UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

Centro de Investigación en Tecnologías de Información y Sistemas

T E S I S

ANÁLISIS, DISEÑO Y SIMULACIÓN DE UN SEGUIDOR PARA UN AMBIENTE VIRTUAL .

PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS COMPUTACIONALES

P R E S E N T A

JOSÉ JUAN ZÁRATE CORONA

DIRECTOR DE TESIS: DR. VIRGILIO LÓPEZ MORALES

PACHUCA DE SOTO, HIDALGO. SEPTIEMBRE DE 2008

Agradecimientos

A DIOS.

A mi esposa Alejandra que es un amor.

A mi hija Alexia que es mi alegría.

A mi hijo Juan Pablo que es la esperanza.

A mis Padres Ale y Manuel.

A mi Familia de Pachuca, Xalapa y USA.

A mi primo Jonathan.

A mi asesor de tesis, Dr. Virgilio López M.

A mis maestros.

A mis compañeros y amigos.

Índice general

Índice de Tablas

Índice de figuras

RESUMEN

El propósito de este trabajo es el desarrollo de un Seguidor de Posición y Orientación Electromecánico; el dispositivo electromecánico tiene dos elementos, el mecánico y el eléctrico; el elemento mecánico es una estructura que se conforma de un conjunto de eslabones, articulaciones y un casco colocado en la cabeza del usuario, dispuestos de tal manera, que se permite el libre movimiento de la cabeza, ya sean movimientos giratorios o de traslación, estos movimientos son detectados por medio de las articulaciones, que en realidad, es también, el elemento eléctrico.

La interfaz recibe las señales de los sensores (elemento eléctrico) y los envía a la computadora a través del puerto serie. Un programa desarrollado en Visual C++, realiza los cálculos de posición y orientación por medio del modelo cinemático basado en el algoritmo de Denavit-Hartenberg

Por último estas coordenadas operacionales ajustan la visión de un ambiente virtual de manera que corresponda al movimiento de la cabeza; la creación del mundo virtual esta realizado con librerías en OpenGL para Visual C++.

Capítulo 1

Introducción

Resumen

Este capítulo inicia con la definición de realidad virtual, para posteriormente dar un poco de historia citando desde los inicios, los precursores, y las nuevas tecnologías, que dieron creación a lo que hoy se conoce como realidad virtual. Por último se dan los objetivos, justificación y otros puntos que sustentan este trabajo de tesis.

1.1. Introducción.

Una herramienta revolucionaria, que ha impactado en todos los sectores sociales, culturales y económicos es la realidad virtual. La realidad virtual es la manipulación de los sentidos humanos (siendo actualmente el tacto, la visión, el olfato, el gusto y oido) por medio de entornos tridimensionales sintetizados por computadora, en el que uno o varios participantes acoplados de manera adecuada al sistema de computación, interactúan rápida e intuitiva de tal forma que la computadora desaparece de la mente del usuario, dejando como real el entorno generado por la computadora.

Cuando se habla de realidad virtual la gente lo asocia principalmente con dos dispositivos que son: el guante (Data Glove) y el casco visor (HMD). Sin embargo cuando se habla de la inmersión de la realidad virtual, se debe mencionar un dispositivo HMD (ver la Figura 1.1). El propósito básico de este dispositivo es visualizar una imagen delante de lo ojos, no importando dónde se pueda mirar. Sobre si, un HMD es bastante inútil, pero combinado con un sistema de seguimiento (tracking), y un software capaz de proyectar una imagen 3D en tiempo real; el HMD es una parte esencial para crear la ilusión de inmersión a un ambiente generado por la computadora.

Figura 1.1: Casco de Realidad Virtual.

1.2. Antecedentes.

El origen de la realidad virtual se sitúa, en el campo de la investigación militar, hace aproximadamente unos 36 años. En aquella época (finales de los sesenta) el Departamento de Defensa americano estaba muy interesado en dos tipos de tecnología. La primera era la tecnología de simulación de vuelo, porque los simuladores permitían entrenar a los pilotos sin riesgo para ellos, ni para los aviones; además, la simulación permitía controlar el entrenamiento, fijando las condiciones ambientales y los escenarios específicos a voluntad. Y la segunda, estaba interesado en la investigación sobre dispositivos de control especializados para la conducción de aeronaves, especialmente cascos de visualización que permitieran combinar información gráfica e imágenes reales.

El primer prototipo de casco visor (la espada de Damocles) fue construido en 1968 en Harvard por Ivan Sutherland (fundador más tarde de la compañia Evans y Sutherland) y David Cohen. Ivan Sutherland había desarrollado como tesis doctoral en el MIT a principios de los sesentas, una serie de programas bidimensionales interactivos utilizando la primera PC transistorizada, el TX-2. David Cohen, por su parte, había desarrollado el primer simulador de vuelo tridimensional, que utilizaba gráficos vectoriales.

La espada de Damocles (ver la Figura 1.2), desarrollada según una idea que el propio Sutherland había propuesto en su tesis doctoral tres años atrás, consistía en dos diminutos tubos de rayos catódicos (de media pulgada de diámetro) mediante los cuales podían contemplarse imágenes gráficas sobreimpresionadas en la escena real, gracias a un sistema de espejos. El dispositivo estaba suspendido del techo mediante un brazo mecánico (de ahí su nombre), lo que permitía conocer la posición y orientación de la cabeza del usuario.

El primer prototipo era un monoscopio, pero luego se añadió la capacidad de estereoscopía, ver la Figura 1.3. Con aquel prototipo, el usuario podía ver la estructura de

Figura 1.2: Espada de Damocles.

un objeto flotando en mitad de la habitación, pudiendo contemplar las distintas caras a medida que se desplazaba por la misma.

Figura 1.3: Visión estereoscópica.

Con los criterios de hoy en día, la espada de Damocles no era muy avanzada desde el punto de vista gráfico, pero para aquélla época se trataba de una auténtica revolución. De hecho, con aquel prototipo acababan de nacer los sistemas de realidad virtual. El sistema incluía ya prácticamente todos los conceptos en los que la tecnología de realidad virtual iba a basarse: gráficos tridimensionales, estereoscopía, localización de la posición del usuario, posibilidad de navegar alrededor de un objeto y contemplarlo desde todas las posiciones, técnicas de inmersión, etc. (Cf. [González, 1995]).

1.3. Planteamiento del problema.

Uno de los dispositivos que son muy útiles en la Realidad Virtual es el casco seguidor. Los diferentes tipos de seguidores realizan mediciones de ciertas variables para indicar la posición y orientación de un cuerpo en el espacio, pero la gran mayoría sólo puede detectar la orientación, esto limitado por las técnicas que utilizan de medición; sin embargo los pocos seguidores que tienen la capacidad de indicar la posición y orientación tienen un alto costo y poca exactitud; por lo tanto es necesario un dispositivo que mida la posición y orientación con mayor exactitud a un menor costo.

1.4. Objetivo principal.

Análisis y diseño de un seguidor de posición y orientación electromecánico, que permita la navegación en un mundo virtual, a través de los movimientos realizados por un casco montado en la cabeza de un usuario.

1.5. Objetivos particulares.

- Diseño y elaboración de la estructura del seguidor electromecánico.
- Modelado cinemático del mecanismo de eslabones articulados, así como validación de algunas propiedades mediante estudios en simulación.
- Diseño y simulación del la interfaz de potencia, utilizando un microcontrolador (el PIC16F877).
- Elaboración del ambiente virtual, desarrollado en OpenGL.

1.6. Justificación.

Si el usuario no tuviera la posibilidad de influir sobre el estado de un mundo virtual de ninguna forma, o cambiar su punto de vista de la escena, entonces no sería necesario sintetizar las imágenes, porque bastaría con grabar de antemano la película de lo que el usuario va a ver. En un sistema de realidad virtual, por el contrario, el usuario dispone de un mayor o menor control, pudiendo influir sobre lo que el sistema va a mostrarle.

Un dispositivo típico de control se puede realizar por medio de un joystick o un Mouse; la interacción es muy simple, se realiza a través de los movimientos de la mano y estos se ven reflejados en un monitor; pero ninguno de los ellos proporciona la sensación adecuada de navegación, ya que sus movimientos son limitados: adelante, atrás, arriba, girar, caminar, brincar, disparar, volar y a su vez todas estas acciones son a través de la mano.

Dentro de las tecnologías de hardware que permiten al usuario moverse independientemente a través del mundo virtual, es por medio de un dispositivo seguidor de posición y orientación que va colocado en la cabeza del usuario. Por lo cual esta tecnología nos proporciona una de las características más básicas de la realidad virtual, que es la navegación.

1.7. Tareas desarrolladas.

Para desarrollar este trabajo fue necesario realizar las siguientes actividades:

- Diseño y elaboración de un robot antropomórfico con seis grados de libertad.
- Obtención del modelo cinemático directo de posición (MCDP) por el método de matrices homogéneas.
- Simulación del MCDP en Matlab
- Construcción del ambiente virtual, empleando librerías de OpenGL.
- Asignación de movimientos de posición y orientación, para poder navegar a través del ambiente virtual.
- Diseño y simulación de un sistema de adquisición de datos por medio del PIC16F877.
- Incorporación del MCDP al ambiente virtual.

1.8. Hipótesis.

Utilizando el algoritmo de Denavit-Hartenberg se obtendra el modelo cinemático directo de posición, con el cual se diseñara y simulara un robot antropomórfico y un sistema de adquisición por medio de un microcontrolador y se desarrollara un programa en OpenGL para el ambiente virtual. Esto con la finalidad de obtener los módulos que se requieren la elaboración de un seguidor electromecánico.

1.9. Aporte tecnológico.

Se proporciona un dispositivo que permite conocer la posición y la orientación real del usuario, reflejándose los movimientos de la cabeza en un escenario virtual, permitiendo un control mucho más natural en la navegación, a través de un mundo virtual.

1.10. Herramientas empleadas.

Las herramientas utilizadas para la realización de este proyecto son diversas, en primer lugar se emplea el método de las matrices homogéneas y ángulos de Euler, para obtener el MCDP, así como el software de Matlab 7 para la procesamiento algebraico y simulación del modelo cinemático del robot antropomórfico. Posteriormente para el diseño y simulación de la interfaz, se utiliza el software de Proteus, ya que éste permite

conocer el comportamiento físico y lógico del PIC16F877. Por último en el desarrollo del ambiente virtual, se emplean librerías de OpenGL para Visual C++.

Capítulo 2

Estado del arte

Resumen

El presente Capítulo, trata principalmente de los principios y de los diferentes tipos de sistemas seguidores, se mencionan sus características principales y se explica su funcionamiento básico; por último se presentan algunos dispositivos seguidores que se encuentran en el mercado de la realidad virtual.

2.1. Introducción.

Un buen casco seguidor HMD es indispensable para generar una visualización inmersiva. Los sistemas seguidores de la cabeza (trackers head) miden la orientación y la posición de la cabeza, o solo la orientación en sistemas más baratos, y pasan esta información a un software, de tal manera que se pueda generar la vista apropiada en un mundo virtual con respecto a la posición y orientación de la cabeza. El seguidor de la cabeza su principio de funcionamiento es el mismo al seguidor de las manos, y así con los distintos seguidores de otras partes de cuerpo.(Cf. [Robin Hollands, 1996]).

2.2. Especificaciones.

Las especificaciones más importantes de un HMD son las siguientes:

- Grados de libertad.

  Esto es tomado generalmente para indicar cuántas diversas variables espaciales puede medir el seguidor. Por ejemplo los trackers de orientación de tres grados de libertad (3DOF) miden elevación (pitch), desviación (yaw) y giro (roll) (Figura 2.1).

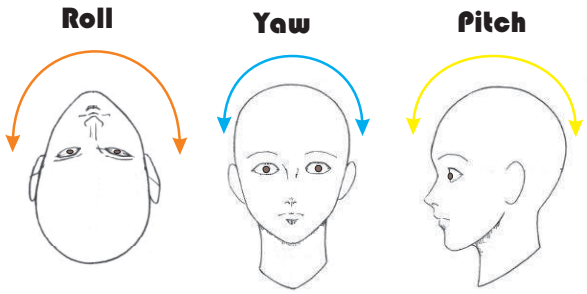


Figura 2.1: Tres grados de libertad.

Un sistema de seis grados de libertad (6DOF) proporciona la orientación y posición que es toda la información que se necesita para ser un seguidor completo, (Figura 2.2). En los seguidores mecánicos se da el caso de que el número de articulaciones nos indica el número de grados de libertad.

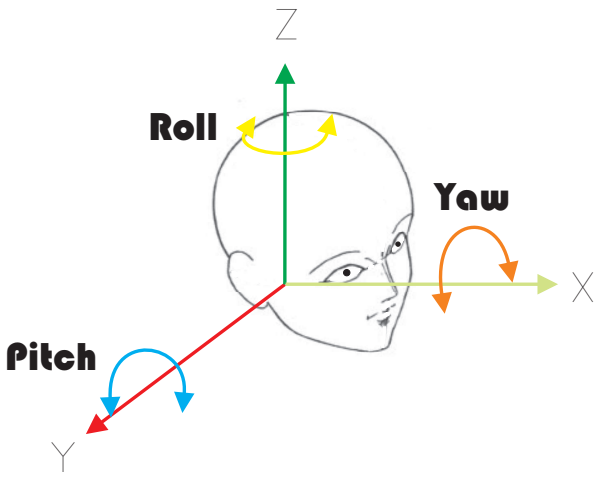


Figura 2.2: Seis grados de libertad.

- Con fuente.

  Los seguidores con fuente requieren una base asegurada en alguna parte en el cuarto además del sensor real en la cabeza. Existe una cierta clase de conexión entre la base y el sensor y todas las lecturas producidas están relacionadas con la base. Los seguidores con fuente tienen siempre un rango de operación limitado esto se debe a la base.(Cf.[Robin Hollands, 1996]).

- Sin fuente.

  Los seguidores sin fuente no requieren una base y por lo tanto teóricamente tienen un rango de operación ilimitado, aunque los cables que van a la PC limitan generalmente esto. Aunque los seguidores sin fuente de orientación son comunes en el usuario de HMD, el seguidor sin fuente de posición es considerado más complejo, e incluso ha sido considerado un funcionamiento no confiable en los laboratorios de investigación todavía.

- Rango.

  Para los seguidores con fuente, el rango describe la distancia máxima de la base en el que el sensor puede funcionar. Esto no es siempre un límite, puesto que la exactitud puede descender significativamente con el rango, y una operación fuera del rango indicado, puede ser posible, pero con exactitud dudosa.

- Exactitud.

  Es la diferencia que existe entre el valor de la posición verdadera y el resultado obtenido de la posición del seguidor. La exactitud tiende a ser alcanzada lo mejor posible bajo condiciones óptimas; a menudo su unidad se expresa en grados rms (eficaces) para la orientación y pulgadas rms (eficaces) para la posición. En condiciones de funcionamiento desfavorable se puede afectar la exactitud, ésta tiende a empeorar mientras se alcanza los límites del rango de operación del seguidor; esto es en el caso de los seguidores con fuente.

- Resolución.

  Contrario a la exactitud, la resolución es una medida de la diferencia más pequeña de la distancia o el ángulo que causan un cambio en las lecturas del seguidor. Esto es a menudo más pequeño que la exactitud. Por ejemplo, un seguidor con una exactitud de 0.03in y una resolución de 0.0002in puede medir un cambio en la posición de 0.0002in, pero la lectura real de la posición podría ser todavía de 0.03in.

- Latencia.

  En sistemas de realidad virtual, el retraso es de importancia vital ya que esto mide el tiempo tomado entre un movimiento que es iniciado en el sensor, y el cambio en la lectura de la PC principal. El retraso del sensor cuando está combinado con el retraso de la representación, conduce a un retraso del sistema; esto se representa

como un solo retardo entre el usuario cuando mueve su cabeza y los cambios de la visión. Si este retraso es demasiado alto puede inducir a mareos, por lo cual la sensación de inmersión es destruida.

- Velocidad de actualización.

  Es el número de muestras que se pueden obtener por segundo. Si el retraso es corto comparado con la velocidad de actualización, entonces la velocidad de actualización puede ser el factor de contribución al retraso del sistema. Por ejemplo, si el sensor solamente actualiza a 15 Hz, entonces 67ms podrá hacerse un movimiento de sensor antes de que aquel movimiento pudiera ser leído, no importado que pequeño es el retraso del sensor (Cf. [Robin Hollands, 1996]).

2.3. Tecnologías del seguidor.

2.3.1. Seguidor Electromecánico

El seguidor electromecánico es uno de los métodos más simples a utilizar, también es uno de los más barato, más rápidos y más exactos. Consiste en una serie de eslabones mecánicos conectados al casco y una base; con suficientes articulaciones para permitir que la cabeza se mueva alrededor libremente, (ver Figura 2.3).

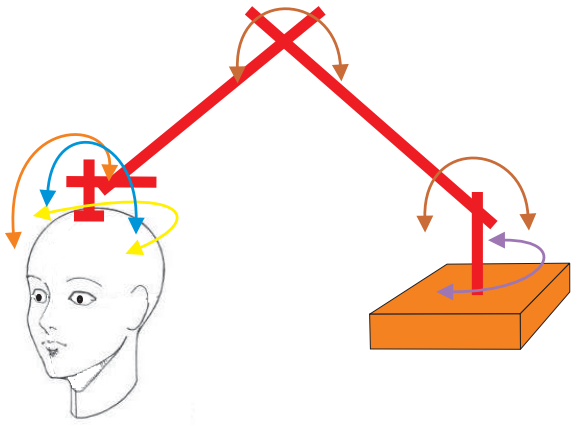


Figura 2.3: Seguidor Electromecánico.

Cada una de las articulaciones del eslabón tiene un sensor midiendo su ángulo, después de tomar las lecturas de los ángulos, se aplican cierto tratamiento matemático (se echa lápiz) para determinar la posición y la orientación del extremo del eslabón. La tecnología detrás de estos seguidores es simple, están libres de interferencia del ambiente que lo rodea. El inconveniente es el rango del seguidor que está limitado a lo

que eslabones alcancen, y no deben haber objetos cercanos que puedan obstruir a los eslabones mientras el seguidor se mueve.

2.3.2. Dispositivos electromecánicos.

Este tipo de dispositivos no han tenido una excesiva aceptación comercial hasta el momento. El BOOM de los laboratorios Fake Space es el más conocido de los sistemas que emplean dispositivos electromecánicos de localización.

Figura 2.4: BOOM.

El dispositivo BOOM (ver la Figura 2.4), es un brazo articulado formando un todo con el dispositivo de visualización, que el usuario maneja como si fuera un periscopio. A través del contrapeso, los problemas del peso en el dispositivo de visualización son eliminados. El BOOM permite que sean usados dispositivos de mayor resolución en visualización CRT (Tubo de rayos catódicos) en lugar de las bajas resoluciones de los dispositivos LCD. Los detalles de las especificaciones del BOOM3C se proporcionan en la Tabla 2.1

La empresa canadiense Shooting Star Technology, por su parte, vende el sistema ADL-1 de bajo costo, que un es brazo articulado acabado en una especie de banda que el usuario se coloca alrededor de la frente (ver la Figura 2.5). Su aplicación principal, en consecuencia, esta en el campo de los sistemas de sobremesa. Los detalles de las especificaciones del ADL-1 se proporcionan en la Tabla 2.2

Tabla 2.1: BOOMC3.

Especificación	**Valor**
Velocidad de muestreo	70 Hz
Latencia	200ms
Exactitud	0.16in
Resolución	0.1°
Volumen de Trabajo	6 ft circular y 2.5 ft vertical
Campo de vision	140°
Precio	$95,000 USD

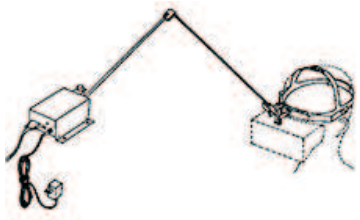


Figura 2.5: ADL-1.

Tabla 2.2: ADL-1.

Especificación	**Valor**
Velocidad de actualización	240Hz
Velocidad de Muestreo	240Hz
Latencia	0.35-1.8ms
Exactitud	0.2in
Resolución Lineal	0.2in
Resolución Angular	0.3°
Volumen de Trabajo	36ft Medio circular y 18ft Altura
Precio	$1,300 USD

2.3.3. Seguidor Electromagnético

Una de las tecnologías comerciales más populares es la del seguidor electromagnético (Figura 2.6). Para este dispositivo la base y el sensor no son conectados por una conexión física, sino por medio de un campo magnético generado por la base. La base contiene tres bobinas ortogonales, que se excitan secuencialmente para producir un campo magnético. El sensor también contiene tres bobinas ortogonales, que generan una corriente en respuesta al campo magnético generado por la estación de base. Mientras que se da vuelta el sensor, toma diversas proporciones del campo transmitido de cada bobina del transmisor y, como el sensor se pone más lejos de la base, la potencia de la señal recibida se reduce. Esta información se puede entonces procesar para determinar la posición y la orientación del sensor. Dos tipos de seguidor electromagnético están disponibles, uno utiliza la corriente alterna (CA) para generar el campo magnético y, la otra aplicación un campo magnético pulsado por corriente directa (CD), que es menos propenso a interferencia de objetos metálicos.

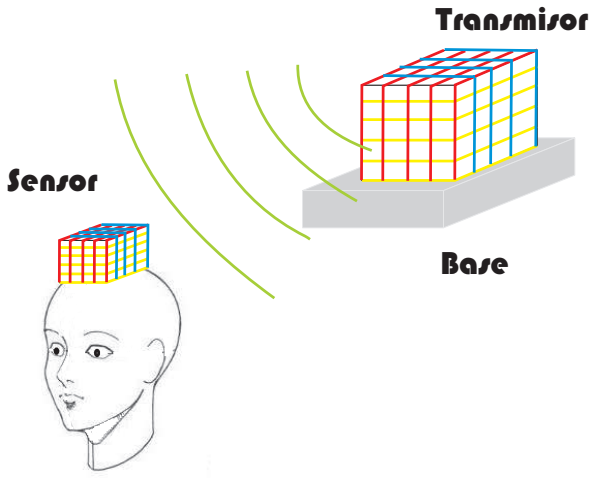


Figura 2.6: Seguidor Electromagnético.

Debido a que en la tecnología del seguidor, utiliza la generación de un campo magnético uniforme, sus lecturas pueden ser alteradas seriamente por los objetos próximos del metal, los cuales distorsionan el campo magnético. La presencia de metales ferromagnéticos (acero suave y ferrita) y cobre afecta a los seguidores de CC y de CA. Sin embargo los seguidores de CC son inmunes al acero no-ferromagnético de los metales (latón, aluminio y acero inoxidable), no así, con los CA que si se ven afectados. Además de esto, las señales recibidas, todavía se procesan antes de que un resultado confiable pueda ser producido.

A pesar de los problemas de interferencia y del retraso, esta tecnología ha tenido aprobación popular debido al tamaño pequeño del sensor (aproximádamente del tamaño

de un dado), y el hecho de que el área entre el sensor y la base no necesita ser despejada de obstáculos.

2.3.4. Dispositivos electromagnéticos.

Los dispositivos electromagnéticos de localización más conocidos son la serie 3SPACE de la empresa Polhemus y los de la firma Ascension Technologies, cuyo nombre comercial es A Flock of Birds.
La serie 3SPACE del Polhemus se compone de los siguientes productos:

- Isotrak II.

