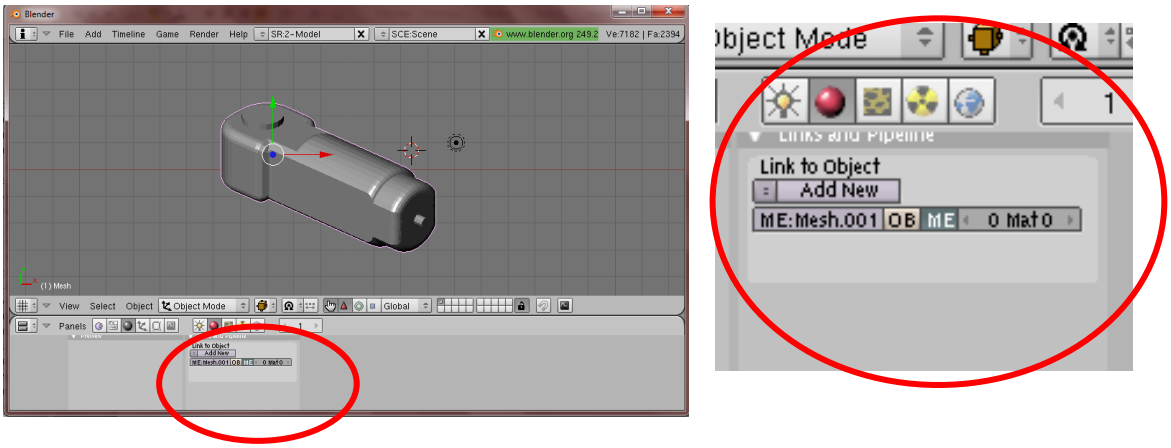


Figura C.11. Propiedades de los materiales.

Cambiamos el nombre *Mesh.001* por el nombre (en este caso “3”) de la malla y damos click en *Add New.*

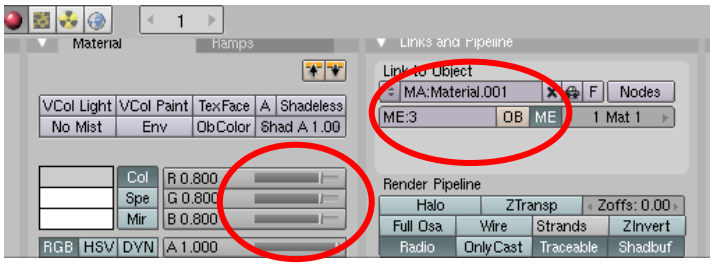


Figura C.12. Panel de colores y efectos de malla.

Cambiamos el nombre del material por el mismo nombre de la malla y modificamos las barras de color RGB para dar una tonalidad a la malla. Luego de esto nos preparamos para exportar la malla, asi que vamos a File – Export – OGRE Meshes.

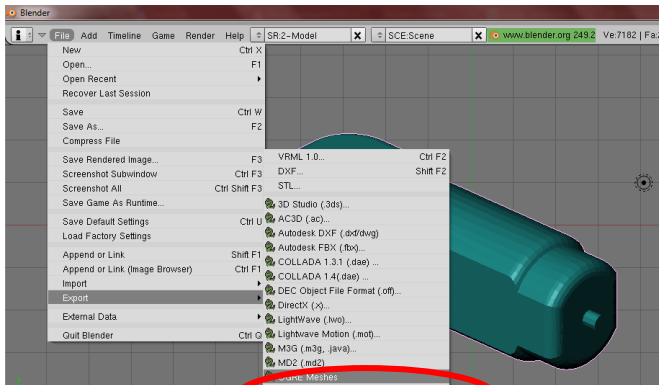


Figura C.13. Ingresando de script de Ogre3D.