  Es uno de los primeros seguidores electromagnéticos desarrollado para sistemas de realidad virtual. Consta de un emisor y uno o dos receptores (ver la Figura 2.7). Por lo tanto, su precio varia dependiendo de el número de receptores, el costo adicional por el receptor es de $800 USD. Los detalles de las especificaciones del Isotrak se proporcionan en la Tabla 2.3

Figura 2.7: Isotrak.

Tabla 2.3: Isotrak.

Especificación	**Valor**
Velocidad de actualización	60Hz entre el no. receptores
Latencia	20ms
Exactitud	0.1in lineal, 0.75° RMS orientación
Resolución Lineal	0.0015in
Resolución Angular	0.1°
Rango	15ft
Precio	$2,900 USD

- Fastrak.

  El Polhemus Fastrak fue desarrollado en base en un rediseño de seguidor de Isotrak, (ver la Figura 2.8). Consta de un emisor y hasta cuatro receptores y hasta 8 sistemas pueden ser multiplexados para permitir hasta 32 receptores. Los detalles de las especificaciones del Isotrak se proporcionan en la Tabla 2.4

Figura 2.8: Fastrak.

Tabla 2.4: Fastrak.

Especificación	**Valor**
Velocidad de actualización	120Hz entre el no. receptores
Latencia	4ms
Exactitud	0.03in RMS lineal, 0.15° RMS orientación
Resolución Lineal	0.0002in
Resolución Angular	0.025°
Rango	10ft
Precio	$6,050 USD

- Insidetrak.

  El último equipo de la gama, es una versión mas pequeña que Fastrak. Es una tarjeta que se inserta directamente en cualquier ranura ISA de una PC (ver la Figura 2.9). Con un emisor y uno o dos receptores, su frecuencia de muestreo es igual al Fastrak. Los detalles de las especificaciones del Insidetrak se proporcionan en la Tabla 2.5

La firma Ascension Technologies fabrica y comercializa A Flock of Birds (ver la Figura 2.10), un sistema de posicionamiento electromagnético que es la competencia directa de los Polhemus. A Flock of Birds presenta la desventaja de que cada unidad de control sólo puede controlar un receptor (aunque pueden fácilmente conectarse entre sí

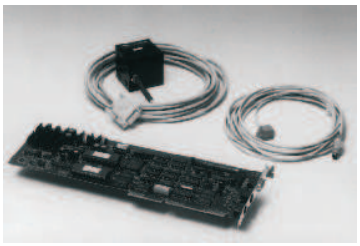


Figura 2.9: Insidetrak.

Tabla 2.5: Insidetrak.

Especificación	**Valor**
Velocidad de actualización	120Hz entre el no. receptores
Latencia	4ms
Exactitud	0.03in RMS lineal, 0.15° RMS orientación
Resolución Lineal	0.0002in
Resolución Angular	0.025°
Rango	5ft
Precio	$2,300 USD

hasta 30 unidades de control, lo que proporciona 30 receptores). Su alcance es similar al Fastrak de Polhemus, así como menor susceptibilidad a la presensia de objetos metálicos. Los detalles de las especificaciones del Flock of Birds se proporcionan en la Tabla 2.6

Figura 2.10: Flock of Birds.

Para la obtención de la orientación y la posición del casco se opta por un sistema de posicionamiento magnético, en concreto el Ascension Flock of Birds. Tras el análisis de las distintas tecnologías disponibles en la actualidad se puede concluir que este tracker es el que mejor se ajusta a las especificaciones requeridas. Han sido los de mayor éxito

Tabla 2.6: Flock of Birds.

Especificación	Valor
Velocidad de actualización	144HZ
Latencia	10ms
Exactitud	0.1in RMS lineal, 0.5° RMS orientación
Resolución Traslación	0.023n
Resolución Angular	0.1°
Rango de Traslación	3ft
Rango Rotación	180° yaw, roll y 90° pitch
Precio	$2,7000 USD

comercial debido fundamentalmente a su alta precisión y su casi nulo impacto en el peso del HMD. además pueden proporcionar el seguimiento de los 6 g.d.l. que necesita esta aplicación. Otras ventajas son su posibilidad de ser trasladados y que no necesitan mantener una línea de observación entre el emisor y el receptor, limitación que presentan muchos otros seguidores. Y aunque su campo de trabajo es limitado, se considera suficiente, puesto que las traslaciones que se van a enviar a la plataforma puesto que el espacio alcanzable por la plataforma Stewart en su movimiento se encuentra dentro de este campo.

2.3.5. Seguidor Ultrasónico

Otras aplicaciones baratas y populares del seguidor es la que utiliza tecnología ultrasónica. De nuevo, no hay conexión física entre la base y el sensor, pero esta vez la conexión es hecha por pulsos ultrasónicos (ver Figura 2.11). La base o el sensor pueden ser el transmisor ultrasónico, y con el otro se crea el receptor ultrasónico. El sistema básico para medir puramente la posición requiere un transmisor y tres receptores. Un pulso ultrasónico se envía del transmisor y tres temporizadores se inicializan. Cuando el pulso llega a cada receptor, su temporizador se para. Los tiempos representan la distancia de cada receptor al transmisor, y con tres receptores diferentes es posible resolver la posición del transmisor en el espacio. Agregando otros dos transmisores, el mismo sistema se puede utilizar para calcular la orientación así como la posición.

2.3.6. Dispositivos ultrasónicos.

Un ejemplo es el Logitech HeadTracker (ver la Figura 2.12), que emplea un conjunto de tres emisores que se activan cíclicamente, midiendo en total 9 distancias para

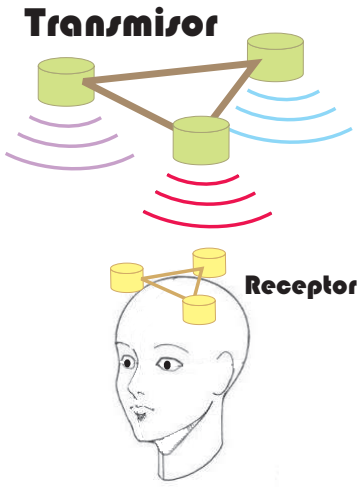


Figura 2.11: Seguidor Ultrasónico.

calcular la orientación y la posición. Los detalles de las especificaciones del Logitech se proporcionan en la Tabla 2.7

Figura 2.12: Logitech.

2.3.7. Seguidor Óptico

Hay una amplia variedad de técnicas del seguidor óptico. La técnica que más demanda tiene, es la del seguidor infrarrojo encontrado en el Atari IVR HMD y utiliza un proceso conocido como PLADAR (dirección de ángulo de pulso de luz y rango). La función del proceso PLADAR es similar al seguidor ultrasónico para la medición de posición, pero este usa LEDs infrarrojos en vez de ultrasónicos. Como la velocidad de luz es muy rápida, la medición de la distancia consiste en medir la intensidad de la señal recibida en tres receptores infrarrojos.

Tabla 2.7: Logitech.

Especificación	**Valor**
Velocidad de actualización	50Hz
Latencia	72ms
Exactitud	0.02 in RMS lineal, 0.1° RMS orientación
Resolución Traslación	0.004in
Resolución Angular	0.1°
Rango de Traslación	5ft
Rango Rotación	100° conico
Precio	$1,100 USD

La medición de la orientación utiliza un método ligeramente diferente, puesto que la intensidad de la señal recibida del LED infrarrojo es proporcional a su ángulo relativo por el receptor, dos LEDs receptores en ángulos ligeramente diferentes sobre el transmisor producirá dos conjuntos de lectura que se utilizan para calcular el ángulo en nivel de los LEDs. Otro conjunto de LEDs se puede utilizan para permitir la lectura de ambos pitch y del yaw (si el sensor se monta en la cabeza debe señalar hacia arriba), pero la medida del roll necesita un nuevo conjunto de receptores montados perpendicularmente con respecto al primer conjunto de receptores.

Otro sistema óptico es en base a una cámara fotográfica y utiliza cualquiera de los dos sistemas Dentro hacia Fuera, o de Fuera hacia Adentro.

Dentro hacia Fuera. El sistema consiste tres cámaras fotográficas montadas en el HMD, y 1000 LED infrarrojos colocados uniformemente a través del techo. La computadora pulsa los LED secuencialmente y procesa las imágenes para detectar los flashes. De acuerdo con las localizaciones de los flashes, la posición y la orientación de la cabeza se calculan. El rango del seguidor óptico del techo es limitada solamente por el área del techo cubierto por LEDs, y es así fácilmente extensible.

Fuera hacia Adentro. El dispositivo consiste en tres LED's infrarrojos, dispuestos en un patrón prescrito en el HMD, son supervisados por una cámara fotográfica montada en una posición fija en el ambiente (ver la Figura 2.13). Los LED son uno a la vez pulsado, y las posiciones de los flashes que resultan respecto a las imágenes de la cámara fotográfica junto con las relaciones sabidas entre los LED se utilizan para calcular la posición y la orientación de la cabeza.

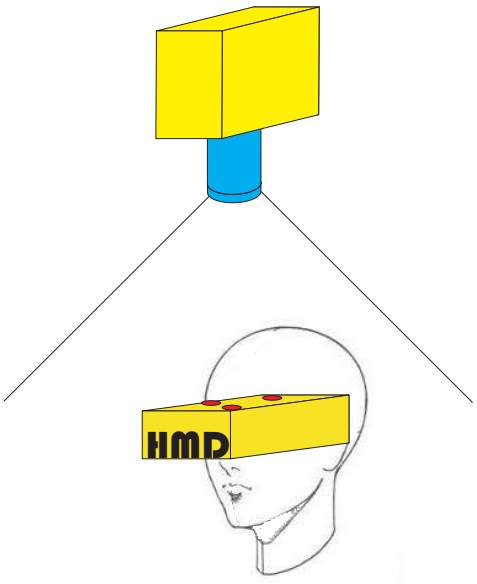


Figura 2.13: Seguidor Óptico.

2.3.8. Dispositivos ópticos.

Dentro hacia Fuera.

En Abril de 1997 un grupo de investigadores de UNC, desarrollaron el proyecto de un sistema seguidor llamado Hiball, el cual utiliza páneles con leds colocados en el techo. Los páneles en el techo fueron diseñados para substituir los plafones ordinarios del techo por una rejilla estándar (2x2 ft). El Hiball es un conjunto de 6 lentes y un arreglo de fotodiodos (ver la Figura 2.14), de modo que cada uno los fotodiodos pueden ver los leds a través de cada uno de los 6 lentes, produciendo 26 vistas. El Hiball incluye un circuito de conversión de analógico a digital.

El Hiball y el panel en techo están sicronizados por una interfaz que permite un alto rango de pulsasiones de los leds. El hardware permite llevar a cabo mas de 3000 pulsasiones por segundo, mientras que todo el sistema seguidor lleva a cabo 2000 muestras por segundo. Los detalles de las especificaciones del Hiball se proporcionan en la Tabla 2.8

Fuera hacia Adentro.

La solución del diseño es ejemplificada por el sistema DynaSight (ver la Figura 2.15), fabricado por la empresa Origin Instruments Corp. El DynaSight es un sensor autónomo que mide la posición tridimensional instantánea de un objetivo pasivo. Los

Figura 2.14: Hiball.

Tabla 2.8: Hiball.

Especificación	**Valor**
Velocidad de actualización	2000Hz
Latencia	1ms
Exactitud	0.4 in RMS lineal, 0.1° RMS orientación
Resolución Traslación	0.002in
Resolución Angular	0.01°
Volumen de trabajo	4000ft cubicos
Precio	$ 10,000 USD

detalles de las especificaciones del DynaSight se proporcionan en la Tabla 2.9

Figura 2.15: DynaSight.

El adaptador de objetivo dinámico permite seguir hasta cuatro objetivos en movimiento sujetos al adaptador (ver la Figura 2.16), mientras seis grados de libertad (6DOF) se pueden alcanzar con tres objetivos activos.

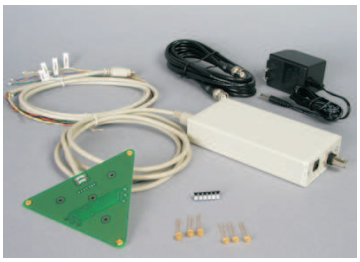


Figura 2.16: Adaptador de Objetivo Dinámico.

Un sistema MX Vicon puede manejar seis grados de libertad (6DOF), usando marcadores reflexivos y cámaras de alta velocidad y alta resolución. De esta manera el sistema puede proporcionar la más alta exactitud de posición y orientación de un objeto.

2.3.9. Seguidor No Inercial

La combinación del sensor de brújula/inclinación, también conocida como sensor gravimétrico, es la clase encontrada en la mayoría del consumidor de HMD actualmente. Esta combinación es sólamente capaz de medir la orientación y no la posición. Sin embargo, puesto que es sin fuente tiene teóricamente un rango de operación infinito y, es muy fácil instalar.

Tabla 2.9: DynaSight.

Especificación	**Valor**
Velocidad de actualización	65Hz
Latencia	10ms
Exactitud	0.2 mm RMS lineal, 0.8° RMS orientación
Resolución Traslación	0.1mm
Resolución Angular	0.4°
Volumen de trabajo	(7.3 x 5.7 x 1.5 in)
Precio	$2,300 USD

La pieza de la brújula utiliza un sensor para detectar la fuerza de un campo magnético, a menudo un sensor de efecto Hall (el efecto Hall consiste en la aparición de un campo eléctrico en un conductor cuando es atravesado por un campo magnético) en sistemas del bajo costo. Cuando el sensor este señalando directamente hacia el Polo Norte magnético, producirá un valor máximo y, cuando este señalado hacia el polo sur, producirá un valor mínimo. El valor entre el norte y el sur varía el campo magnético según la dirección del sensor, es decir cos(ángulo). Esto puede conducir al mismo valor para los ángulos simétricos alrededor del eje polar, por ejemplo noreste y noroeste, así que un segundo sensor magnético se utiliza perpendicular al primero, dando bastante información adicional para determinar el direccionamiento (heading).

Así como las brújulas electrónicas dependen del campo magnético de la tierra; de la misma manera un seguidor electromagnético depende del campo magnético del transmisor, ellos sufren problemas similares. De la inclinación del campo magnético de la tierra, y la dirección del norte magnético varía según la localización geográfica, y cualquier objeto del metal en la vecindad puede también afectar el campo. De hecho un investigador cuenta eso en un cuarto típico, el norte magnético puede variar tanto como 30 ° grados dependiendo de donde se tome la lectura.

Los sensores de inclinación son típicamente del tipo electrolítico, donde un cilindro de cristal pequeño semilleno de líquido electrolítico (ver la Figura 2.17). El cilindro contiene cinco electrodos uno en el centro y cuatro perpendiculares, mientras que el sensor de la inclinación se inclina, la cantidad de líquido en cada electrodo varía, y ésta da lugar a un cambio de la resistencia proporcional a la inclinación. Un problema con estos sensores es que pueden detectar inclinaciones solamente cuando se encuentren secos, y por lo tanto ninguna otra inclinación puede ser medida. Otro problema es que los sensores están midiendo realmente la dirección máxima aceleración. Este problema, combinado con el tiempo de asentamiento líquido, puede conducir al mundo virtual a vibrar, si se agita su cabeza usando un HMD con los sensores de inclinación.

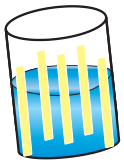


Figura 2.17: Seguidor de Inclinación.

2.3.10. Dispositivos No Inerciales.

CyberTrack 3.2 es de la compañía General Reality (ver la Figura 2.18), es un seguidor sin fuente de 3 grados de libertad, que se monta en el Cyber-Eye HMD. El seguidor mide 3 x 2.5 x 1.5 pulgadas y pesa 2 onzas. Otros detalles de las especificaciones se dan en la Tabla 2.10. Ofrece una corrección automática, compensación de temperatura, detección de distorsión automática y calibración automática.

Figura 2.18: Cybertrack 3.2

Tabla 2.10: Cybertrack 3.2.

Especificación	**Valor**
Velocidad de actualización	30Hz
Latencia	50ms
Exactitud	1.25° heading, 0.25° inclinación
Resolución Heading	0.15°
Resolución Inclinación	0.12°
Volumen de trabajo	360° horizontal, 55° inclinación
Precio	$850 USD

2.3.11. Seguidor Inercial (Giroscopios).

Otra tecnología sin fuente es la que utiliza los giroscopios para medir la orientación. Los giroscopios no son afectados por la mayoría de la cosas en el ambiente y producen lecturas exactas con un retraso muy pequeño.

Los giroscopios utilizan la inercia para permitir el monitoreo de la orientación. El giroscopio tradicional consiste en una rueca relativamente pesada montada en una estructura de hilar libre. La inercia de la rueca significa que su eje señala siempre en la misma dirección, independientemente de lo que sucede en la base sobre la que este montado. Al montar los dos ejes sobre la máquina de hilar, el giroscopio actúa como sensor de la orientación que es capaz de medir el roll y el pitch, si el eje del giroscopio está señalando verticalmente.

Además del giroscopio de rueca, hay versiones disponibles en estado sólido las cuales usan el efecto de Coriolis (ver la Figura 2.19). La fuerza de Coriolis es experimentada por cualquier cosa que se mueve dentro de un cuerpo que rota, y se ejerce en una forma perpendicular para los ejes de rotación y en la dirección de velocidad; un experimento simple es pararse en el costado de un carrusel mirando hacia el centro y tratar de patear el eje; la fuerza de Coriolis hará que su pie se vaya hacia un lado.

Figura 2.19: Giroscopio.

Un giroscopio de estado sólido consiste en un prisma triangular, con un excitador en una cara y los receptores en las otras dos. El excitador envía pulsos en la frecuencia resonancia del prisma, que son captadas igualmente por ambos detectores. Cualquier rotación alrededor del eje del prisma tiene como resultado la fuerza de Coriolis la cual causa la amplitud de la señal recibida sea mayor en un detector que en el otro. La velocidad de rotación entonces puede ser medida como una proporción de la diferencia entre las dos lecturas del detector.

Los giroscopios mecánicos de rueca continúan solamente señalando en la misma dirección en sistemas teóricamente ideales. En un sistema real la fricción en los cojinetes del giroscopio hace la rueca pierda algo de su energía y comiencen ir a la deriva. Los giroscopios de estado sólido de velocidad no tienen ninguna parte móvil, y por lo tanto la fricción no es un problema.

Sin embargo, para conseguir una lectura de posición angular la lectura del sensor velocidad debe ser integrada. Lamentablemente cualquier error en la lectura de velocidad también será integrada, dando por resultado que quede a la deriva similar a los giroscopios de rueca. La deriva mínima del giroscopio encontrada en sistemas profesionales es 3°/min. es decir si usted se queda parado, su mundo virtual dará vuelta completamente en dos horas. Mientras que esto no será sensible para el yaw, el tener su mundo virtual de cabeza después de una hora debido a deriva en el giroscopio (pitch) es probable que no sea adecuado; los seguidores básicos de giroscopios tratan de librarse de este problema calibrando los giroscopios antes de ser usados.

Los sistemas más exactos utilizan una tecnología diferente de seguimiento para reajustar las lecturas del giroscopio en intervalos uniformes de tiempo.

2.3.12. Dispositivos Inerciales.

MotionPak es un producto de Systron-Donner (ver la Figura 2.20), es un sistema de sensores de inercia de 6 DOF, de estado sólido usado, para medir aceleraciones lineales y velocidades angulares. Es el seguidor idóneo para el monitoreo del movimiento humano. Utiliza tres sensores micromecanizados de cuarzo de velocidad angular montados ortogonalmente y tres servo aceleradores lineales de alto rendimiento montados en un compacto paquete, los detalles de las especificaciones se dan en la Tabla 2.11

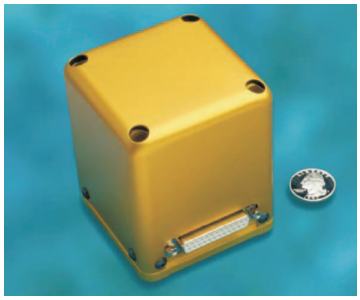


Figura 2.20: MotionPak.

Tabla 2.11: MotionPak.

Especificación	Valor
Velocidad de actualización	60Hz
Latencia	No la da
Exactitud	1.25° heading, 0.25° inclinación
Resolución Heading	0.15°
Resolución Angular	0.004°
Volumen de trabajo	360° horizontal, 55° inclinación
Precio	$10,000 USD

Conclusiones del capítulo

Éste capítulo tuvo como objetivo dar a conocer las diferentes tecnologías y dispositivos, más usados en el mercado, con sus ventajas y desventajas tanto técnicas como económicas y que ubican al seguidor electromecánico como ideal para el presente trabajo, por la facilidad de modelación, por su gran exactitud y comparativamente por el bajo costo de realización.

Capítulo 3

Sistema seguidor electromecánico

Resumen.

En este capítulo se describen el seguidor electromecánico y sus características; más adelante se describe la forma de obtener modelo cinemático directo de posición, por medio del algoritmo de Denavit-Hartenberg y el uso de matrices de transformación homogénea. Por último se realiza una simulación del modelo.

3.1. Introducción.

La cinemática de un robot estudia su movimiento del mismo con respecto a un sistema de referencia. Así, la cinemática de un robot se interesa por la descripción analítica del movimiento como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares.

En el presente trabajo, se propone la construcción de un robot antropomórfico de cadena serial, en el cual, en vez del actuador (pinzas), se conecta un casco que se sujeta a la cabeza de la persona, esto se realizará con finalidad utilizar el robot como un seguidor electromecánico.

3.2. Modelo cinemático.

El **problema cinemático** consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot (ver la Figura 3.1).

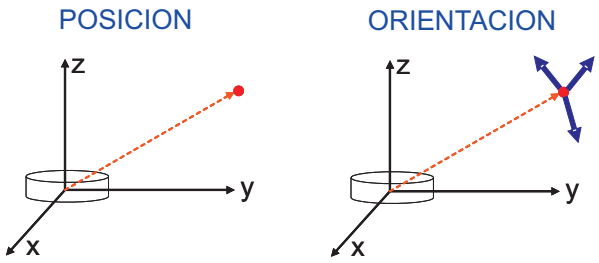


Figura 3.1: Movimiento del efector final.

Denavit-Hartenberg propusieron un método sistemático para describir y representar la geometría espacial de los elementos de una cadena cinemática, y en particular de un robot, con respecto a un sistema de referencia fijo. Este método utiliza una matriz de transformación homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciéndose el problema cinemático directo a encontrar una matriz de transformación homogénea $4X4$ que relacione la localización espacial del extremo (actuador) con respecto al sistema de coordenadas de su base.

Metodología para la obtención del modelo cinemático directo de posición (MCDP)

La metodología para obtener el MCDP se basa en la definición de:

1. La cadena cinemática.
2. Los marcos ortonormales de referencia.
3. Los parámetros de Denavit-Hartenberg.
4. Las matrices elementales.
5. La matriz de transformación homogénea generalizada.
6. El MCDP en términos de:

   Coordenadas cartesianas para posición.

   Ángulos de Euler para orientación.

3.3. Cadena cinemática

La cadena cinemática está constituida por los eslabones, las articulaciones y en su extremo final por un órgano terminal o efector final. Los eslabones y articulaciones se enumeran comenzando por 1 hasta n (número de grados de libertad), considerando a la base fija del robot como el eslabón cero (ver la Figura 3.2).

Existen reglas para construir cadenas cinemáticas de manipuladores, tal como las utilizadas para describir la del robot antropomórfico (Cf. [Anibal Ollero, 2001]).

3.4. Marcos ortonormales

La definición de un marco ortonormal, permite tener una referencia para conocer la situación de cada uno de los eslabones y del efector final, con respecto a la base (referencia fija).
Los marcos de referencia se construyen bajo los siguientes criterios.

1. A los ejes de giro de las articulaciones, se designarán como ejes $\hat{Z}$
2. El origen O se localiza en la normal común a los ejes de las articulaciones $\hat{Z}_1$ y $\hat{Z}_{i+1}$.
   Si los ejes de las articulaciones son paralelos o están alineados, la perpendicular común se selecciona arbitrariamente.
   Es recomendable seguir un criterio de simplicidad.
3. El vector $\hat{X}_{i+1}$ se define sobre la perpendicular común a los ejes de las articulaciones $\hat{Z}_i$ y $\hat{Z}_{i+1}$; es decir por el producto cruz: $\hat{Z}_i \times \hat{Z}_{i+1}$.
   Si los ejes de las dos articulaciones están alineados, la orientación de $\hat{X}_{i+1}$ es arbitraria.
4. $\hat{Y}_{i+1}$se define de tal manera que se complete un marco dextrogiro, es decir:
   $\hat{Y}_{i+1} = \hat{Z}_{i+1} \times \hat{X}_{i+1}$

Los marcos de referencia y la cadena cinemática resultante se muestra en la Figura 3.2

3.5. Parámetros Denavit-Hartenberg (DH)

Los parámetros DH (Cf. [Jorge Angeles, 1996]) son cuatro términos geométricos asociados con cada elemento que permiten describir una articulación prismática o de revolución y se definen a continuación :

θ_i : Es el ángulo de rotación alrededor del eje $\hat{Z}_{i-1}$ medido de $\hat{X}_{i-1}$ a $\hat{Y}_{i-1}$, el sentido está definido por la regla de la mano derecha.

d_i : Es la distancia de traslación a lo largo del eje $\hat{Z}_i$.

a_i : Es la distancia de traslación sobre el eje $\hat{X}_i$

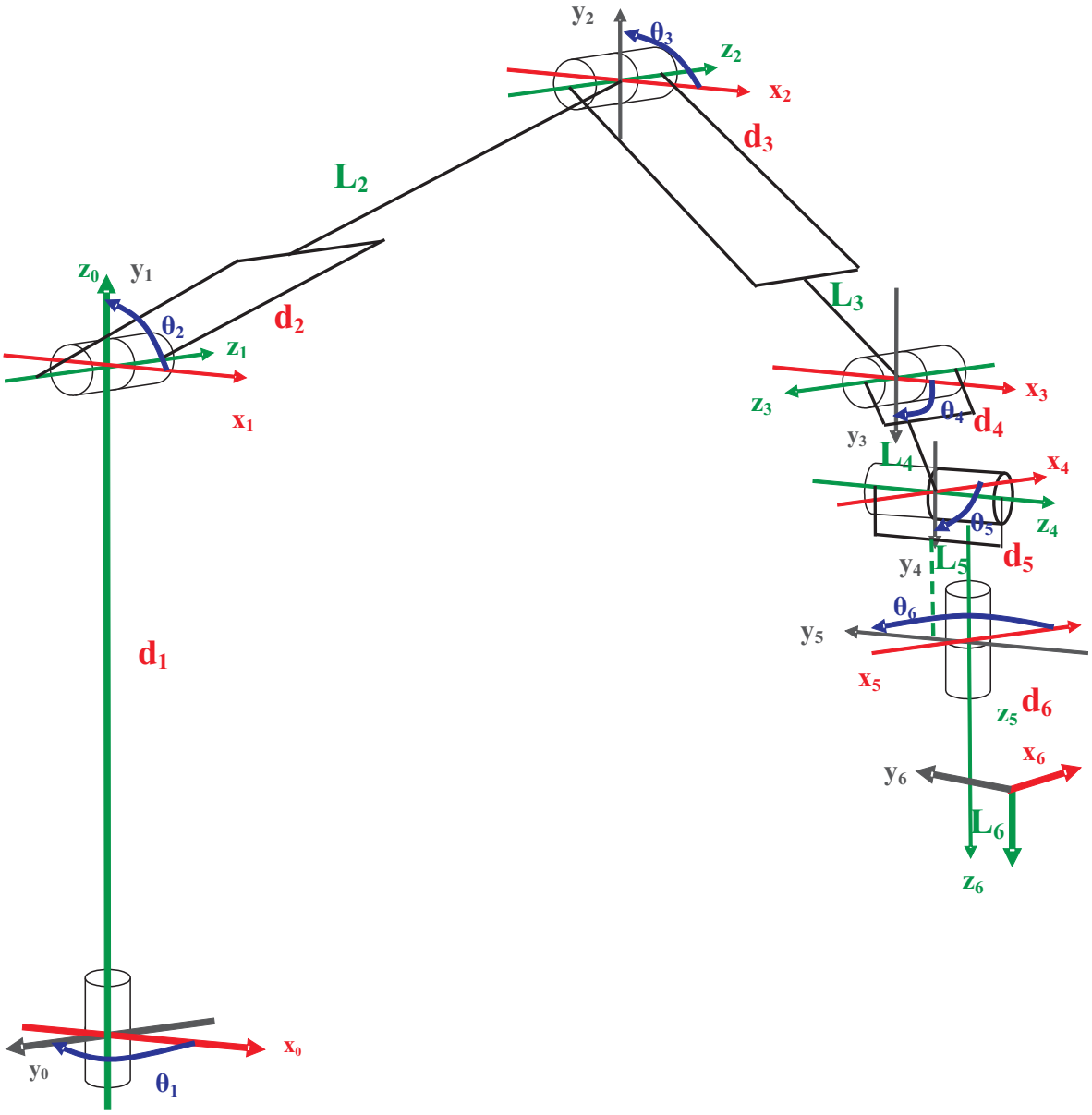


Figura 3.2: Cadena cinemática y marcos ortonormales.

Tabla 3.1: Parámetros Denavit-Hartenberg

Eslabón i	θ_i	d_i	a_i	α_i
1	θ_1	d_1	0	$-\frac{\pi}{2}$
2	θ_2	d_2	L_2	0
3	θ_3	d_3	L_3	0
4	θ_4	d_4	L_4	$\frac{\pi}{2}$
5	θ_5	d_5	L_5	$-\frac{\pi}{2}$
6	θ_6	d_6	L_6	0

α_i : Es el ángulo de separación del eje $\hat{Z}_i$ al eje $\hat{Z}_{i+1}$ medido respecto a $\hat{Z}_{i+1}$

Los parámetros DH que se obtienen se presentan en la Tabla 3.1

3.6. Matrices elementales

La matriz de transformación elemental se obtiene sustituyendo los parámetros Denavit-Hartenberg en (3.1)
.

$$ {}^{i}_{i+1}T = \begin{bmatrix} cos\theta_i & -sen\theta_i cos\alpha_i & sen\theta_i sen\alpha_i & a_i cos\theta_i \\ sen\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sen\alpha_i & a_i sen\theta_i \\ 0 & sen\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1} $$

Sustituyendo los parámetros DH se obtienen las matrices elementales : (3.2), (3.3), (3.4), (3.5), (3.6) y (3.7),

$$ {}^{1}_{0}T = \begin{bmatrix} Cos\theta_1 & 0 & -Sen\theta_1 & 0 \\ Sen\theta_1 & 0 & Cos\theta_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2} $$

$$ {}^{2}_{1}T = \begin{bmatrix} Cos\theta_2 & -Sen\theta_2 & 0 & L_2 Cos\theta_2 \\ Sen\theta_2 & Cos\theta_2 & 0 & L_2 Sen\theta_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3} $$

$$ {}^{3}_{2}T = \begin{bmatrix} Cos\theta_3 & -Sen\theta_3 & 0 & L_2Cos\theta_3 \\ Sen\theta_3 & Cos\theta_3 & 0 & L_3Sen\theta_3 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4} $$

$$ {}^{4}_{3}T = \begin{bmatrix} Cos\theta_4 & 0 & Sen\theta_4 & L_4Cos\theta_4 \\ Sen\theta_4 & 0 & -Cos\theta_4 & L_4Sen\theta_4 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5} $$

$$ {}^{5}_{4}T = \begin{bmatrix} Cos\theta_5 & 0 & Sen\theta_5 & L_5Sen\theta_5 \\ -Sen\theta_5 & 0 & Cos\theta_5 & -L_5Cos\theta_5 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6} $$

$$ {}^{6}_{5}T = \begin{bmatrix} Cos\theta_6 & -Sen\theta_6 & 0 & L_6Cos\theta_6 \\ Sen\theta_6 & Cos\theta_6 & 0 & L_6Sen\theta_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7} $$

3.7. Matriz de transformación homogénea generalizada

La matriz de transformación homogénea generalizada se obtiene multiplicando las 6 matrices elementales, como se indica en (3.8)
:

$$ T^6_0 = \left[{}^{1}_{0}T\right] \left[{}^{2}_{1}T\right] \left[{}^{3}_{2}T\right] \left[{}^{4}_{3}T\right] \left[{}^{5}_{4}T\right] \left[{}^{6}_{5}T\right] \tag{3.8} $$

Empleando identidades trigonométricas para la suma de ángulos y realizando simplificaciones en (3.8) se obtiene (3.9)

$$ {}^{6}_{0}T = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9} $$

Para facilitar la escritura, se emplea: $Cos\theta_1 = C_1$, $Cos\theta_2 = C_2$, $Cos\theta_5 = C_5$, $Cos\theta_6 = C_6$, $Sen\theta_1 = S_1$, $Sen\theta_2 = S_2$, $Sen\theta_2 = S_2$, $Sen\theta_5 = S_5$, $Sen\theta_6 = S_6$, $Cos(\theta_2+\theta_3) = C_{23}$, $Sen(\theta_2+\theta_3) = S_{23}$, $Cos(\theta_2+\theta_3+\theta_4) = C_{234}$ y $Sen(\theta_2+\theta_3+\theta_4) = S_{234}$.

$$t_{11} = [C_1C_{234}C_5 - S_1S_5]C_6 - C_1S_{234}S_6$$

$$t_{21} = [S_1C_{234}C_5 + C_1S_5]C_6 - S_1S_{234}S_6$$

$$t_{31} = -S_{234}C_5C_6 - C_{234}S_6$$

$$t_{12} = [-C_1C_{234}C_5 + S_1S_5]S_6 - C_1S_{234}C_6$$

$$t_{22} = -(S_1C_{234}C_5 + C_1S_5)S_6 - S_1S_{234}C_6$$

$$t_{32} = S_{234}C_5S_6 + C_{234}C_6$$

$$t_{13} = -C_1C_{234}S_5 - S_1C_5$$

$$t_{23} = -S_1C_{234}S_5 + C_1C_5$$

$$t_{33} = S_{234}S_5$$

$$t_{14} = C_1[(C_{234}C_5 + S_1S_5)C_6 + (S_{23}C_4 - C_{23}S_4)S_6]L_6 + C_1[C_{234}S_5 - S_1C_5]d_6$$
$$+C_1C_{234}L_5C_5 - S_1L_5S_5 + C_1S_{234}d_5 + C_1C_{234}L_4 - S_1d_4 + C_1L_3C_{23} - S_1d_3$$
$$+C_1L_2C_2 - S_1d_2$$

$$t_{24} = S_1[(C_{234}C_5 - C_1S_5)C_6 + (S_{23}C_4 - C_{23}S_4)S_6]L_6 + S_1[C_{234}S_5 + C_1C_5]d_6$$
$$+S_1C_{234}L_5C_5 + C_1L_5S_5 + S_1S_{234}d_5 + S_1C_{234}L_4 + C_1d_4 + S_1L_3C_{23} + C_1d_3$$
$$+S_1L_2C_2 + C_1d_2$$

$$t_{34} = [C_{234}S_6 - S_{234}C_5C_6]L_6 - S_{234}S_5d_6 - S_{234}L_5C_5 + C_{234}d_5 - S_{234}L_4$$
$$-L_3S_{23} - L_2S_2 + d_1$$

3.8. Modelo cinemático directo de posición (MCDP)

El MCDP es el conjunto de ecuaciones que proporcionan las coordenadas operacionales a partir de los ángulos de las articulaciones, así como los ángulos de Euler al primer sistema de referencia para hacerlo coincidir con el último.

3.8.1. Coordenadas cartesianas para posición

La posición del elemento terminal en coordenada del sistema base p_x, p_y y p_z esta dado directamente por coeficientes de la matriz de transformación generalizada los cuales son t_{14}, t_{24}, t_{34} respectivamente.

$$p_x = C_1[(C_{234}C_5 + S_1S_5)C_6 + (S_{23}C_4 - C_{23}S_4)S_6]L_6 + C_1[C_{234}S_5 - S_1C_5]d_6 \quad (3.10)$$
$$+C_1C_{234}L_5C_5 - S_1L_5S_5 + C_1S_{234}d_5 + C_1C_{234}L_4 - S_1d_4 + C_1L_3C_{23} - S_1d_3$$
$$+C_1L_2C_2 - S_1d_2$$

$$p_y = S_1[(C_{234}C_5 - C_1S_5)C_6 + (S_{23}C_4 - C_{23}S_4)S_6]L_6 + S_1[C_{234}S_5 + C_1C_5]d_6 \quad (3.11)$$
$$+S_1C_{234}L_5C_5 + C_1L_5S_5 + S_1S_{234}d_5 + S_1C_{234}L_4 + C_1d_4 + S_1L_3C_{23} + C_1d_3$$
$$+S_1L_2C_2 + C_1d_2$$

$$p_z = [C_{234}S_6 - S_{234}C_5C_6]L_6 - S_{234}S_5d_6 - S_{234}L_5C_5 + C_{234}d_5 - S_{234}L_4 \quad (3.12)$$
$$-L_3S_{23} - L_2S_2 + d_1$$

3.8.2. Ángulos de Euler para orientación

Los ángulos de Euler se definen para la orientación mediante tres giros consecutivos en los tres ejes de coordenadas. Realizando de forma diferente los giros, se obtienen otras representationes. En el caso de los denominados **ángulos de Euler Z-Y-Z**, se

comienza con B coincidente con la referencia A y se efectúa sucesivamente la rotación de B alrededor de Z_B un ángulo α, y a continuación alrededor de $Y_{B'}$ un ángulo β. Por último, la tercera rotación es diferente ya que ahora se vuelve a rotarse alrededor de $Z_{B''}$ un ángulo γ.

Obteniendo una matriz de rotación:

$$ {}^{B}_{A}R_{zyz}(\alpha,\beta,\gamma) = \begin{bmatrix} C\alpha C\beta C\gamma - S\alpha S\gamma & -C\alpha C\beta S\gamma - S\alpha C\gamma & C\alpha S\beta \\ S\alpha C\beta C\gamma + C\alpha S\gamma & -S\alpha C\beta S\gamma - C\alpha C\gamma & S\alpha S\beta \\ -S\beta C\gamma & S\beta C\gamma & C\beta \end{bmatrix} \tag{3.13} $$

El problema inverso se resuelve ahora mediante las expresiones:

$$ \begin{aligned} \gamma &= Arct2(t_{32}, -t_{31}); \\ \beta &= Arct2(\sqrt{t_{31}^2 + t_{32}^2}, t_{33}); \\ \alpha &= Arct2(t_{23}, t_{13}) \end{aligned} \tag{3.14} $$

suponiendo $Sen\beta \neq 0$. El ángulo β se elige de forma que $0 \leq \beta \leq 180°$, que corresponde a la raíz positiva. La solución es degenerada cuando $\beta = 0$ o $\beta = 180°$.

Para obtener la matriz de rotación se obtiene multiplicando las tres últimas matrices.

$$ T_3^6 = \left[{}^4_3T\right]\left[{}^5_4T\right]\left[{}^6_5T\right] \tag{3.15} $$

$$ {}^3_6T = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5S_6 - S_4C_6 & C_4S_5 & C_4 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 & S_4S_5 \\ -S_5S_6 & S_5S_6 & C_5 & C_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.16} $$

Por lo tanto comparando la (3.16) con los ángulos de Euler, de (3.13), se muestra que las tres últimas articulaciones θ_4, θ_5 y θ_6 son los ángulos de Euler (α, β, γ), respectivamente, con respecto marco $O_3X_3Y_3Z_3$.

3.9. Simulación del MCDP

En esta sección se desarrollaron pruebas para simular el modelo cinemático directo de posición (MCDP) de las 6 articulaciones del robot antropomórfico. Todas las pruebas realizaron con un giro de 180° por cada una de las articulaciones, el cual es superior a

giro que puede realizar el movimiento de la cabeza como pitch roll y yaw.

La simulación como se comentó en la Sección 1.10, utiliza el programa de Matlab version 7. La simulación de la posición y orientación se puede ver en el Apéndice B y C respectivamente.

3.9.1. Posición X Y Z

Esta simulación consiste en calcular la posición de la vision de la persona que tiene colocado el casco, con respecto a un sistema de referencia (X_0, Y_0, Z_0) y graficar resultados.

El desarrollo de la simulación se realizó de la siguiente forma:

1. Resolver la matriz de transformación homogénea en forma algebraica (**Apéndice A**).
2. Seleccionar el vector posición de la matriz.
3. Establecer condiciones iniciales:

   $\theta_1 = 0°$

   $\theta_2 = 0°$

   $\theta_3 = 0°$

   $\theta_4 = 90°$

   $\theta_5 = 90°$

   $\theta_6 = -90°$
4. Girar las articulaciones:.

   $0° < \theta_1 < 180°$

   $0° < \theta_2 < 180°$

   $0° < \theta_3 < 180°$

   $0° < \theta_4 < 180°$

   $0° < \theta_5 < 180°$

   $0° < \theta_6 < -180°$
5. Calcular

   px py pz

   Gráficar (ver la Figura 3.3)

6. Resultados

   Al revisar la información proporcionada por las gráficas (la posición en función del giro de la articulación); y al hacer una evaluación practica con el robot antropomórfico, se puede concluir que es valido MCDP, debido a que las coordenadas presentadas son las mismas que coordenadas generadas por la simulación.

3.9.2. Orientación $\alpha\ \beta\ \gamma$

Esta simulación consiste en calcular la orientación de la vision de la persona que tiene colocado el casco, con respecto a un sistema (X_3, Y_3, Z_3) de referencia y graficar resultados.

El desarrollo de la simulación se realizó de la siguiente forma:

1. Resolver la matriz de transformación homogénea en forma algebraica (**Apéndice B**).
2. Seleccionar la matriz de rotación.
3. Establecer condiciones iniciales:

   $\theta_4 = 0°$

   $\theta_5 = 0°$

   $\theta_6 = 0°$
4. Girar las articulaciones:

   $0° < \theta_4 < 180°$

   $0° < \theta_5 < 180°$

   $0° < \theta_6 < -180°$
5. Calcular:

   $\alpha\ \beta\ \gamma$

   Gráficar (ver la Figura 3.4)
6. Resultados

   Observando la información proporcionada por las gráficas (los ángulos de orientación en función del giro de una articulación) y realizando una evaluación practica con el robot antropomórfico, se concluye que son validos los ángulos, debido a que la orientación presentada es la misma que los ángulos de Euler generados por la simulación.

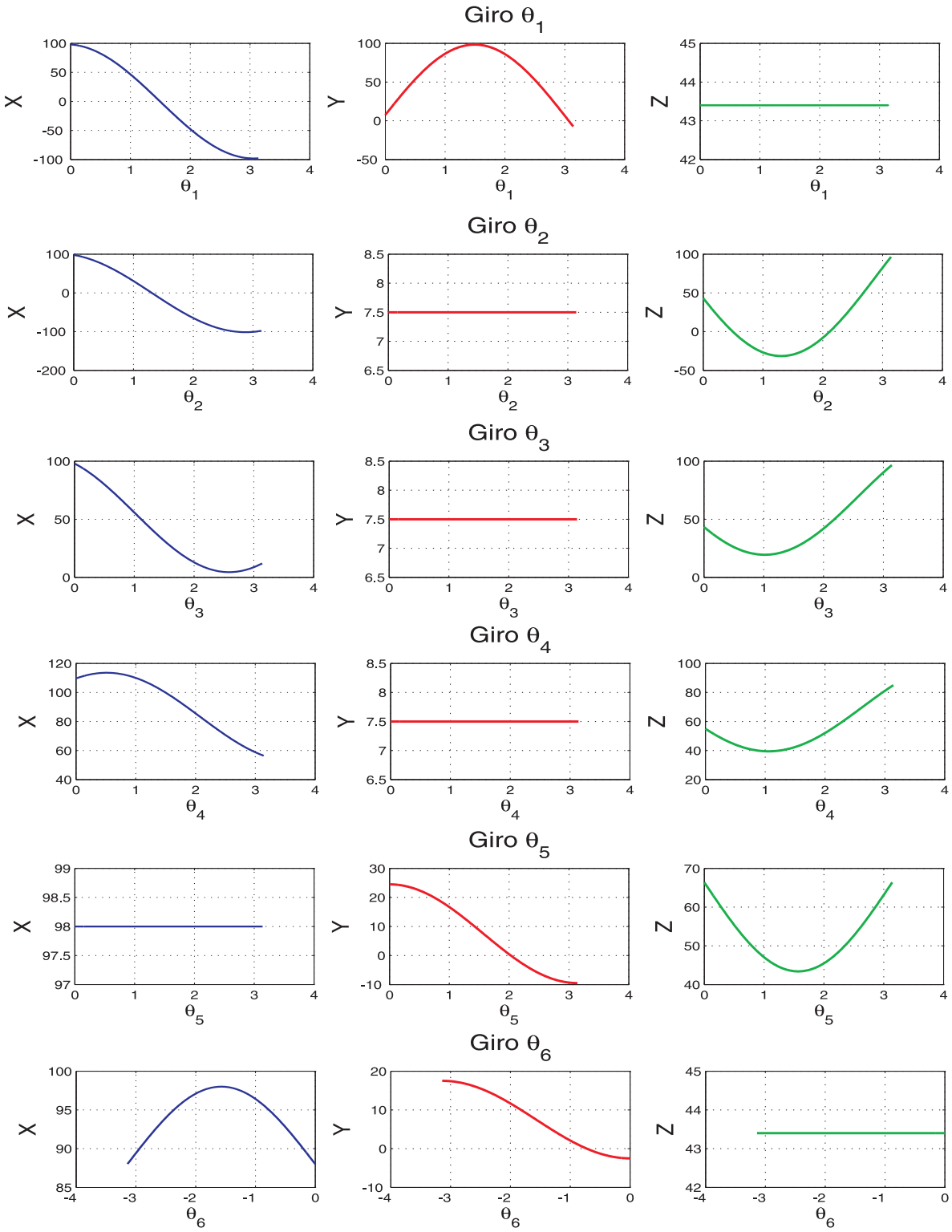


Figura 3.3: Simulación de la Posición.