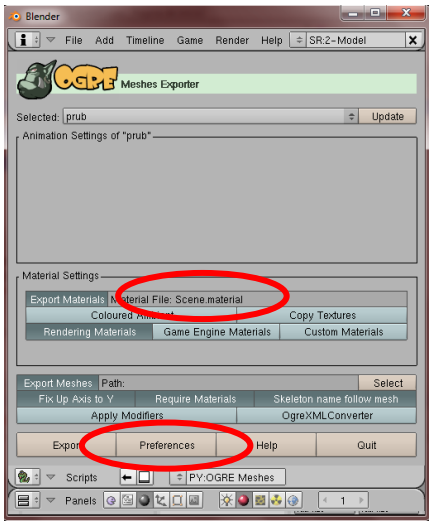


Figura C.14. Interfaz de OGRE Meshes

El círculo rojo muestra el nombre del material de salida *Scene.material* lo cambiamos al nombre de nuestro archivo (*3.material)*. Como último paso de configuración seleccionamos la casilla *Preferences.*

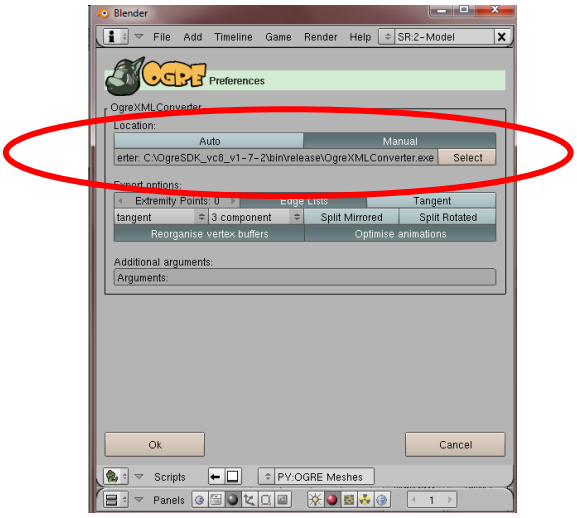


Figura C.15. Preferencias de OGRE Meshes.

Activamos el botón manual y seleccionamos el ejecutable *OgreXMLConverter.exe* que se encuentra en la carpeta bin\release de Ogre y le damos en OK.

Buscamos una carpeta para exportar los archivos y presionamos el botón Export. De esta forma ya tenemos el *.mesh* y el *.material* que son guardados en la carpeta de Ogre de la siguiente forma:

- ✓ El *.mesh* en la carpeta: …\media\models de Ogre.
- ✓ El *.material* en la carpeta: …\media\materials\scripts de Ogre.

Y podemos utilizar la pieza para ser renderizada.

Figura C.16. Pieza renderizada en Ogre3D.

C.3. COMANDOS UTILES EN EL SOFTWARE.

Para ofrecer una mayor comodidad con la interacción con el software se han agregado diferentes teclas funcionales que están con la visibilidad del entorno, la ubicación de la cámara con se observa el robot.

TECLAS	FUNCION
Q	Subir el robot
W	Bajar el robot
A	Habilitar dispositivo auxiliar
S	Deshabilitar dispositivo auxiliar
D	Habilita el movimiento de la cámara del robot
F	Deshabilita el movimiento de la cámara del robot
Z	Calcula el MGD en MGD_posActual_JOYST.txt
U	Subir cámara del ambiente
J	Bajar cámara del ambiente
K	Ver los polígonos del ambiente
L	Ver los polígonos de la mini pantalla

UP	Cámara del ambiente en -Z
DOWN	Cámara del ambiente en +Z
RIGHT	Cámara del ambiente en +X
LEFT	Cámara del ambiente en -X

Tabla C.1. Comandos con el teclado.

Para girar la cámara del ambiente se hace uso del ratón presionando los dos botones, izquierdo y derecho del dispositivo y moviéndolo según sea necesario. Un movimiento en X del ratón genera un movimiento en Y de la cámara ante el ambiente, y un movimiento del ratón en Y produce un movimiento en X de la cámara.