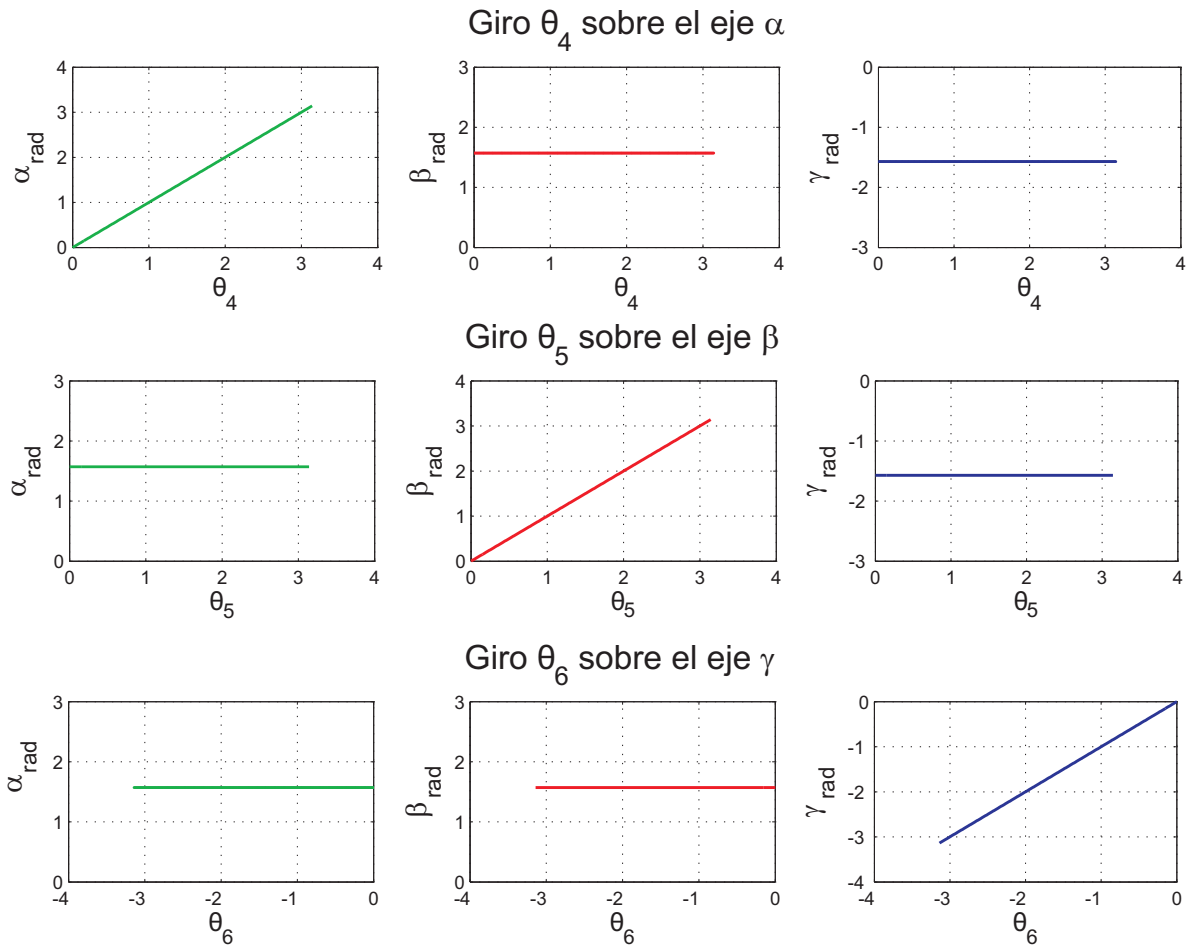


Figura 3.4: Simulación de la Orientación.

Capítulo 4

Diseño y simulación del sistema de adquisición de variables cinemáticas

Resumen.

En el siguiente capitulo se hace una descripción de las necesidades y especificación que comprende este proyecto, del sistema de adquisición (detallando cada una de las partes que lo componen así como sus dispositivos interconectados); posteriormente se explica la metodología de programación del PIC. Todo esto capitulo se desarrolla, con la finalidad de realizar la simulación del PIC16F877.

4.1. Introducción.

En la actualidad el vertiginoso desarrollo de la electrónica y la microelectrónica han motivado que todas las esferas de la vida humana se estén automatizando, por ejemplo: la industria, el hogar, los comercios, la agricultura, la ganadería, el transporte, las comunicaciones, etc. En todo ese proceso de automatización el microprocesador y el microcontrolador juegan un papel de suma importancia. Ellos han permitido el desarrollo de sistemas inteligentes que resuelven los mas diversos problemas, son los llamados Sistemas de Adquisición de Datos.

Existen muchas aplicaciones en las que los datos analógicos se deben digitalizar (convertir a digitales) y transferir a la memoria de una computador. Al proceso mediante el cual la computadora adquiere estos datos analógicos y los digitaliza se le denomina **adquisición de datos**.

4.2. Identificación de necesidades y especificaciones.

4.2.1. Necesidades

.

Se propone que el sistema a diseñar sea un sistema de adquisición de datos de 6 canales de entrada analógicos basado en un PIC16F877. Como el uso del sistema de adquisición de datos será análogo al de un instrumento de medición de voltaje de DC, se requiere determinar las especificaciones del diseño para el sistema y para la variable a medir.

4.2.2. Especificaciones

.

Las especificaciones a cumplir en el diseño son:

- Canales de entrada analógicos - Analog Inputs (AI): 6 canales y cada canal medirá la rotación de articulación. Esta especificación es propuesta del proyecto para tener la medición de la variable.
- Canales de entrada/salida digitales DI/DO: No tendrá.
- Contadores: No tendrá.
- Resolución del CAD: 10 bits por especificación del PIC16F877
- Comunicación: serie asíncrona RS-232 por especificación que sea una velocidad mínima de 9600 baudios.
- Voltaje DC: Medir en el rango de 0 a 5 V DC. Esta especificación se da porque la fuente de alimentación.

4.3. Diseño del sistema de adquisición de variables cinemáticas

Para el análisis del diseño del sistema de adquisición se divide en tres etapas:

1. **Hardware:**

   Variables cinemáticas de posición 3D y orientación del seguidor.

2. **Software:**

   PIC16F877.

   Ambiente virtual PC.

3. **Simulación del PIC.**

4.4. Variables cinemáticas de posición 3D y orientación del seguidor

A continuación se muestra el sistema referencial del casco para virtualización (ver la Figura 4.1).

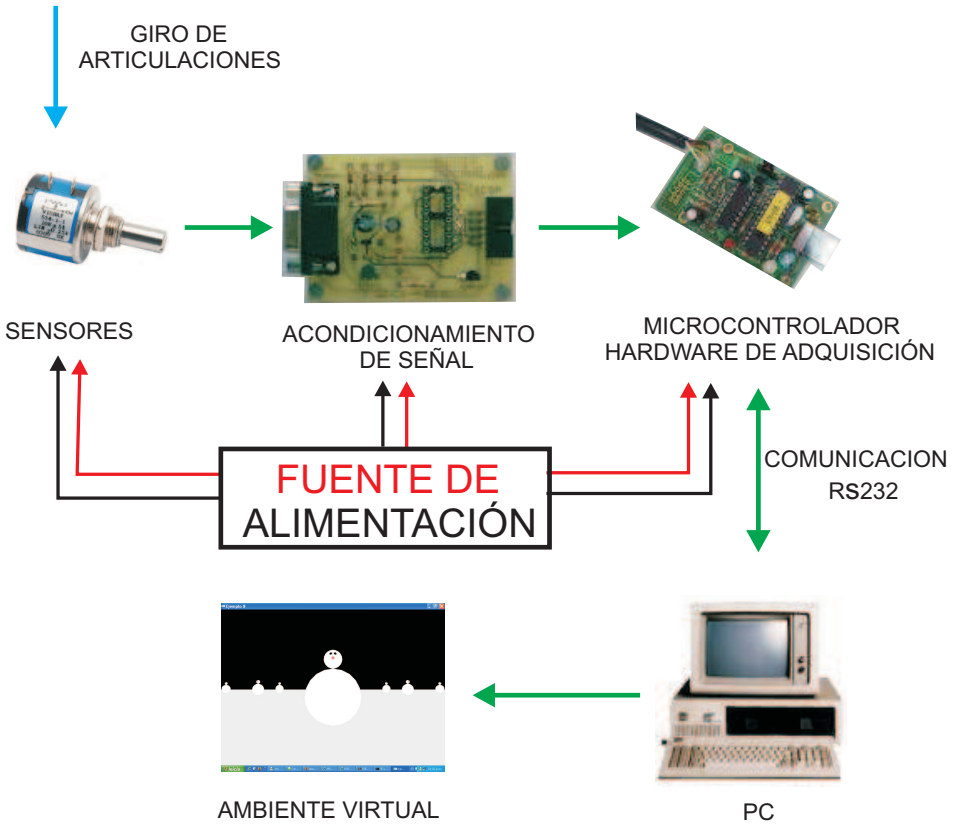


Figura 4.1: Diagrama a bloques del sistema de adquisición de datos.

Las articulaciones tienen un vector $\Theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$ el cual debe ser medido para poder obtener a través de la trasformación vista el Sección 3.8, la posición y orientación en que se encuentra el casco para virtualización.

Para lo anterior es necesario:

- **Módulo de medición** del ángulo de las articulaciones con respecto al eslabón anterior.
- **Módulo de configuración de señal.**

- **Módulo de conversión A/D**.
- **Interfaz del seguidor** entre dicho sistema y la PC, en donde se encuentra el casco en un mundo virtual que sera accionado por el casco para virtualización (ver la Figura 4.2).
- **Módulo de fuente potencia**

4.4.1. Medición

En esta primera etapa del sistema de adquisición de datos se refiere a que los sensores o transductores se usarán para medir la variable física. De acuerdo a las especificaciones de diseño la variable a medir es un voltaje DC.

El elemento sensor a utilizar es un potenciómetro R_1, el cual tiene un contacto deslizante que puede desplazarse a lo largo de una resistencia. Esta característica del potenciómetro se utiliza en el desplazamiento rotacional que tiene las articulaciones del robot antropomórfico; por ultimó dicho desplazamiento se convierte en una diferencia de potencial. En resumen el potenciómetro tiene como entrada un movimiento rotacional y como salida una diferencia de potencial.

Para este módulo se propone utilizar dos potenciómetros una de presición R_1 y uno lineal para el ajuste voltaje de referencia para los seis potenciómetros R_2. Conectados en serie con un voltaje de 5 Volt (Circuito divisor de tensión).

Para entender la manera de cómo podemos analizar un circuito divisor de tensión se analiza desde el punto de vista eléctrico como se describe a continuación:

La división de tensión ocurre cuando una fuente dependiente o independiente de tensión se conecta en serie con dos resistencias, como se ilustra en en la Figura 4.3.

Evidentemente la tensión a través de R_2 es:

De acuerdo a la **Ley de Ohm** (4.1):

$$V_T = I_T R_T \tag{4.1}$$

Sustituimos I_T para obtener V_2 (4.2):

$$V_2 = R_2 I_T = R_2 \frac{V_T}{R_1 + R_2} \tag{4.2}$$

Reduciendo queda (4.4):

Figura 4.2: Prototipo.

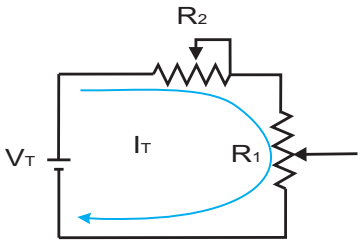


Figura 4.3: Divisor de tensión.

$$V_2 = \frac{R_2 V_T}{R_1 + R_2} \tag{4.3}$$

Y análogamente, la tensión a través de R_1 es (4.4):

$$V_1 = \frac{R_1 V_T}{R_1 + R_2} \tag{4.4}$$

Esto nos indica que para un circuito en serie, la tensión que existe en cualquier resistor (o combinación de resistores en serie) es igual al valor de ese resistor (o a la suma de dos o más resistencias en serie) multiplicando por la diferencia de potencias de todo el circuito en serie y dividido entre las resistencia total del circuito.

Se propone utilizar un potenciómetro de $2K\Omega$ de precisión y potenciómetro lineal de 100Ω para el ajuste del voltaje referencia.

De acuerdo a la fórmula (4.1), la corriente que fluye a través del divisor de tensión se obtiene despejando y sustituyendo valores. Debido a que V_T es el voltaje de alimentación de 5 V DC y R_T es la suma de los potenciómetros en serie a su máxima escala $R_1 + R_2$, del divisor de tensión obtenemos (4.5).

$$I = \frac{V_T}{R_T} = \frac{5V}{3K\Omega} = 1{,}6mA \tag{4.5}$$

El valor de corriente total obtenido para el circuito de la fuente de señal es de $2{,}3mA$, más bajo que lo que pide especificación del microcontrolador PIC16F877 de una corriente de 25 mA.

Entonces, para obtener la tensión R_2 utilizamos la ecuación (4.6)

$$V_2 = R_2 \frac{V_T}{R_1 + R_2} = \frac{1K\Omega X 5V}{3K\Omega} = 1{,}667V \tag{4.6}$$

Con este resultado de 1.667 V podemos ajustar el potenciómetro R_2 a un valor de 1 V , para obtener un voltaje referencia de $4V$ para cada uno de los sensores.

4.4.2. Configuración de señal

Como la salida del divisor de tensión utilizado como sensor de voltaje DC no es necesario amplificar la señal debido a que el valor máximo de esta es de 5V, el cual es el valor máximo aceptado por el PIC16F877 para sus entradas analógicas.

Como se muestra en el diagrama eléctrico (ver la Figura 4.4), la salida del divisor de tensión utilizado como sensor de voltaje DC, solamente se tiene un seguidor de voltaje no inversor para mantener constante el voltaje de salida del mismo (especificaciones del TL081A en el Apéndice C).

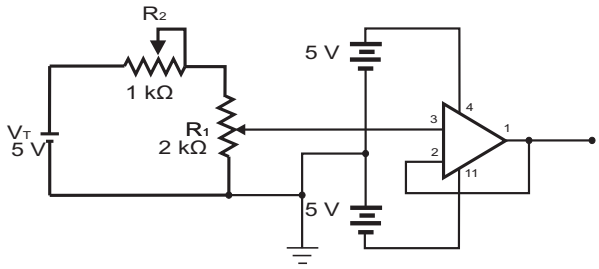


Figura 4.4: Diagrama eléctrico del acondicionador de señal de voltaje DC.

4.4.3. Conversión A/D e interfaz serial

El microcontrolador del PIC16F877 de Microchip pertenece a una gran familia de microcontroladores de 8 bits (bus de datos) que tienen las siguientes características generales que los distinguen de otras familias:

- Arquitectura Harvard.
- Tecnología RISC.
- Tecnología CMOS.

De acuerdo a la propuesta se utilizará un PIC16F877. Se decidió utilizar este microcontrolador por las características y periféricos con los que cuenta y fueron ocupados para el desarrollo del sistema. Estas características y periféricos son los siguientes:

- Solo 35 instrucciones.
- Frecuencia de operación de 0 a 20 MHz.
- Rango de operación de 0 a 5 V DC.

- Oscilador 4 MHz.
- CAD de 10 bits, hasta 8 canales.
- 8 canales de entrada analógicos.
- Módulo de comunicación Universal Synchronous Asynchronous Receiver Transmitter (USART).
- Fuentes de interrupción por conversión analógica-digital, recepción y transmisión de datos.

Estas características se conjugan para lograr un dispositivo altamente eficiente en el uso de la memoria de datos y programa y, por lo tanto en la velocidad de ejecución.

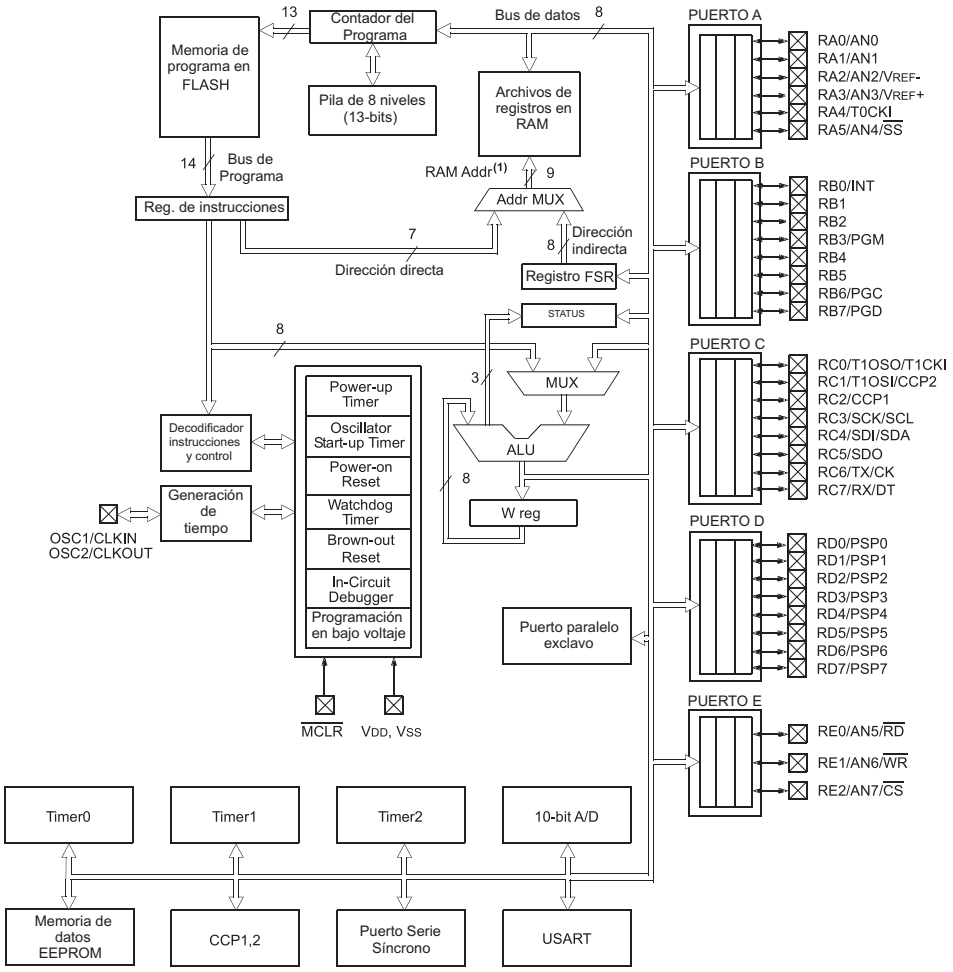


Figura 4.5: Arquitectura del microcontroladores PIC16F877.

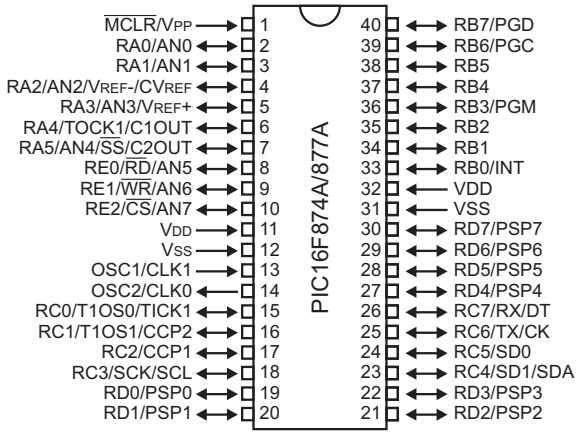


Figura 4.6: Diagrama del microcontrolador PIC16F877.

La arquitectura aplicada en microcontrolador PIC16F877 (ver Figura 4.5), se caracteriza por la independencia entre la memoria de código y la de datos. También se muestra en la Figura 4.6 su diagrama de terminales, para tener una visión conjunta de interior y exterior del Chip.

En el **Apéndice D**, se pueden ver más características del microcontrolador PIC16F877.

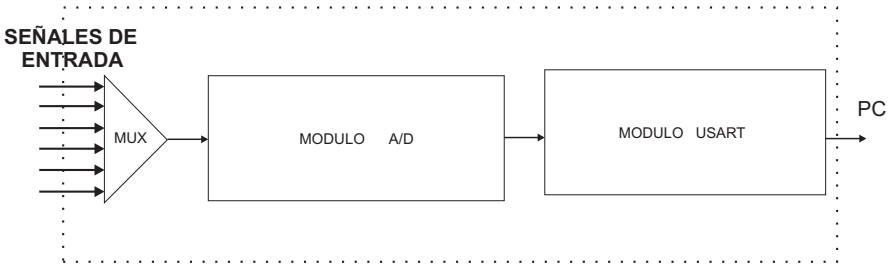


Figura 4.7: Proceso del microcontrolador en la Etapa de Adquisición.

En el siguiente diagrama a bloques de la Figura 4.7 se pueden ver el proceso que lleva a cabo el microcontrolador. Todo esto se lleva dentro del microcontrolador utilizando algunos de los periféricos que contiene este.

Estos periféricos son:

- Convertidor (A/D).
- USART **(nota)**.

nota: `Este periférico se explica en el módulo de comunicación serial RS-232.`

Módulo del convertidor (A/D)

El módulo del convertidor analógico a digital (A/D) tiene ocho entradas. La entrada analógica carga una muestra y la sostiene en el capacitor que la retiene son la entrada para el convertidor. Entonces el convertidor genera un resultado digital de este valor analógico a través de aproximaciones sucesivas. La conversión A/D de la señal de entrada analógica resulta en número digital de 10 bits.

El módulo A/D tiene cuatro registros que son:

- Registro de resultado alto A/D (ADRESH).
- Registro de resultado bajo A/D (ADRESL).
- Registro 0 de control A/D (ADCON0).
- Registro 1 de control A/D (ADCON1).

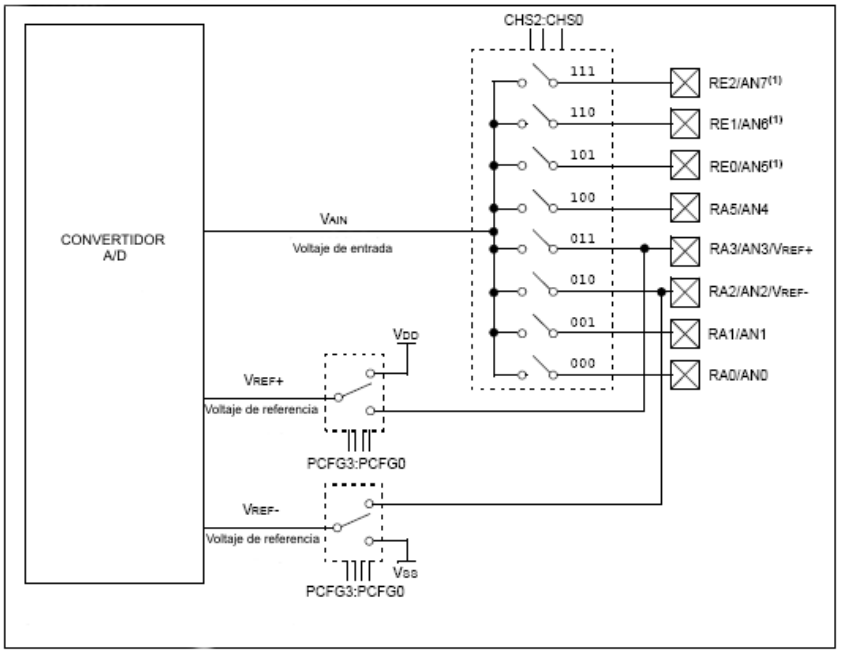


Figura 4.8: Diagrama del convertidor A/D.

Interfaz serial RS-232

La comunicación serie requiere sólo una línea para la transmisión esto abarata los costos en líneas de transmisión y no sólo esto, ya que este hecho también hace posible que los datos puedan ser enviados no necesariamente por un conductor eléctrico, sino inclusive por aire o por vacío si en lugar de pulsos eléctricos se usan impulsos electromagnéticos, tales como: ondas de radio, microondas, pulsos luminosos, infrarrojo, ultrasonido, láser (a través de fibra óptica), etc.

La gran mayoría de los sistemas de comunicación de datos actuales utilizan la comunicación en serie, debido a las grandes ventajas que representa esta manera de comunicar los datos:

- **Económica.-** Utilizan pocas líneas de transmisión inclusive puede usar sólo una línea.
- **Confiable.-** Los estándares actuales permiten transmitir datos con bits de paridad y a niveles de voltaje o corriente que los hacen poco sensibles a ruido externo. Además por tratarse de información digital, los cambios en amplitud de las señales (normalmente causadas por ruido) afectan muy poco o nada a la información
- **Versátil.-** No está limitada a usar conductores eléctricos como medio de transmisión, pudiendo usarse también: fibra óptica, aire, etc. Además el tipo de energía utilizada puede ser diferente: luz visible, infrarroja, ultrasonido, pulsos eléctricos, radio frecuencia , microondas, etc.

A diferencia de la comunicación en paralelo, en la comunicación en serie se hace necesario establecer métodos de sincronización para evitar la interpretación errónea de los datos transmitidos.

Existen dos formas de comunicación serial:

1. Síncrona
2. Asíncrona

Método síncrono: Cada mensaje o bloque de transmisión va precedido de unos caracteres de sincronismo. Así, cuando el receptor identifica una configuración de bits igual a la de los caracteres de sincronismo da por detectado el inicio y el tamaño de los datos.

Método asíncrono: Cada carácter va señalizado mediante dos bits: un bit de inicio y un bit de paro, estos dos bits permiten al receptor reconocer el inicio y el final de

cada carácter.

La comunicación a emplear en este módulo es serial asíncrona utilizando el protocolo RS-232, en el que se tienen a los siguientes elementos:

1. USART
2. MAX232

USART

La USART también conocida como Serial Communication Interface (SCI) puede configurarse como una unidad de comunicación en serie para la transmisión de datos asíncrona con dispositivos tales como terminales de computadora o computadoras personales, o bien para comunicación síncrona con dispositivos tales como convertidores CAD O CDA, circuitos integrados o memorias EEPROM con comunicación serie, etc.

La USART del PIC puede ser configurada en tres modos:

1. Modo asíncrono (full duplex (transmisión y recepción simultáneas)).
2. Modo Síncrono-Maestro (half duplex).
3. Modo Síncrono-Esclavo (half duplex).

El modo de transmisión utilizado en este proyecto como se menciono es el modo asíncrono, el cual se explica a continuación:

En este modo la USART usa un formato estándar NRZ asíncrono, el cual para la sincronización usa: 1 bit de inicio (I), 8 o 9 de datos y 1 bit de paro (P). Mientras no se están trasmitiendo datos la USART envía continuamente un bit de marca. El modo asíncrono se selecciona limpiando el bit SYNC del registro TXSTA (98H). El modo asíncrono es deshabilitado durante el modo SLEEP.

Cada dato es transmitido y recibido comenzando por el LSB. El hardware no maneja bit de paridad, pero el noveno bit puede ser usado para este fin y manejando por software como se ve en la Figura 4.9.

De manera mas explícita se explica en las siguientes reglas:

- Cuando no se envían datos la línea debe mantenerse en estado 1.
- Cuando se va a mandar un carácter se envía primero un bit de inicio de valor 0.

- A continuación se envían todos los bits del carácter a transmitir al ritmo marcado por el reloj de transmisión.
- Después del último bit del carácter se envía un bit de paro del valor 1.

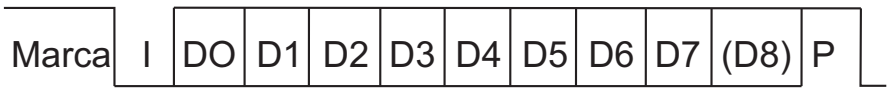


Figura 4.9: Modo asíncrono de transmisión.

El módulo asíncrono de la USART consta de 3 módulos fundamentales:

1. El circuito de muestreo.
2. El generador de frecuencia de transmisión (Baud Rate).
3. El transmisor-receptor asíncrono.

El PIC posee internamente un puerto serial por lo que comunicarse con la PC solo es necesario convertir los voltajes de 5 V DC del PIC a + - 12 V DC como marca el protocolo RS-232 y que son los voltajes que tienen el puerto serie de una PC de escritorio. Esta conservación de voltajes se realiza por medio de un circuito integrado especializado como el MAX232 el se explica en la siguiente sección.

EL MAX232

Este circuito integrado soluciona los problemas de niveles de voltaje cuando se requiere enviar señales digitales sobre una línea RS-232. El MAX232 se usa en aquellas aplicaciones donde no se dispone de fuentes dobles de $+-12V$. El MAX232 necesita solamente una fuente de +5 v para su operación; un elevador de voltaje interno convierte el voltaje de $+5V$ al de doble polaridad de$+-12V$.

Como la mayoría de las aplicaciones de RS-232 necesitan de un receptor y un emisor, el MAX232 incluye en un solo empaque 2 parejas completas de driver y receiver, como lo ilustra la estructura interna del integrado que se muestra en la figura . El MAX232 tiene un doblador de voltaje de +5V a +10 volts y un inversor de voltaje para obtener la polaridad de -10V. El primer convertidor utiliza el condensador C1 para doblar los +5V de entrada a +10V sobre el condensador C3 en la salida positiva V+. El segundo convertidor usa el condensador C2 para invertir +10V a -10V en el condensador C4 de la salida V-. El valor mínimo de estos condensadores los sugiere el fabricante en la siguiente tabla , aunque en la práctica casi siempre se utilizan condensadores de tantalio

de 10μF.(Cf. [Edison Duque, 1998]).

Debido a que nuestro sistema de adquisición utiliza una fuente bipolar, **el circuito MAX2323 se utiliza como una interfaz para evitar un daño eléctrico tanto en el PIC como en la PC.**

Una aplicación clásica consiste en conectar las salidas para transmisión serial TX y RX de un microcontrolador a una interface RS-232 con el fin de intercambiar información con una computadora. La mayoría de los sistemas concentradores de datos están compuestos por sensores conectados a microcontroladores que, a su vez, vía RS-232 le comunican los datos recolectados a una PC. El MAX232 implementa la interfaz con la misma fuente de alimentación de +5V. En la Figura 4.10 se ilustra la conexión serial de un microcontrolador a través del MAX232.

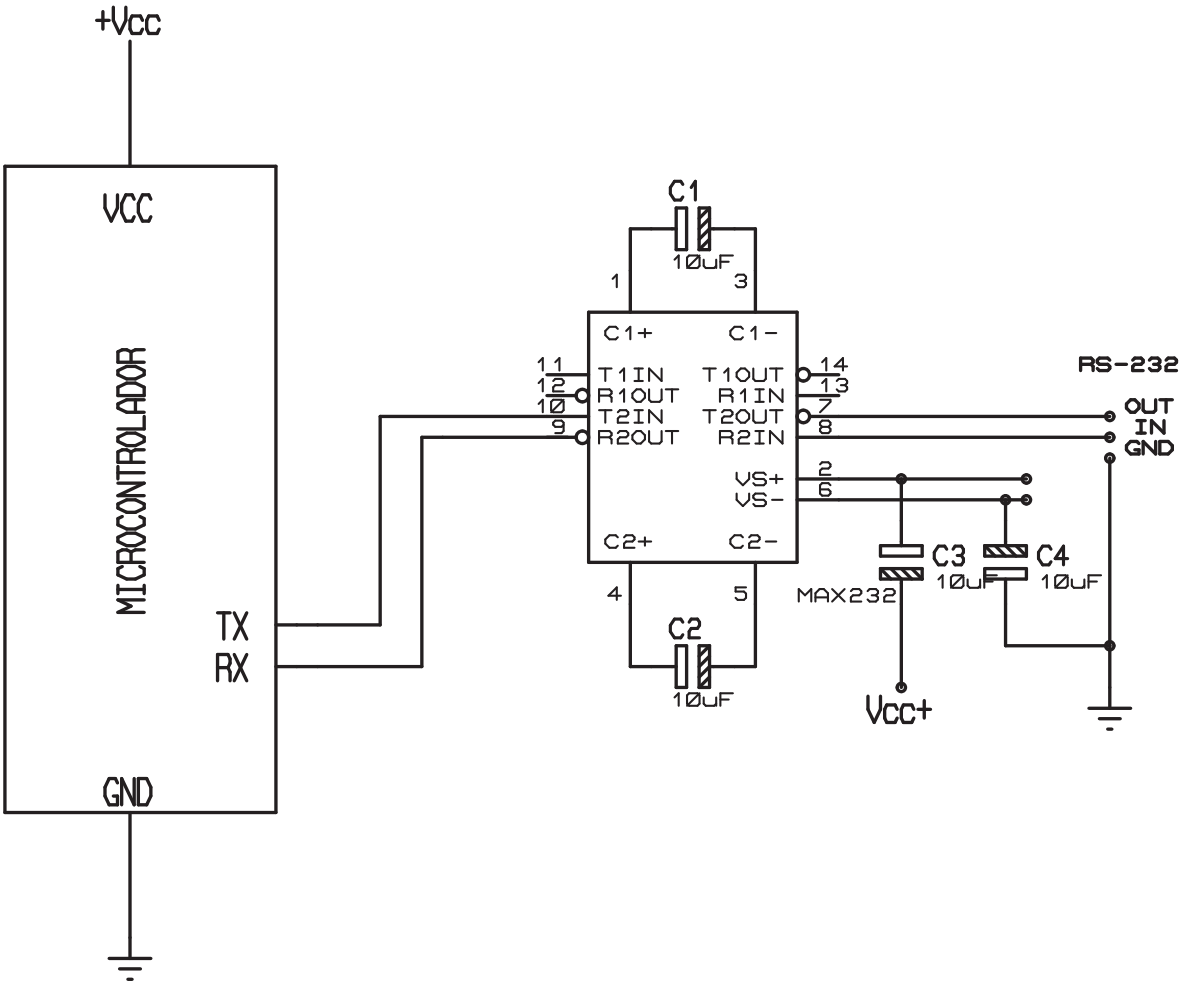


Figura 4.10: Aplicación típica del MAX-232.

4.4.4. Fuente de potencia.

Este módulo corresponde a la fuente de voltaje necesaria para el funcionamiento del microcontrolador PIC16F877 y para los circuitos que se utilizan en la etapa de sensores, acondicionamiento y comunicación.

Se necesita un voltaje V_{cc} de 5 V DC para el PIC16F877 a una corriente de 250 mA. El sensor que detecta el movimiento usa esta misma alimentación, al igual que la interfaz de comunicación (MAX232), ver especificaciones en el **Apéndice E**.

El módulo acondicionador usan amplificadores operacionales (TL 084, TL081) los cuales necesitan voltajes de +− 10 V DC.

Entonces se propone que este módulo este formado por una fuente lineal de +5 V DC y otra fuente bipolar de +− 10 V DC (ver la Figura 4.11). Los dispositivos utilizados para la fuente de potencia se muestra en la figura (en el Apéndice C se puede consultar la Hoja de datos de este dispositivo). Este circuito convierte los voltajes evitando daño eléctrico tanto en el PIC como en la PC.

Por lo tanto el módulo de comunicación que corresponde a conectar este circuito integrado al PIC y al PC. Se utiliza un conector DB-9 para conectarse a la PC.

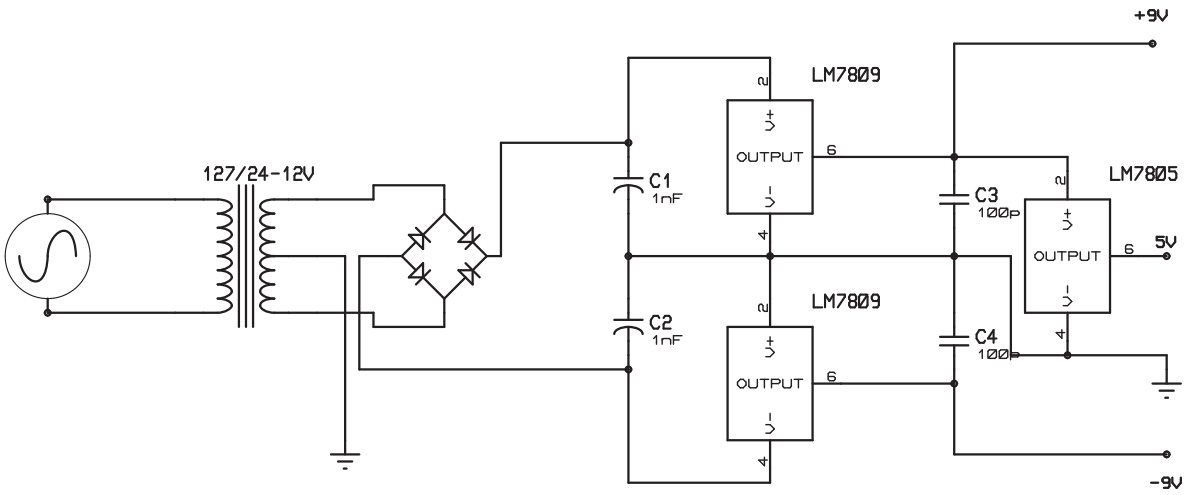


Figura 4.11: Fuente de Potencia.

4.5. Programación del PIC16F877

En este sección se tratará la programación del PIC16F877 y se verá en el siguiente capítulo la programación del ambiente virtual en OpenGL.

Esta etapa es la parte principal de nuestro sistema de adquisición, en donde se programa las siguientes funciones: **El PIC realiza adquisición de señales analógicas, las convierte en digitales y la envía a la PC**. La programación se hará por medio del lenguaje ensamblador que es un lenguaje de bajo nivel.

El desarrollo de este programa es por medio de la programación modular y consiste en desarrollar pequeños programas que realicen una función específica del sistema de adquisición y por último se unen los módulos para formar el programa principal.

Los principales acciones que realiza el PIC son:

1. Configuración de los puertos como entradas y salidas, del CAD y puerto serie a utilizar.
2. Multiplexor o selector de datos donde se aceptan varias entradas de datos y permite una sola de ellas alcanzar la salida.
3. Conversion Analógico a Digital.
4. El transmisor-receptor asíncrono.

Todas estas acciones se encuentran dentro del programa principal cuyo diagrama de flujo se muestra en la Figura 4.12.

4.5.1. Configuración

Como se muestra en diagrama flujo (ver Figura 4.13), lo primero que se realiza en esta acción es configurar los puertos A y E del microcontrolador para que actúen como entradas de las señales a adquirir todos los pines del mismo, luego se configura el puerto C para que el microcontrolador pueda transmitir información, posteriormente se configura el CAD del microcontrolador. Después se configura los registros de transmisión y del generador de Baude Rate (velocidad de transmisión).

4.5.2. Multiplexor

En esta acción comprende una rutina (ver Figura 4.14), la cual por medio del contador del programa (PC) va incrementando realizando la selección de una sentencia posterior, en dicha sentencia se manda a otro salto el cual se en encarga de seleccionar el canal de entrada.

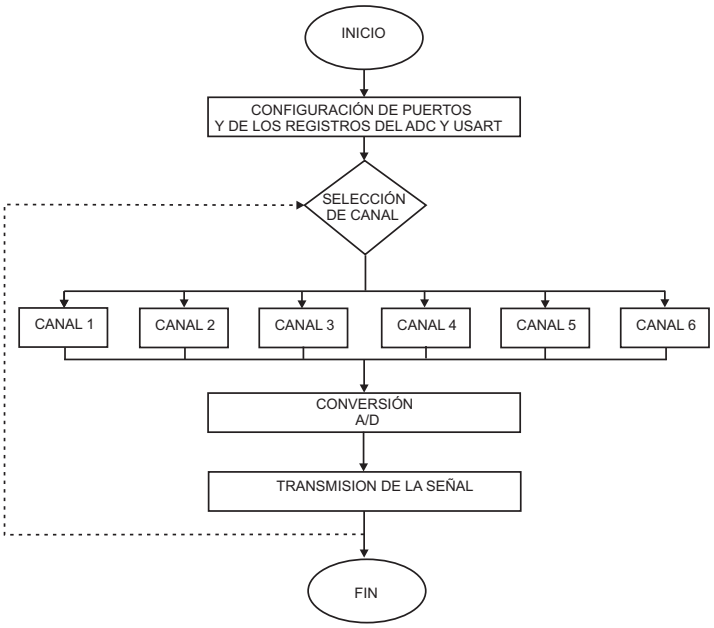


Figura 4.12: Diagrama de flujo del programa principal.

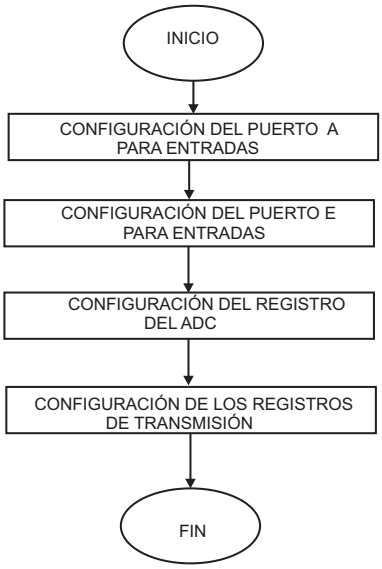


Figura 4.13: Diagrama de flujo para la acción de configuración del PIC16F877.

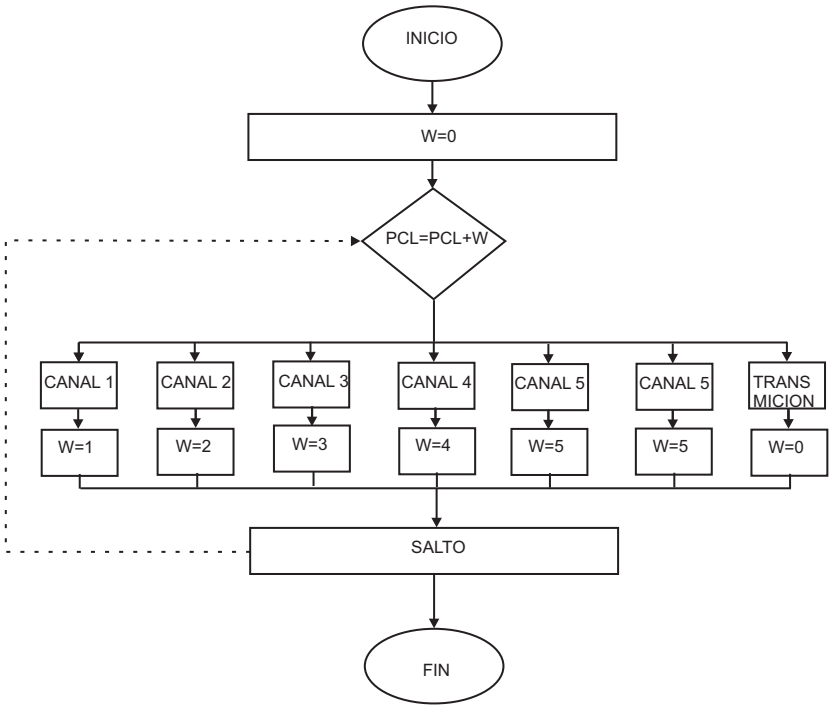


Figura 4.14: Diagrama de flujo para la acción del multiplexor.

4.5.3. Conversión Analógico a Digital

Una vez seleccionado el canal de entrada se realiza la conversión por medio del CAD de la señal analógica correspondiente al canal elegido; posteriormente el dato digitalizado de la parte alta se guarda en la variable OHMS5H y parte baja se guarda en la variable OHMS5L.

4.5.4. El transmisor-receptor asíncrono

Una vez obtenidas las señales digitales de los 6 canales de entrada la siguiente acción es la transmisión (ver Figura 4.15).

En el **Apéndice F** se muestra el programa en lenguaje ensamblador.

4.6. Simulación del PIC16F877 vía PROTEUS

La simulación de sensores, acondicionamiento de señal, adquisición y transmisión se realizo por medio del software **PROTEUS ISIS** Versión 7.

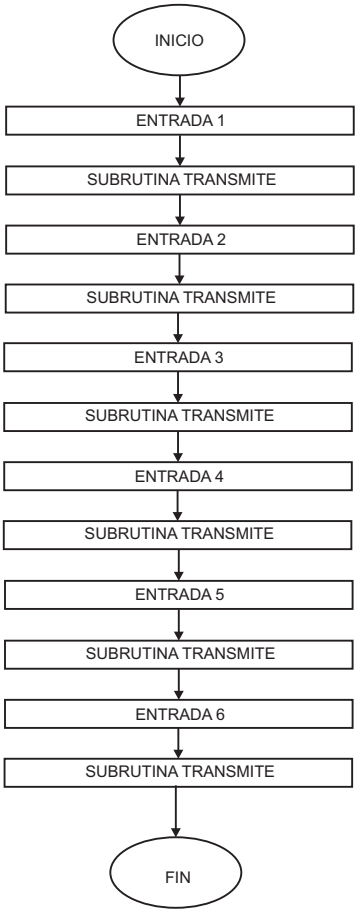


Figura 4.15: Diagrama de flujo para la acción de transmisión.

Proteus es un software de diseño electrónico desarrollado por Labcenter Electronics que consta de dos módulos: Ares e Isis y que incluye un tercer módulo opcional denominado Electra.

- **Isis** Mediante este programa podemos diseñar el circuito que deseemos con componentes muy variados, desde una simple resistencia hasta algún que otro microprocesador o microcontrolador, incluyendo fuentes de alimentación, generadores de señales y muchas otras prestaciones. Los diseños realizados en Isis pueden ser simulados en tiempo real. Una de estas prestaciones es VSM, una extensión de la aplicación con la cual podremos simular, en tiempo real, todas las características de varias familias de microcontroladores, introduciendo nosotros mismos el programa que queramos que lleven a cabo.

- **Ares**es la herramienta de rutado de Proteus, se utiliza para la fabricación de placas de circuito impreso, esta herramienta es manual, es decir que el usuario debe trazar las pistas él mismo, aquí es donde entra el tercer módulo, Electra (Electra Auto Router), el cual, una vez colocados los componentes trazará automaticamente las pistas realizando varias pasadas para optimizar el resultado.

4.6.1. Captura del circuito electrónicos

El proceso de captura del esquema de circuitos electrónicos en ISIS consiste en realizar las siguiente tareas:

- Elegir en las librerías de componentes todos aquellos elementos que se utilizan en el circuito a realizar (ver Figura 4.16 a).

- Situar espacialmente los componentes que forman el circuito en la hoja de trabajo (ver Figura 4.16 b).