C.4. Habilitando el Joystick en Ogre3D

Para poder implementar el Joystick en Ogre3D se tiene que seguir unos cuantos pasos, primero que todo se debe llamar a la librería de OISJoyStck.h la cual contiene todas las funciones y la estructura para su implementación, si se está programando orientado a objetos, como se realizó en este proyecto, se debe declrar como publica el listener del JoyStick que se encargara de leer todos las variables provenientes de este.

```
public OIS::JoyStickListener
```

Una vez hecho esto se deben declarar las funciones que se van a utilizar, hay que tener cuidado en este punto ya que solo se podrán definir en el archivo de cabecera las funciones que estén en el OISJoyStick.h declaradas como funciones virtuales puras, estas funciones se pueden reconocer ya que tienen un = 0; al final de esta.

```
virtual bool buttonPressed( const JoyStickEvent &arg, int button ) = 0;

virtual bool buttonReleased( const JoyStickEvent &arg, int button ) = 0;

virtual bool axisMoved( const JoyStickEvent &arg, int axis ) = 0;
```

De esta manera estas tres funciones ya podrán ser heredadas al archivo principal y ser utilizadas de manera normal, la primera función se llamara cada vez que un botón especificado dentro de la función se presionado, la segunda función se llamara en caso contrario cada vez que el mismo botón sea liberado y por último la

última función será llamada en el momento en que la palanca del joystick sea movida en cualquier dirección.

Para lograr que lo anterior funcione se debe definir una variable de tipo OISJoyStick de esta manera.

- **`OIS::JoyStick* mJoystick;`**

Una vez declarado lo anterior en el archivo de cabecera se debe configurar sus parámetros en el código fuente, en el create frame listener se debe incializar el OIS (Outout Imput Systems) este punto sirve para incializar los distintos tipos de mandos que se le puedan conectar al programa como los son el teclado, el mouse y el JoyStick.

```
Ogre::LogManager::getSingletonPtr()->logMessage("* Initializing OIS *");
OIS::ParamList pl;
size_t windowHnd = 0;
std::ostringstream windowHndStr;
Window->getCustomAttribute("WINDOW", &windowHnd);
windowHndStr << windowHnd;
pl.insert(std::make_pair(std::string("WINDOW"), windowHndStr.str()));
mInputManager = OIS::InputManager::createInputSystem( pl );


//inicializando las variables  con buffer
mJoystick = static_cast<OIS::JoyStick*>(mInputManager->createInputObject(
OIS::OISJoyStick, true ));
```

El create frame listener es una funcion la cual solo se llama una vez por lo que debe incializarse la variable global declarada en el archivo de cabecera.

- **`mJoystick->setEventCallback(this);`**

Ahora cualquier cambio de estado de tipo *JoyStick* se le asignara a *mJoyStick*. Para terminar de configurar el *JoyStick* se tienen que capturar cada cambio en el mismo, los cuales deben ser leídos cada Frame, por lo que dentro del *frame rendering queued se* le ordena a mJoyStick que lea los cambios de estado durante el último *Frame*.

- **`mJoystick->capture();`**

Ya terminado la configuración del JoyStick, se puede empezar a realizar las acciones que desee dependiendo de las entradas del mismo, por ejemplo como se mostró en la declaración de las funciones en la página anterior, tenemos dos variables “axis” y “button”, axis puede tomar uno de 2 valores 0 o 1 indicando si está realizando movimientos en X o Y , por lo tanto podemos usar algo como esto.

- **`Ogre::int32 x = evt.state.mAxes[axis].abs;`**

Hemos definido un entero X de 32 bits de tipo Ogre al cual se le asignan los valores del cambio en el movimiento de la palanca que pueden ir de -32767 hasta 32767 dependiendo del valor de axis, así podríamos generar un switch como este.

```
switch(axis)
{
	case 0:
		V1yJoys = x;
		break;
	case 1:
		V1xJoys = x;
		break;
}
```

Con el cual le asignó cambios al movimiento del Hibou dependiendo del movimiento de la palanca de mando y este comando funciona muy bien aun cuando la palanca apunta a valores distintos de cero en X y Y simultáneamente.

Para el caso de la variable button, ésta devuelve un valor entero dependiendo de la cantidad de botones que posea el Joystick. Así dependiendo de que botón se esté presionando se puede realizar una acción distinta, igualmente para cada vez que se deje de presionar el botón.