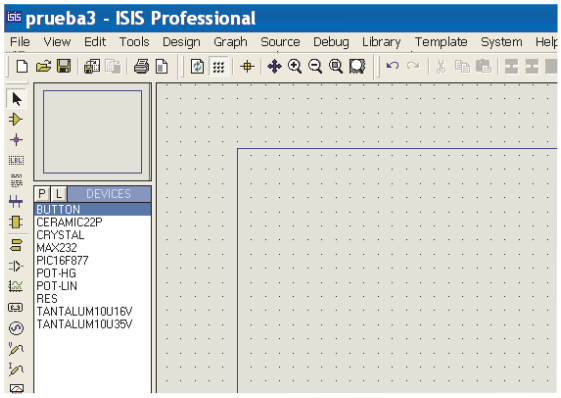
- Conectar las terminales de los componentes entre sí (ver Figura 4.16 c)

- Editar las propiedades de los componentes utilizados: valores nominales encapsulados etc. (ver Figura 4.16 d).

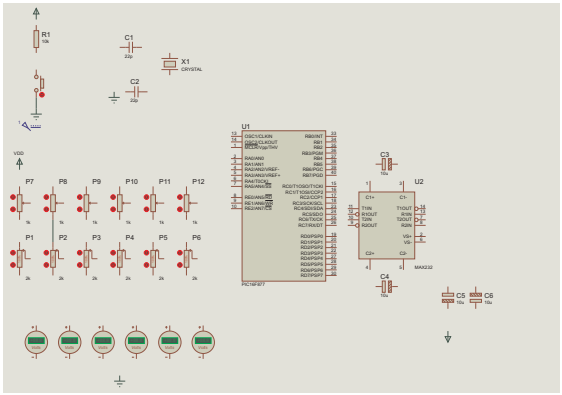
4.6.2. Simulación

Una vez realizado el diseño del hardware e inserción del programa fuente, el siguiente paso es realizar la simulación del funcionamiento del circuito (ver Figura 4.17), esta animación simplemente se realiza pulsando la tecla play.

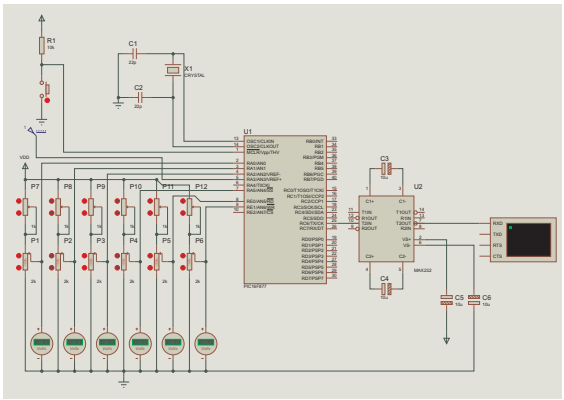
Resultados de la simulación



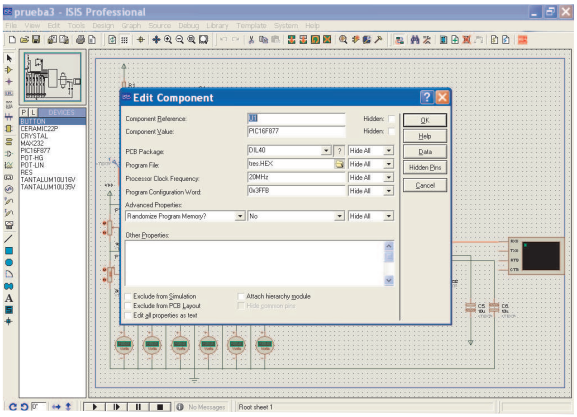
a) Elegir librerias



b) Ubicar componentes



c) Conectar terminales



d) Editar propiedades

Figura 4.16: Captura del circuito electrónico.

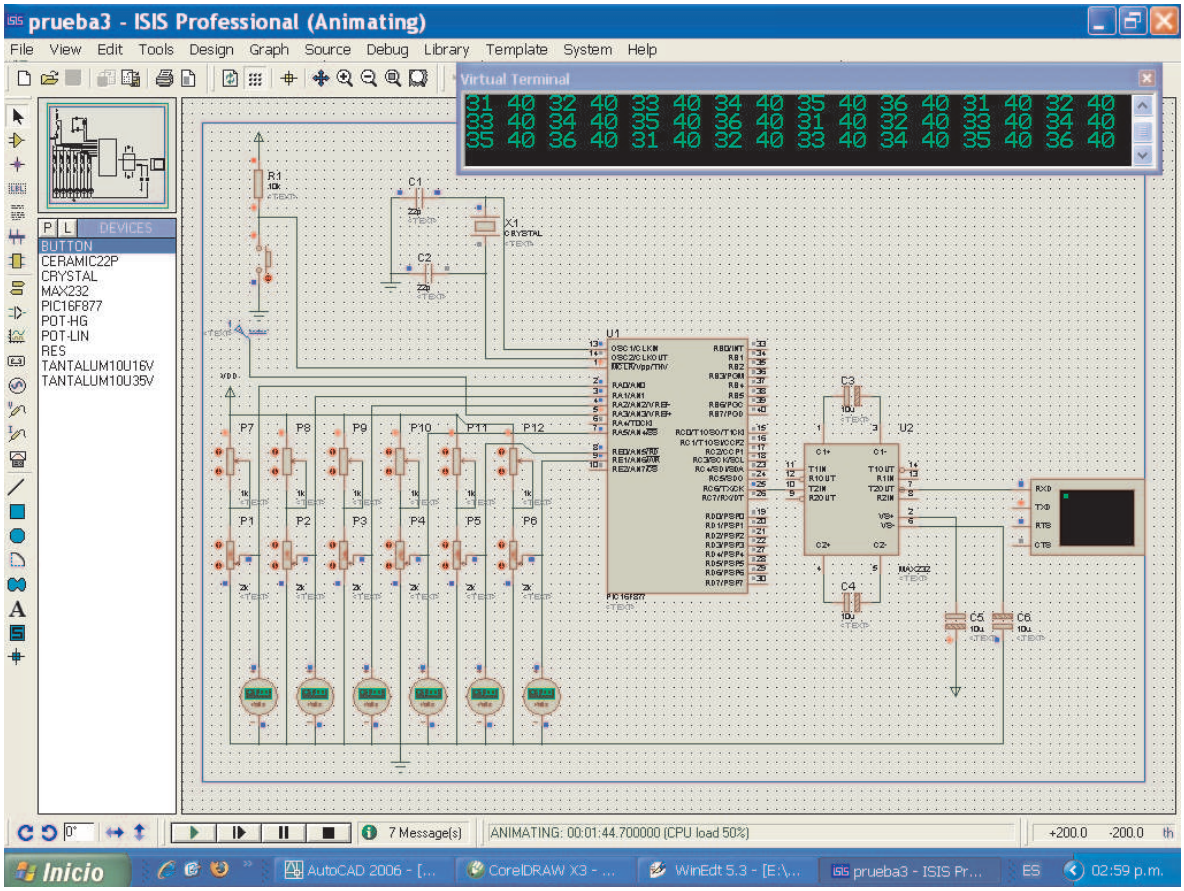


Figura 4.17: Simulación del sistema de adquisición.

Se observo que al girar el potenciómetro de precision el voltaje se incrementa 1 Volt y en la terminal virtual se aprecia que el numero en hexadecimal se incrementa en una unidad. Esta actividad se realizo con cada uno de los potenciómetros, siendo afectado el numero hexadecimal correspondiente. También se verifico cuando se disminuye el voltaje 1Volt ,obteniéndose un decremento en el display de la unidad virtual.

Cabe mencionar que se llevo a cabo otra simulación utilizando el trabajo de tesis (Análisis y diseño de una interfaz de visualización virtual del dispositivo háptico PHANToM 1.0 en la dinámica de Euler-Lagrange) del M. C. Herbert Lara Ordaz. La actividad se realizo modificando el Programa Interfaz virtual PHANToM 1.0, el cual mostraba un robot con tres grados, el cual se le implemento tres grados de libertad más, obteniendo un robot antropomórfico de seis grados de libertad.

Esta simulación permitió mayor facilidad para comparar los datos de las coordenadas cartesianas y los ángulo de las articulaciones con respecto a la posición del robot virtual.

Conclusiones del capítulo

Una parte fundamental del trabajo es la simulación del circuito ya que esto permite verificar el diseño funcionamiento del hardware y el software (programa en ensamblador), por lo tanto el programa de PROTEUS es una herramienta indispensable en el diseño de circuitos.

Capítulo 5

Ambiente virtual

Resumen.

En este capítulo se explica porque se decidió emplear OpenGL para el desarrollo del ambiente virtual, se mencionan las características de OpenGL y Visual C++, se describe como se construyen un escenario con objetos virtuales, como se da animación a los objetos virtuales y como se mueve uno a través del mundo virtual; así también como se integro la cinemática del seguidor y por último como se realizo la interfaz.

5.1. Introducción.

OpenGL (librería de gráficos libre) se centra exclusivamente en servicios para la composición de gráfico 2D/3D.

OpenGL apareció en el año de 1992 como resultado de la colaboración de varias empresas Microsoft, IBM e Intel, entre ellas, cuyo objetivo era ampliar las plataformas en las que podía utilizarse una interfaz de programación de gráficos previa, conocida como IRIS GL, y desarrollada por la empresa Silicon Graphics para sus estaciones de trabajo. OpenGL fue creado como un estándar abierto. Esto significa que cualquiera puede obtener una licencia y efectuar su propia implementación. Dicho estándar es controlado por un comité conocido como ARB , formado por empresas como SGI, NVIDIA, ATI, Intel y Microsoft. Actualmente es una de las tecnologías más empleadas en el diseño de aplicaciones 3D.

Diseñada inicialmente como una solución de nivel profesional para estaciones gráficas de alto rendimiento, hoy día, OpenGL puede utilizarse prácticamente con cualquier tarjeta gráfica más o menos actual, tanto en Windows como en Unix, Linux y Mac OS X. Esta disponibilidad en múltiples plataformas, tanto hardware como sistemas operativos.

5.2. Visual C++ y Ambientes virtuales con OpenGL

OpenGL se define estrictamente como **una interfaz software para gráficos por hardware** (Wrigth y Sweet, 1997). ¿Pero, qué es OpenGL realmente? OpenGL no es un lenguaje de programación; Se puede decir que OpenGL es un API para desarrollo de aplicaciones 3D.

En realidad es más que un simple Interfaz. Son unas definiciones estándar para la definición, gestión y manipulación de gráficos 3D. Cuando se dice que una aplicación esta basada en OpenGL, o que es una aplicación OpenGL, quiere decir que está escrita en un lenguaje de programación que hace llamadas a una o más de las librerías de OpenGL. Fue definido por SGI dirigido a sus máquinas de desarrollo 3D a partir de un lenguaje llamado GL que ya poseían.

5.2.1. Visual C++

OpenGL está principalmente preparado para utilizarse con C. Lenguaje que se ha utilizado en éste proyecto. Pero no por ello es el único lenguaje con el que se puede utilizar, y como muestra, aquí se menciona algunos de ellos.

- JAVA.
- TCL/TK.
- PERL.
- DELPHI.
- VISUAL BASIC.
- C++ Y VISUAL C++.

Para poder utilizar OpenGL con C++ no es necesario nada aparte de las librerías, tal y como se utiliza con C. Viene bien recordar que en Internet se pueden encontrar otras librerías diferentes a las aquí mencionadas, que facilitan la programación. Entre éstas se encuentra la GLUI (GLUT User Interface). GLUI es una librería de Interface en C++ basada en la librería GLUT. Proporciona entre otras cosas controles como buttons, checkboxes, radio buttons, etc ... En el caso de Visual C++, es necesario además de las funciones de OpenGL otras funciones de Windows específicas para la unión entre OpenGL y el GDI.

5.2.2. OpenGL

¿Por qué se determinó emplear OpenGL?

Después de trabajar en VRML, se llegó a la conclusión de que no cubría las necesidades para el ambiente virtual que se estaba proyectando. También se trabajo con Java 3D, pero éste lenguaje presentaba una desventaja, se complicaba demasiado el código para realizar la animación de objetos en un espacio virtual(Cf. [Daniel Selman, 2002]).

Además OpenGL es ampliamente utilizado en la creación de ambientes virtuales. Por lo tanto, se decidió emplear OpenGL.

5.3. Estructura de OpenGL.

OpenGL es una interfaz software para gráficos por hardware es decir, es un medio de enlace entre un ambiente gráfico y el equipo periférico de una computadora; también es definido como una librería de gráficos 3D, de modelación portátil y rápida.

Estas librerías proporcionan una determinada calidad para construir ambientes gráficos en 3D, debido a las características que OpenGL ofrece.

OpenGL se diseñó para ser utilizado en los sistemas operativos: Mac OS, Linux, Unix, Windows y Solaris.

Inicialmente se concibió para programar en máquinas nativas Silicon Graphics bajo el nombre de GL, posteriormente se consideró extenderse a cualquier tipo de plataforma y asegurar así su portabilidad y extensibilidad de uso con lo que se llegó al término Open Graphics Library, es decir, OpenGL (Cf. [Clayton Walnum, 1995]).

5.3.1. Librerías

La API OpenGL está dividida en tres librerías distintas que se mencionan a continuación:

1. OpenGL.DLL (Librería de Gráficos Abierta): podemos encontrarla también con el nombre de OpenGL32.DLL (para el caso de los sistemas operativos de 32 bits) o bien simplemente como GL.DLL, ésta es la librería principal y contiene la mayoría de las funciones que se utilizan en la aplicación; las funciones contenidas en esta librería inician con las letras gl.

2. GLU.DLL (Graphics Utility Library, Librería de Utilerías de Gráficos): también la podemos encontrar como GLU32.DLL, contiene funciones para objetos comunes a dibujar como esferas, donas, cilindros, cubos; estas figuras ya predefinidas pueden llegar a ser útiles en ciertos casos, así como funciones para el manejo de la cámara entre muchas otras; Estas funciones se pueden identificar porque llevan antepuestas en su nombre las siglas glu.

3. GLUT.DLL (GL Utility Toolkit, Juego de Herramientas para uso general de OpenGLŹ).- Esta también permite crear objetos complejos como GLU, aunque la principal función de ésta librería es permitir que los programas se vuelvan interactivos, o sea que posibilita la libre creación de ventanas, así como el acceso al ratón y al teclado.

Existe una librería más llamada GLAUX que no es tan conocida , esta librería contiene algunos procedimientos para crear figuras prediseñadas tales como esferas, cubos, etc. como en las anteriores, la diferencia está en que cuenta con un método que permite cargar bitmaps directamente desde archivos BMP; GLAUX proporciona métodos para crear objetos GLBitmap directamente desde archivos, para ser utilizados en texturas. Actualmente está declarada obsoleta y ya no está ni soportada ni actualizada

Todas las funciones de las librerías OpenGL32.dll y glu32.dll están disponibles cuando se usa la librería AUX para la estructura de trabajo del programa.

5.3.2. Características

A continuación se menciona algunas características de OpenGL:

- **Primitivas geométricas** permiten construir descripciones matemáticas de objetos. Las actuales primitivas son: puntos, líneas, polígonos, imágenes y mapas de bits.
- **Codificación del Color** Existen modos RGBA o de color indexado.
- **Visualización y modelado** permite disponer objetos en una escena tridimensional, es posible mover la cámara por el espacio y seleccionar un posición deseada.
- **Mapeado de texturas** esta característica da mayor realismo a los modelos por medio del dibujo de superficies en las caras de los objetos.
- **La iluminación de materiales** dadas las propiedades del material y las fuentes de luz en la habitación, OpenGL provee de comandos para calcular el color de cualquier punto de acuerdo a la iluminación del objeto.

- **Doble buffering** ayuda a eliminar el defecto intermitente de las animaciones. Cada fotograma consecutivo en una animación se construye en un buffer separado de memoria y es mostrado solo cuando está completo.

- **El Anti-alizado** reduce los bordes escalonados en las líneas dibujadas sobre una pantalla de ordenador. Los bordes escalonados aparecen a menudo cuando las líneas se dibujan con baja resolución. El anti-alizado es una técnica común en gráficos de ordenador que modifica el color y la intensidad de los píxeles cercanos a la línea para reducir el zig-zag artificial.

- **El sombreado Gouraud** es una técnica usada para aplicar sombreados suaves a un objeto 3D y producir una sutil diferencia de color por sus superficies.

- **El Z-buffering** mantiene registros de la coordenada Z de un objeto 3D. El Z-buffer se usa para registrar la proximidad de un objeto al observador, y es también crucial para el eliminado de superficies ocultas.

- **Efectos atmosféricos** proporciona efectos como la niebla, el humo y las neblinas hacen que las imágenes producidas por ordenador sean más realistas.

- **El Alpha blending** El Alpha blending permite simular la transparencia de objetos, de tal manera que objetos colocados detrás del objeto transparente, aparezcan con un tono magenta.

- **Los planos de plantilla** permiten restringir el trazado aciertas regiones de la pantalla.

- **Las listas de Display** permiten almacenar comandos de dibujo en una lista para un trazado posterior.

- **Los Evaluadores Polinómicos** sirven para soportar B-splines racionales no uniformes, esto es para ayudar a dibujar curvas de contorno suave a través de unos cuantos puntos de referencia, evitando la necesidad de acumular grandes cantidades de puntos intermedios.

- **Características de Feedback** ayuda a crear aplicaciones que permiten al usuario seleccionar una región de la pantalla o elegir un objeto dibujado en la misma. El modo de feedback permite al desarrollador obtener los resultados de los cálculos de trazado.

- **Primitivas de Raster (bitmaps y rectángulos de pixels)** es muy útil para composiciones que no precisen grandes calidades de impresión y son ideales para acompañar estudios y documentos.

- **Operaciones con Pixels** una vez que las imágenes han sido digitalizadas y guardadas en forma de matriz en el correspondiente sector de memoria (buffer) de la computadora, resulta inmediata la posibilidad de realizar operaciones aritméticas y lógicas entre ellas.

- **Transformaciones** realiza rotación, escalado y perspectivas en 3D.

- **Perceptiva a la red**, de manera que es posible separar la aplicación OpenGL en un servidor y un cliente que verdaderamente produzca los gráficos. Existe un protocolo para enviar por la red los comandos OpenGL entre el servidor y el cliente. Gracias a su independencia del sistema operativo, el servidor y el cliente no tienen porque ejecutarse en el mismo tipo de plataforma, muy a menudo el servidor será una supercomputadora ejecutando una compleja simulación y el cliente una simple estación de trabajo mayormente dedicada a la visualización gráfica.

5.4. Desarrollo de un ambiente virtual

Para construir un ambiente virtual con OpenGL es necesario tener un espacio de trabajo donde se desarrollará la animación del ambiente virtual en este espacio de trabajo se definen algunas características tales como tamaño de al ventana, posición de la ventana, color del fondo (background), encabezado de la ventana, etc. Después se proporcionan las funciones correspondientes que integran al ambiente virtual, para más detalles del desarrollo del ambiente virtual ver **Apéndice G**.

5.5. Animación del ambiente virtual

Una vez hemos definido el ambiente virtual en coordenadas mundo, se realiza Şla fotoŤ. Para ello, se realizan dos cosas: colocar la cámara en el mundo (o sea, en la escena) y definir el tipo de proyección que realizará la cámara.

Hay que definir no sólo la posición de la cámara (o donde está), sino también hacia dónde mira y con qué orientación (no es lo mismo mirar con la cara torcida que rectaĚ aunque veamos lo mismo). Para hacer esto, basta con modificar la matriz ModelView para mover toda la escena de manera que parezca que hemos movido la cámara.

La librería GLU tiene una función gluLookAt. Su sintaxis es la siguiente:

```
void gluLookAt( eyeX, eyeY, eyeZ, cenX, cenY, cenZ, vp_X, vp_Y,
vp_Z);
```

donde **eye** corresponde a la posición de la cámara, **cen** corresponde al punto hacia donde mira la cámara y **vp** es un vector que define la orientación de la cámara.

Por lo tanto la animación del ambiente virtual se hace con la función **gluLookAt** () que nos proporcionara cambios de posición y orientación.

Se observa que el desplazamiento en eje X se hará con la función:

```
void moverx(float px)
x = px + i*(1)*0.01;
glLoadIdentity();
gluLookAt(x, y, z,
x+lx ,y+ly ,z+lz ,
vx,vy,vz );
```

Se observa que el desplazamiento en eje Y se hará con la función:

```
void movery(float py)
y = y + py*(1)*0.01;
glLoadIdentity();
gluLookAt(x, y, z,
x + lx,y + ly,z + lz,
vx,vy,vz );
```

Se observa que el desplazamiento en eje Z se hará con la función:

```
void moverz(float pz)
y = y + pz*(ly)*0.1;
z = z + k*(lz)*0.1;
glLoadIdentity();
gluLookAt(x, y, z,
x + lx,y + ly,z + lz,
vx,vy,vz );
```

La rotación para el ángulo α la función es:

```
void Yaw(float alfa)
vy = cos(alfa);
vx = sin(alfa);
glLoadIdentity();
gluLookAt(x, y, z,
x + lx,y+ly ,z + lz ,
vx,vy,vz );
```

La rotación para el ángulo β la función es:

```
void Pitch(float beta)
ly = sin(beta);
```

```
lz = -cos(beta);
glLoadIdentity();
gluLookAt(x, y, z,
lx+x , y+ly ,lz+z ,
vx,vy,vz );
```

La rotación para el ángulo γ la función es:

```
void Roll(float gama)
lx = sin(gama);
lz = -cos(gama);
glLoadIdentity();
gluLookAt(x, y, z,
x + lx,y + ly,z + lz ,
vx,vy,vz );
```

Es importante explicar que los ejes están definidos en el espacio de trabajo virtual de la siguiente forma:

- El movimiento sobre la horizontal se hace sobre el eje X.
- El movimiento vertical sobre el eje Y.
- hacia el fondo o acercar sobre el eje Z (ver la Figura 5.1.

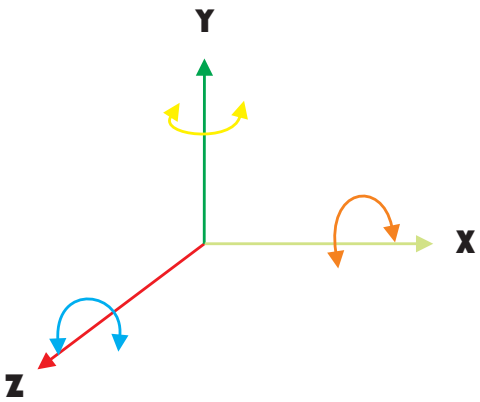


Figura 5.1: Disposición de ejes en el espacio virtual con OpenGL.

Esta definición de eje se manejará de aquí en adelante para el espacio virtual construido con OpenGL, y que es diferente a la definición de ejes establecida por el modelo cinemático.

5.6. Integración y validación del modelo cinemático en el robot virtual

Para integrar el modelo cinemático del ambiente virtual se tiene que de acuerdo a la cadena cinemática, la posición y la orientación definida en capitulo 3. El desarrollo de la simulación se realizo de la siguiente forma:

- Establecer condiciones iniciales (ver la Figura 5.2).
- El movimiento horizontal de **10cm** a la izquierda sobre el eje **X** (ver la Figura 5.3 a).
- El movimiento horizontal de **10cm** a la derecha sobre el eje **X** (ver la Figura 5.3 b).
- El movimiento vertical de **15cm** hacia bajo sobre el eje Y (ver la Figura 5.3 c).
- El movimiento vertical de **6cm** hacia arriba sobre el eje **Y** (ver la Figura 5.3 d).
- El movimiento de **(3cm)** hacia adelante sobre el eje **Z** (ver la Figura 5.3 e).
- El movimiento hacia atrás de **15cm** sobre el eje **Z** (ver la Figura 5.3 f).
- Un giro sobre el eje **Z** de -40° (ver la Figura 5.4 a).
- Un giro sobre el eje **Z** de 40° (ver la Figura 5.4 b).
- Un giro sobre el eje **X** de -15° (ver la Figura 5.4 c).
- Un giro sobre el eje **X** de 15° (ver la Figura 5.4 d).
- Un giro sobre el eje **Y** de -21° (ver la Figura 5.4 e).
- Un giro sobre el eje **Y** de 21° (ver la Figura 5.4 f).

5.7. Comunicación serial RS-232 PC

El puerto serial de las computadoras, conocido también como puerto RS-232, es muy útil ya que permite la comunicación no sólo con otras computadoras, sino también con otros dispositivos tales como el mouse, impresoras y por supuesto, microcontroladores.

Existen dos formas de intercambiar información binaria: la paralela y la serial. La comunicación paralela transmite todos los bits de un dato de manera simultánea y tiene la ventaja que la transferencia es rápida, pero la desventaja de necesitar una gran cantidad de hilos o líneas,situación que encarece los costos y se agrava cuando las distancias

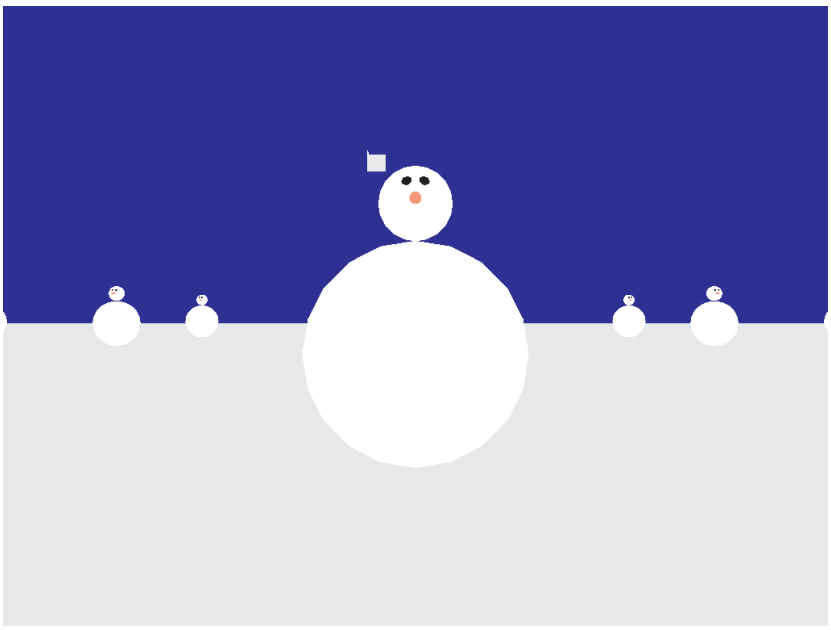


Figura 5.2: Estado Inicial.

que separan los equipos entre los cuales se hace el intercambio es muy grande,debido a las capacitancias entre los conductores, la cual limita el correcto intercambio de datos a unos pocos metros.

En la computadora se requiere un programa que se encargue de configurar el puerto con los valores adecuados (9600, 8, N, 1) y de recibir el dato a la rutina **conectar( )** (**ver Apéndice H**) . En este caso utilizamos un programa en lenguaje Visual C++, debido a que es uno de los más utilizados en aplicaciones electrónicas y permite configurar fácilmente los puertos.

Conclusiones

Se realizo un ambiente virtual en el cual se puede navegar al introducir valores para X,Y,Z (posición) y α β γ (orientación).

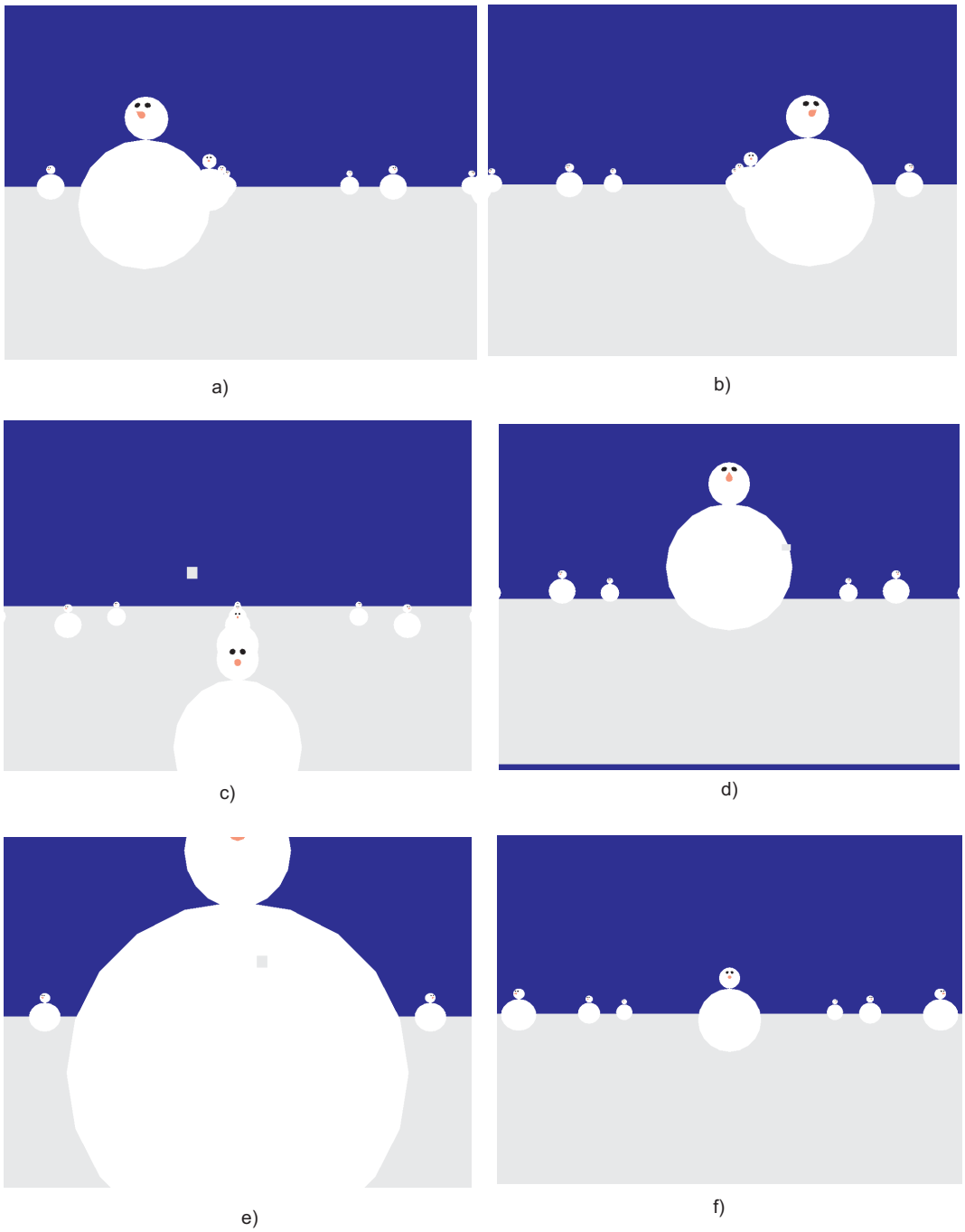


Figura 5.3: Simulación de Posición.

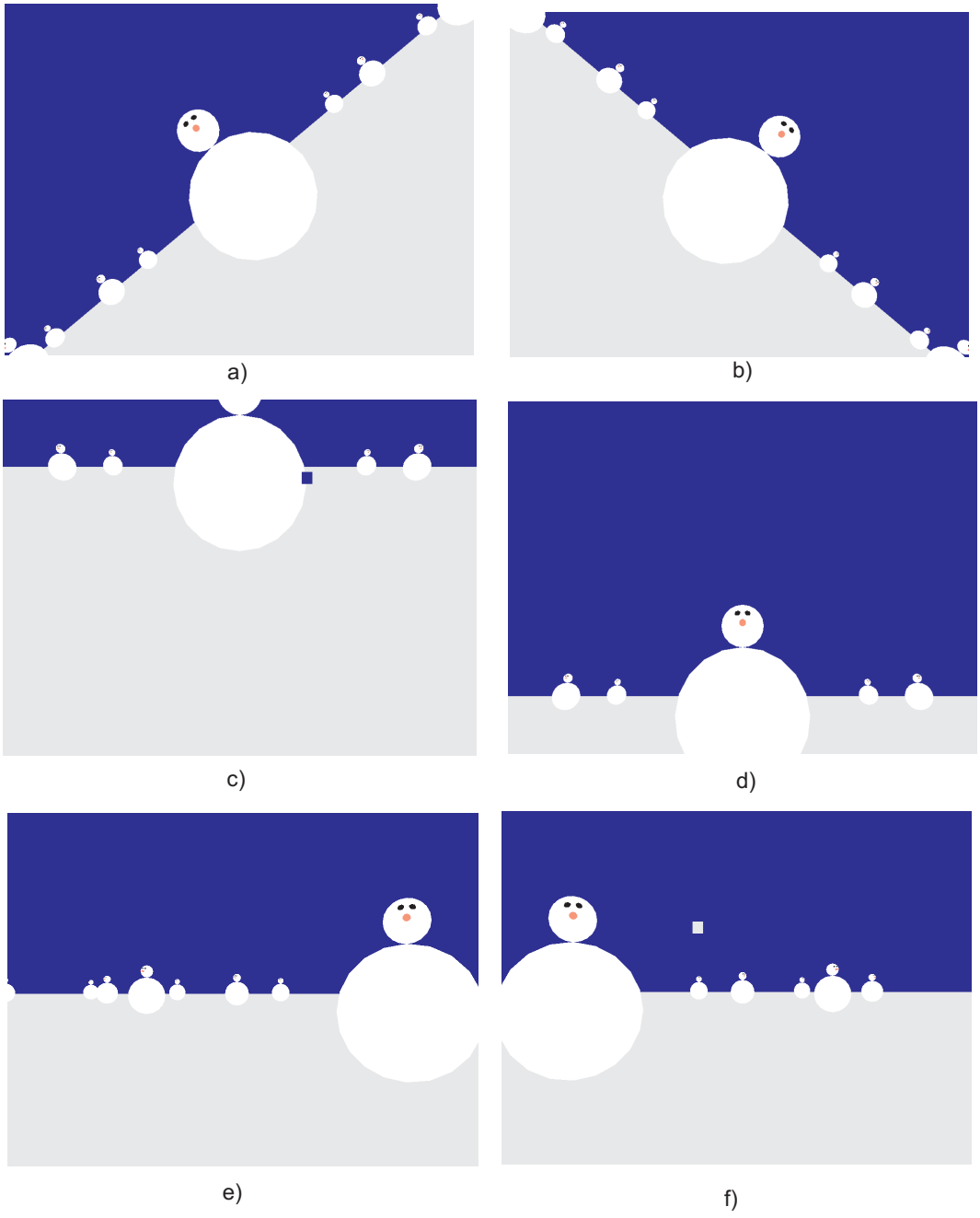


Figura 5.4: Simulación de Orientación.

Conclusiones y perspectivas.

Conclusiones

Se diseño y construyo físicamente un seguidor de posición y orientación con materiales de tubular, pero debido al peso que tiene usuario en la cabeza, se tubo que colocar un contrapeso.

La modelación cinemática se realizó de acuerdo al método Denavit-Hartenberg; este modelo fue simulado en Matlab y con un programa Interfaz virtual PHANToM 1.0.

Al inicio de este proyecto se planeó la construcción, pero debido a tiempo entrega de la tesis, solamente se realizo el simulación de sistema de adquisición y del mundo virtual.

PROTEUS facilita la programación del PIC ya que permite simular dispositivos virtualmente y así verificar programación del PIC16F877.

OpenGL es una buena elección para generación de efectos visuales, escenas, etc, debido a su portabilidad, estandarización y potencia gráfica.

Perspectivas.

Por lo antes menciono se proponen los siguientes trabajo a futuro:

1. Obtener el MCDP considerando las masas (esto debido al contrapeso que tiene en seguidor).
2. Elaboración del sistema mínimo basado en microcontrolador PIC16F877 con el seguidor e interconetarlo con mundo virtual.
3. Enlazar el programa Interfaz virtual PHANToM 1.0 al programa del mundo virtual desarrollado en OpenGL(con finalidad de verificar el MCDP).

Bibliografía

[González, 1995] González, L. M. D. P. (1995). *Realidad Virtual*, paginas 182–184. Paraninfo.

[Robin Hollands, 1996] Robin Hollands. (1996). *The Virtual Reality Homebrewe'rs Handbook*. Jhon Wiley y Sons.

[Anibal Ollero, 2001] Anibal Ollero Bature. (2001). *ROBÓTICA Manipuladores y robots móviles*. Alfaomega.

[Jorge Angeles, 1996] Jorge Angeles. (1996). *Fundamentals of robotic mechanical systems*. Springer.

[Edison Duque, 1998] Edison Duque C. (1998). *Curso avanzado de Microcontroladores PIC*. CEKIT s.a.

[Daniel Selman, 2002] Daniel Selman. (2002). *JAVA 3D Programming*. MANNING.

[Clayton Walnum, 1995] Clayton Walnum. (1995). *3-D Graphics Programming with OpenGL*. QUE.

Glosario de siglas términos técnicos

HMD. - Head Mounted Display.

MIT.- Massachusetts Institute of Technology

VCASS.- Visually-Coupled Airborne System Simulator

BOOM.- Binocular Omni-Orientation Monitor

RMS.- Root Mean Square

CA.- Corriente Alterna.

DC.- Corriente Directa o Corriente Continua.

DOF.- Degrees of Freedom.

MCDP.- Modelo Cinemático Directo de Posición.

RISC.- Reduced Instruction Set Computer.

CMOS.- Complementary metal oxide semiconductor.

PIC.- Program Integrated Circuit.

USART.- Universal Synchronous Asynchronous Receiver Transmitter

.

LSB.- Least Significant Bit

.

CAD.- Convertidor Analógico a Digital.

BCD.- Binary-coded decimal.

OpenGL.- Open Graphics Library.

ARB.- Architectural Review Board.

SGI.- Silicon Graphics Incorporated.

API.- Application Programming Interface.

GL.- Graphics Library.

VRML.- Virtual Reality Modeling Language.

RGBA.- Red Green Blue Alpha.

Apéndice A

Programa en Matlab para realizar la simulación de la posición

```
d1=70;
d2=2.5;
d3=2.5;
d4=2.5;
d5=5;
d6=20;


L2=55;
L3=28;
L4=3.6;
L5=3;
L6=10;


t1=0;
t2=0;
t3=0;
t4=pi/2;
t5=pi/2;
t6=0:-0.01:-pi;
```

$$PX = (((cos(t1)*cos(t2)*cos(t3)-cos(t1)*sin(t2)*sin(t3))*cos(t4)+(-cos(t1)*cos(t2)*sin(t3)-cos(t1)*sin(t2)*cos(t3))*sin(t4))*cos(t5)+sin(t1)*sin(t5))*L6*cos(t6)+(-(cos(t1)*cos(t2)*cos(t3)-cos(t1)*sin(t2)*sin(t3))*sin(t4)-(-cos(t1)*cos(t2)*sin(t3)-cos(t1)*sin(t2)*cos(t3))*cos(t4))*L6*sin(t6)+(((cos(t1)*cos(t2)*cos(t3)-cos(t1)*sin(t2)*sin(t3))*cos(t4)+(-cos(t1)*cos(t2)*sin(t3)-cos(t1)*sin(t2)*cos(t3))*sin(t4))*sin(t5)-sin(t1)*cos(t5))*d6+((cos(t1)*cos(t2)*cos(t3)-cos(t1)*sin(t2)*sin(t3))*cos(t4)+(-cos(t1)*cos(t2)*sin(t3)-cos(t1)*sin(t2)*cos(t3))*sin(t4))*L5*sin(t5)+sin(t1)*L5*cos(t5)+((cos(t1)*cos(t2)*cos(t3)-cos(t1)*sin(t2)*sin(t3))*sin(t4)+(-cos(t1)*cos(t2)*sin(t3)-cos(t1)*sin(t2)*cos(t3))*cos(t4))*d5+(cos(t1)*cos(t2)*cos(t3)-cos(t1)*sin(t2)*sin(t3))*L4*cos(t4)+(-cos(t1)*cos(t2)*sin(t3)-cos(t1)*sin(t2)*cos(t3))*L4*sin(t4)-sin(t1)*d4+cos(t1)*cos(t2)*L3*cos(t3)-cos(t1)*sin(t2)*L3*sin(t3)-sin(t1)*d3+cos(t1)*L2*cos(t2)-sin(t1)*d2;$$

$PY = (((sin(t1)*cos(t2)*cos(t3)-sin(t1)*sin(t2)*sin(t3))*cos(t4)+(-sin(t1)*cos(t2)*sin(t3)-sin(t1)*sin(t2)*cos(t3))*sin(t4))*cos(t5)-cos(t1)*sin(t5))*L6*cos(t6)+(-(sin(t1)*cos(t2)*cos(t3)-sin(t1)*sin(t2)*sin(t3))*sin(t4)-(-sin(t1)*cos(t2)*sin(t3)-sin(t1)*sin(t2)*cos(t3))*cos(t4))*L6*sin(t6)+(((sin(t1)*cos(t2)*cos(t3)-sin(t1)*sin(t2)*sin(t3))*cos(t4)+(-sin(t1)*cos(t2)*sin(t3)-sin(t1)*sin(t2)*cos(t3))*sin(t4))*sin(t5)+cos(t1)*cos(t5))*d6+((sin(t1)*cos(t2)*cos(t3)-sin(t1)*sin(t2)*sin(t3))*cos(t4)+(-sin(t1)*cos(t2)*sin(t3)-sin(t1)*sin(t2)*cos(t3))*sin(t4))*L5*sin(t5)-cos(t1)*L5*cos(t5)+((sin(t1)*cos(t2)*cos(t3)-sin(t1)*sin(t2)*sin(t3))*sin(t4)+(-sin(t1)*cos(t2)*sin(t3)-sin(t1)*sin(t2)*cos(t3))*cos(t4))*d5+(sin(t1)*cos(t2)*cos(t3)-sin(t1)*sin(t2)*sin(t3))*L4*cos(t4)+(-sin(t1)*cos(t2)*sin(t3)-sin(t1)*sin(t2)*cos(t3))*L4*sin(t4)+cos(t1)*d4+sin(t1)*cos(t2)*L3*cos(t3)-sin(t1)*sin(t2)*L3*sin(t3)+cos(t1)*d3+sin(t1)*L2*cos(t2)+cos(t1)*d2;$

$PZ = ((-sin(t2)*cos(t3)-cos(t2)*sin(t3))*cos(t4)+(sin(t2)*sin(t3)-cos(t2)*cos(t3))*sin(t4))*cos(t5)*L6*cos(t6)+(-(-sin(t2)*cos(t3)-cos(t2)*sin(t3))*sin(t4)-(sin(t2)*sin(t3)-cos(t2)*cos(t3))*cos(t4))*L6*sin(t6)+((-sin(t2)*cos(t3)-cos(t2)*sin(t3))*cos(t4)+(sin(t2)*sin(t3)-cos(t2)*cos(t3))*sin(t4))*sin(t5)*d6+((-sin(t2)*cos(t3)-cos(t2)*sin(t3))*cos(t4)+(sin(t2)*sin(t3)-cos(t2)*cos(t3))*sin(t4))*L5*sin(t5)+((-sin(t2)*cos(t3)-cos(t2)*sin(t3))*sin(t4)+(sin(t2)*sin(t3)-cos(t2)*cos(t3))*cos(t4))*d5+(-sin(t2)*cos(t3)-cos(t2)*sin(t3))*L4*cos(t4)+(sin(t2)*sin(t3)-cos(t2)*cos(t3))*L4*sin(t4)-sin(t2)*L3*cos(t3)-cos(t2)*L3*sin(t3)-L2*sin(t2)+d1;$

subplot(3,3,1); hndl=plot(t6,PX); set(hndl,'Color','blue'); set(hndl,'LineWidth',1.5); set(gca,'XGrid',ón','YGrid',ón'); xlabel('\ \theta_6','FontSize',14); ylabel('X','FontSize',16);

subplot(3,3,2); hndl=plot(t6,PY); set(hndl,'Color','red'); set(hndl,'LineWidth',1.5); set(gca,'XGrid',ón','YGrid',ón'); xlabel('\ \theta_6','FontSize',14); ylabel('Y','FontSize',16); title('Giro\ \theta_6','FontSize',18);

subplot(3,3,3); hndl=plot(t6,PZ); set(hndl,'Color','green'); set(hndl,'LineWidth',1.5); set(gca,'XGrid',ón','YGrid',ón'); xlabel('\theta_6','FontSize',14); ylabel('Z','FontSize',16);

Apéndice B

Programa en Matlab para realizar la simulación de la orientación

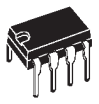
```
d1=70;
d2=2.5;
d3=2.5;
d4=2.5;
d5=5;
d6=20;

L2=55;
L3=28;
L4=3.6;
L5=3;
L6=10;

t4=pi/2;
t5=pi/2;
t6=0.0:-0.01:-pi;
t32 =sin(t5)*sin(t6);
t31 =-sin(t5)*cos(t6);
t33 =cos(t5);
t23 =sin(t4)*sin(t5);
t13 =cos(t4)*sin(t5);

alfa= atan2(t32,-t31);
beta = atan2(sqrt((t31).^2 + (t32),2), t33);
gama= atan2(t23, t13);

subplot(3,3,1); hndl = plot(t6, gama); set(hndl,'Color','green'); set(hndl,'LineWidth',1,5);
set(gca,'XGrid','on','YGrid','on'); xlabel('\theta_6','FontSize',14); ylabel('Y_rad','FontSize',1

subplot(3,3,2); hndl = plot(t6, beta); set(hndl,'Color','red'); set(hndl,'LineWidth',1,5);
set(gca,'XGrid','on','YGrid','on'); xlabel('\theta_6','FontSize',14); ylabel('X_rad','FontSize',1
title('Giro \theta_6 sobre el ejeZ','FontSize',18);

subplot(3,3,3); hndl = plot(t6, alfa); set(hndl,'Color','blue'); set(hndl,'LineWidth',1,5);
set(gca,'XGrid','on','YGrid','on'); xlabel('\theta_6','FontSize',14); ylabel('Z_rad','FontSize',1
```

Apéndice C

TL081
TL081A - TL081B

GENERAL PURPOSE J-FET SINGLE OPERATIONAL AMPLIFIER

- WIDE COMMON-MODE (UP TO V_{CC}^+) AND DIFFERENTIAL VOLTAGE RANGE
- LOW INPUT BIAS AND OFFSET CURRENT
- OUTPUT SHORT-CIRCUIT PROTECTION
- HIGH INPUT IMPEDANCE J–FET INPUT STAGE
- INTERNAL FREQUENCY COMPENSATION
- LATCH UP FREE OPERATION
- HIGH SLEW RATE : 16V/ µs (typ)



N
DIP8
(Plastic Package)

D
SO8
(Plastic Micropackage)

DESCRIPTION

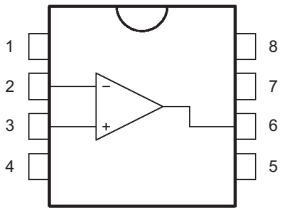
The TL081, TL081A and TL081B are high speed J–FET input single operational amplifiers incorporating well matched, high voltage J–FET and bipolar transistors in a monolithic integrated circuit.

The devices feature high slew rates, low input bias and offset currents, and low offset voltage temperature coefficient.

ORDER CODES

Part Number	Temperature Range	Package	
		N	D
TL081M/AM/BM	–55°C, +125°C	ω	ω
TL081I/AI/BI	–40°C, +105°C	ω	ω
TL081C/AC/BC	0°C, +70°C	ω	ω
Examples : TL081CD, TL081IN			

PIN CONNECTIONS (top view)

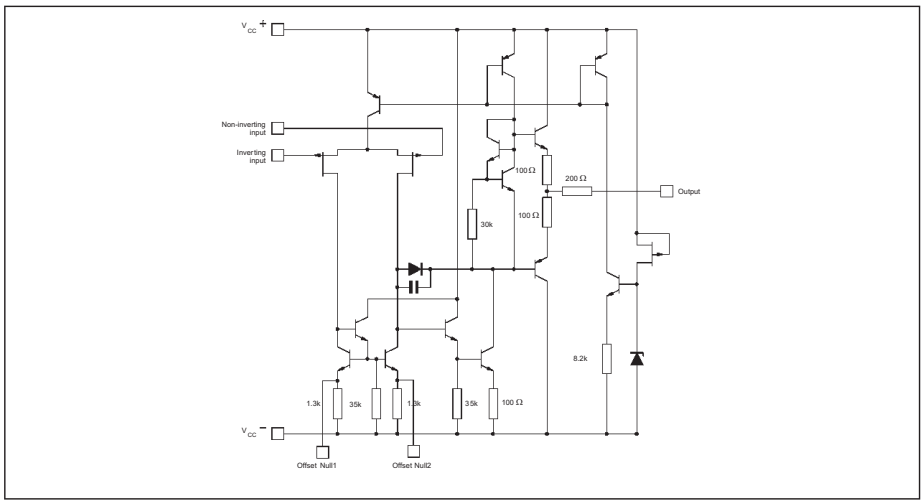


1 - Offset Null 1
2 - Inverting input
3 - Non-inverting input
4 - V_{CC}^-
5 - Offset Null 2
6 - Output
7 - V_{CC}^+
8 - N.C.

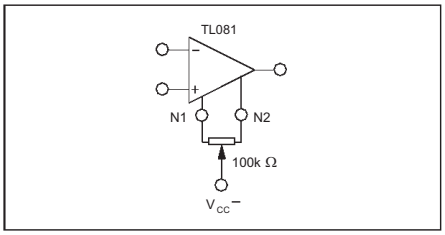
December 1998

TL081 - TL081A - TL081B

SCHEMATIC DIAGRAM



INPUT OFFSET VOLTAGE NULL CIRCUITS



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter		Value	Unit
V_{CC}	Supply Voltage - (note 1)		+18	V
V_i	Input Voltage - (note 3)		+15	V
V_{id}	Differential Input Voltage - (note 2)		+30	V
P_{tot}	Power Dissipation		680	mW
	Output Short-circuit Duration - (note 4)		Infinite	
T_{oper}	Operating Free Air Temperature Range	TL081C,AC,BC TL081I,AI,BI TL081M,AM,BM	0 to 70 –40 to 105 –55 to 125	°C
T_{stg}	Storage Temperature Range		–65 to 150	°C

Notes : 1. All voltage values, except differential voltage, are with respect to the zero reference level (ground) of the supply voltages where the zero reference level is the midpoint between V_{CC}^+ and V_{CC}^-.
2. Differential voltages are at the non-inverting input terminal with respect to the inverting input terminal.
3. The magnitude of the input voltage must never exceed the magnitude of the supply voltage or 15 volts, whichever is less.
4. The output may be shorted to ground or to either supply. Temperature and /or supply voltages must be limited to ensure that the dissipation rating is not exceeded.

ELECTRICAL CHARACTERISTICS

V_{CC} =+ 15V, T_{amb} = 25°C (unless otherwise specified)

Symbol	Parameter		TL081I,M,AC,AI, AM,BC,BI,BM			TL081C			Unit
			Min.	Typ.	Max.	Min.	Typ.	Max.	
V_{io}	Input Offset Voltage (R_S = 50Ω) T_{amb} = 25°C $T_{min.}$ ≤ T_{amb} ≤ $T_{max.}$	TL081 TL081A TL081B TL081 TL081A TL081B		3 3 1	10 6 3 13 7 5		3	10 13	mV
DV_{io}	Input Offset Voltage Drift			10			10		µV/°C
I_{io}	Input Offset Current * T_{amb} = 25°C $T_{min.}$ ≤ T_{amb} ≤ $T_{max.}$			5	100 4		5	100 4	pA nA
I_{ib}	Input Bias Current * T_{amb} = 25°C $T_{min.}$ ≤ T_{amb} ≤ $T_{max.}$			20	200 20		20	400 20	pA nA
A_{vd}	Large Signal Voltage Gain (R_L = 2kΩ, V_O = +10V) T_{amb} = 25°C $T_{min.}$ ≤ T_{amb} ≤ $T_{max.}$		50 25	200		25 15	200		V/mV
SVR	Supply Voltage Rejection Ratio (R_S = 50Ω) T_{amb} = 25°C $T_{min.}$ ≤ T_{amb} ≤ $T_{max.}$		80 80	86		70 70	86		dB
I_{CC}	Supply Current, no Load T_{amb} = 25°C $T_{min.}$ ≤ T_{amb} ≤ $T_{max.}$			1.4	2.5 2.5		1.4	2.5 2.5	mA
V_{icm}	Input Common Mode Voltage Range		+11	+15 -12		+11	+15 -12		V
CMR	Common Mode Rejection Ratio (R_S = 50Ω) T_{amb} = 25°C $T_{min.}$ ≤ T_{amb} ≤ $T_{max.}$		80 80	86		70 70	86		dB
Ios	Output Short-circuit Current T_{amb} = 25°C $T_{min.}$ ≤ T_{amb} ≤ $T_{max.}$		10 10	40	60 60	10 10	40	60 60	mA
$+V_{OPP}$	Output Voltage Swing T_{amb} = 25°C $T_{min.}$ ≤ T_{amb} ≤ $T_{max.}$ 	R_L = 2kΩ R_L = 10kΩ R_L = 2kΩ R_L = 10kΩ	10 12 10 12	12 13.5		10 12 10 12	12 13.5		V
SR	Slew Rate (V_{in} = 10V, R_L = 2kΩ,C_L = 100pF, T_{amb} = 25°C, unity gain)		8	16		8	16		V/µs
t_r	Rise Time (V_{in} = 20mV, R_L = 2kΩ, C_L = 100pF, T_{amb} = 25°C, unity gain)			0.1			0.1		µs
K_{OV}	Overshoot (V_{in} = 20mV, R_L = 2kΩ, C_L = 100pF, T_{amb} = 25°C, unity gain)			10			10		%
GBP	Gain Bandwidth Product (f = 100kHz, T_{amb} = 25°C, V_{in} = 10mV, R_L = 2kΩ, C_L = 100pF)		2.5	4		2.5	4		MHz
R_i	Input Resistance			10^{12}			10^{12}		Ω
THD	Total Harmonic Distortion (f = 1kHz, A_V = 20dB, R_L = 2kΩ, C_L = 100pF, T_{amb} = 25°C, V_O = $2V_{PP}$)			0.01			0.01		%
e_n	Equivalent Input Noise Voltage (f = 1kHz, R_s = 100Ω)			15			15		$\frac{nV}{\sqrt{Hz}}$
φm	Phase Margin			45			45		Degrees

* The input bias currents are junction leakage currents which approximately double for every 10°C increase in the junction temperature.

Apéndice D

2.2.- Características generales del PIC16F877

La siguiente es una lista de las características que comparte el PIC16F877 con los dispositivos más cercanos de su familia:

PIC16F873	PIC16F874	PIC16F876	PIC16F877

CPU:
- Tecnología RISC
- Sólo 35 instrucciones que aprender
- Todas las instrucciones se ejecutan en un ciclo de reloj, excepto los saltos que requieren dos
- Frecuencia de operación de 0 a 20 MHz (200 nseg de ciclo de instrucción)
- Opciones de selección del oscilador

Memoria:
- Hasta 8k x 14 bits de memoria Flash de programa
- Hasta 368 bytes de memoria de datos (RAM)
- Hasta 256 bytes de memoria de datos EEPROM

- Lectura/escritura de la CPU a la memoria flash de programa
- Protección programable de código
- Stack de hardware de 8 niveles

Reset e interrupciones:
- Hasta 14 fuentes de interrupción
- Reset de encendido (POR)
- Timer de encendido (PWRT)
- Timer de arranque del oscilador (OST)
- Sistema de vigilancia Watchdog timer.

Otros:
- Modo SLEEP de bajo consumo de energía
- Programación y depuración serie “In-Circuit” (ICSP) a través de dos patitas
- Rango de voltaje de operación de 2.0 a 5.5 volts
- Alta disipación de corriente de la fuente: 25mA
- Rangos de temperatura: Comercial, Industrial y Extendido
- Bajo consumo de potencia:
  - Menos de 0.6mA a 3V, 4 Mhz
  - 20 µA a 3V, 32 Khz
  - menos de 1µA corriente de standby (modo SLEEP).

Periféricos:

Periférico	PIC16F873 PIC16F876	PIC16F874 PIC16F877	Características
3 a 5 Puertos paralelos	PortA,B,C	PortA, B,C,D,E	con líneas digitales programables individualmente
3 Timers	Timer0	Timer0	Contador/Temporizador de 8 bits con pre-escalador de 8 bits
	Timer1	Timer1	Contador/Temporizador de 16 bits con pre-escalador
	Timer2	Timer2	Contador/Temporizador de 8 bits con pre-escalador y post-escalador de 8 bits y registro de periodo
2 módulos CCP	Captura	Captura	16 bits, 1.5 nseg de resolución máxima
	Comparación	Comparación	16 bits, 200 nseg de resolución máxima
	PWM	PWM	10 bits
1 Convertidor A/D	AN0,...,AN4	AN0,...,AN7	de 10 bits, hasta 8 canales
Puertos Serie	SSP	SSP	Puerto Serie Síncrono
	USART/SCI	USART/SCI	Puerto Serie Universal
	ICSP	ICSP	Puerto serie para programación y depuración “in circuit”
Puerto Paralelo Esclavo	PSP	PSP	Puerto de 8 bits con líneas de protocolo

2.5.- Conjunto de Instrucciones de Rango Medio

En la siguiente tabla se resumen las 35 instrucciones que reconoce la CPU de los PIC de medio rango, incluyendo su mnemónico, tiempo de ejecución, código de máquina y afectación de banderas:

Mnemónico	Descripción	Ciclos	Código de Máquina	Banderas afectadas
Operaciones con el archivo de registros orientadas a bytes				
ADDWF f,d	Suma f + W	1	00 0111 dfff ffff	C,DC,Z
ANDWF f,d	W AND f	1	00 0101 dfff ffff	Z
CLRF f	Limpia f	1	00 0001 1fff ffff	Z
CLRW	Limpia W	1	00 0001 0xxx xxxx	Z
COMF f,d	Complementa los bits de f	1	00 1001 dfff ffff	Z
DECF f,d	Decrementa f en 1	1	00 0011 dfff ffff	Z
DECFSZ f,d	Decrementa f, escapa si 0	1(2)	00 1011 dfff ffff	
INCF f,d	Incrementa f en 1	1	00 1010 dfff ffff	Z
INCFSZ f,d	Incrementa f, escapa si 0	1(2)	00 1111 dfff ffff	
IORWF f,d	W OR f	1	00 0100 dfff ffff	Z
MOVF f,d	Copia el contenido de f	1	00 1000 dfff ffff	Z
MOVWF f	Copia contenido de W en f	1	00 0000 1fff ffff	
NOP	No operación	1	00 0000 0xx0 0000	
RLF f,d	Rota f a la izquierda	1	00 1101 dfff ffff	C
RRF f,d	Rota f a la derecha	1	00 1100 dfff ffff	C
SUBWF f,d	Resta f – W	1	00 0010 dfff ffff	C,DC,Z
SWAPF f,d	Intercambia nibbles de f	1	00 1110 dfff ffff	
XORWF f,d	W EXOR f	1	00 0110 dfff ffff	Z
Operaciones con el archivo de registros orientadas a bits				
BCF f,b	Limpia bit b en f	1	01 00bb bfff ffff	
BSF f,b	Pone bit b en f	1	01 01bb bfff ffff	
BTFSC f,b	Prueba bit b en f, escapa si 0	1(2)	01 10bb bfff ffff	
BTFSS f,b	Prueba bit b en f, escapa si 1	1(2)	01 11bb bfff ffff	
Operaciones con literales y de control del programa				
ADDLW k	Suma literal k + W → W	1	11 111x kkkk kkkk	C,DC,Z
ANDLW k	k AND W → W	1	11 1001 kkkk kkkk	Z
CALL k	Llamado a subrutina	2	10 0kkk kkkk kkkk	
CLRWDT	Limpia timer del watchdog	1	00 0000 0110 0100	$\overline{TO}$, $\overline{PD}$
GOTO k	Salto a la dirección k	2	10 1kkk kkkk kkkk	
IORLW k	k OR W → W	1	11 0000 kkkk kkkk	Z
MOVLW k	Copia literal a W	1	11 00xx kkkk kkkk	
RETFIE	Retorna de interrupción	2	00 0000 0000 1001	
RETLW k	Retorna con literal k en W	2	11 01xx kkkk kkkk	
RETURN	Retorna de subrutina	2	00 0000 0000 1000	
SLEEP	Activa Modo standby	1	00 0000 0110 0011	$\overline{TO}$, $\overline{PD}$
SUBLW k	Resta k – W → W	1	11 110x kkkk kkkk	C,CD,Z
XORLW k	k EXOR W → W	1	11 1010 kkkk kkkk	Z

Notación: **d**= destino del resultado ® d = 0 destino W; d = 1 destino registro

f =dirección del registro (memoria RAM), **b**= número de bit (0 a 7), **k**= dato de 8 bits

El Archivo de Registros

Aunque el archivo de registros en RAM puede variar de un PIC a otro, la familia del PIC16F87x coincide casi en su totalidad. En la siguiente figura se muestra a detalle el mapa de este archivo de registros y su organización en los cuatro bancos que ya se describieron.

Dirección	registro	Dirección	registro	Dirección	registro	Dirección	registro
00h	INDF(*)	80h	INDF(*)	100h	INDF(*)	180h	INDF(*)
01h	TMR0	81h	OPTION_REG	101h	TMR0	181h	OPTION_REG
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PORTA	85h	TRISA	105h		185h	
06h	PORTB	86h	TRISB	106h	PORTB	186h	TRISB
07h	PORTC	87h	TRISC	107h		187h	
08h	PORTD(1)	88h	TRISD(1)	108h		188h	
09h	PORTE(1)	89h	TRISE(1)	109h		189h	
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh	INTCON
0Ch	PIR1	8Ch	PIE1	10Ch	EEDATA	18Ch	EECON1
0Dh	PIR2	8Dh	PIE2	10Dh	EEADR	18Dh	EECON2
0Eh	TMR1L	8Eh	PCON	10Eh	EEDATH	18Eh	RESERVADO(2)
0Fh	TMR1H	8Fh		10Fh	EEADRH	18Fh	RESERVADO(2)
10h	T1CON	90h		110h		190h	
11h	TMR2	91h	SSPCON2	111h		191h	
12h	T2CON	92h	PR2	112h		192h	
13h	SSPBUF	93h	SSPADD	113h		193h	
14h	SSPCON	94h	SSPSTAT	114h		194h	
15h	CCPR1L	95h		115h		195h	
16h	CCPR1H	96h		116h	Registros de propósito general (16 bytes)	196h	Registros de propósito general (16 bytes)
17h	CCP1CON	97h		117h		197h	
18h	RCSTA	98h	TXSTA	118h		198h	
19h	TXREG	99h	SPBRG	119h		199h	
1Ah	RCREG	9Ah		11Ah		19Ah	
1Bh	CCPR2L	9Bh		11Bh		19Bh	
1Ch	CCPR2H	9Ch		11Ch		19Ch	
1Dh	CCP2CON	9Dh		11Dh		19Dh	
1Eh	ADRESH	9Eh	ADRESL	11Eh		19Eh	
1Fh	ADCON0	9Fh	ADCON1	11Fh		19Fh	
20h	Registros de propósito General (96 bytes)	A0h	Registros de propósito General (80 bytes)	120h	Registros de propósito General (80 bytes)	1A0h	Registros de propósito General (80 bytes)
7Fh	**Banco 0**	EFh		16Fh		1EFh	
		F0h	Acceso a regs 70h–7Fh	170h	Acceso a regs 70h–7Fh	1F0h	Acceso a regs 70h–7Fh
		FFh	**Banco 1**	17Fh	**Banco 2**	1FFh	**Banco 3**

Notas: (1): Estos registros no están implementados en el PIC16F876
(2): Estos registros están reservados, manténgalos limpios
(*) no es un registro físico

Localidades de memoria de datos no implementadas, se leen como '0'

Cada uno de los registros de propósito especial, está asociado a un dispositivo interno del µcc. En el siguiente capítulo se tratará con detalle el uso de cada uno de estos dispositivos y de los registros asociados a él.

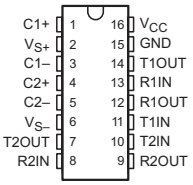
Apéndice E

MAX232, MAX232I
DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047I – FEBRUARY 1989 – REVISED OCTOBER 2002

- **Meet or Exceed TIA/EIA-232-F and ITU Recommendation V.28**
- **Operate With Single 5-V Power Supply**
- **Operate Up to 120 kbit/s**
- **Two Drivers and Two Receivers**
- **±30-V Input Levels**
- **Low Supply Current . . . 8mAT ypical**
- **Designed to be Interchangeable With Maxim MAX232**
- **ESD Protection Exceeds JESD 22 – 2000-V Human-Body Model (A114-A)**
- **Applications**
  - **TIA/EIA-232-F**
  - **Battery-Powered Systems**
  - **Terminals**
  - **Modems**
  - **Computers**

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube	MAX232N	MAX232N
	SOIC (D)	Tube	MAX232D	MAX232
		Tape and reel	MAX232DR	
	SOIC (DW)	Tube	MAX232DW	MAX232
		Tape and reel	MAX232DWR	
	SOP (NS)	Tape and reel	MAX232NSR	MAX232
–40°C to 85°C	PDIP (N)	Tube	MAX232IN	MAX232IN
	SOIC (D)	Tube	MAX232ID	MAX232I
		Tape and reel	MAX232IDR	
	SOIC (DW)	Tube	MAX232IDW	MAX232I
		Tape and reel	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

MAX232, MAX232I
DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047I – FEBRUARY 1989 – REVISED OCTOBER 2002

DRIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (see Note 3)

PARAMETER			TEST CONDITIONS	MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	T1OUT, T2OUT	R_L = 3 kΩ to GND	5	7		V
V_{OL}	Low-level output voltage‡	T1OUT, T2OUT	R_L = 3 kΩ to GND		–7	–5	V
r_o	Output resistance	T1OUT, T2OUT	V_{S+} = V_{S-} = 0, V_O = ±2 V	300			Ω
I_{OS}§	Short-circuit output current	T1OUT, T2OUT	V_{CC} = 5.5 V, V_O = 0		±10		mA
I_{IS}	Short-circuit input current	T1IN, T2IN	V_I = 0			200	∝A

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
§ Not more than one output should be shorted at a time.
NOTE 3: Test conditions are C1–C4 = 1 ∝F at V_{CC} = 5 V ± 0.5 V.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Note 3)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
SR	Driver slew rate	R_L = 3 kΩ to 7 kΩ, See Figure 2			30	V/∝s
SR(t)	Driver transition region slew rate	See Figure 3		3		V/∝s
	Data rate	One TOUT switching		120		kbit/s

NOTE 3: Test conditions are C1–C4 = 1 ∝F at V_{CC} = 5 V ± 0.5 V.

RECEIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (see Note 3)

PARAMETER			TEST CONDITIONS	MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	R1OUT, R2OUT	I_{OH} = –1 mA	3.5			V
V_{OL}	Low-level output voltage‡	R1OUT, R2OUT	I_{OL} = 3.2 mA			0.4	V
V_{IT+}	Receiver positive-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V, T_A = 25°C		1.7	2.4	V
V_{IT-}	Receiver negative-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V, T_A = 25°C	0.8	1.2		V
V_{hys}	Input hysteresis voltage	R1IN, R2IN	V_{CC} = 5 V	0.2	0.5	1	V
r_i	Receiver input resistance	R1IN, R2IN	V_{CC} = 5, T_A = 25°C	3	5	7	kΩ

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
NOTE 3: Test conditions are C1–C4 = 1 ∝F at V_{CC} = 5 V ± 0.5 V.

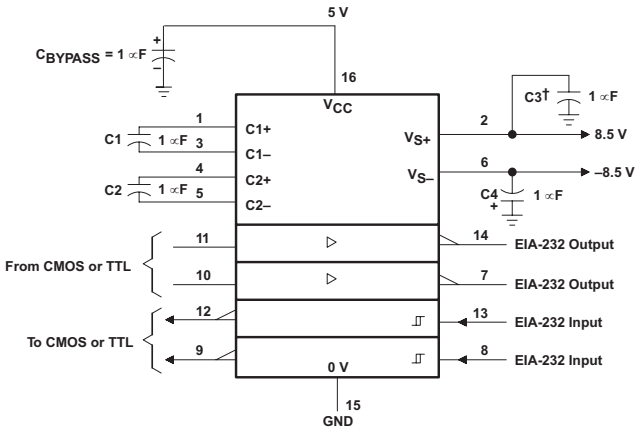
switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Note 3 and Figure 1)

PARAMETER		TYP	UNIT
$t_{PLH(R)}$	Receiver propagation delay time, low- to high-level output	500	ns
$t_{PHL(R)}$	Receiver propagation delay time, high- to low-level output	500	ns

NOTE 3: Test conditions are C1–C4 = 1 ∝F at V_{CC} = 5 V ± 0.5 V.

SLLS047I – FEBRUARY 1989 – REVISED OCTOBER 2002

APPLICATION INFORMATION



† C3 can be connected to V_{CC} or GND.

Figure 4. Typical Operating Circuit

Apéndice F

PROGRAMA EN ENSAMBLADOR

```
                List P=16F877
                include "P16F877.INC"

OHMS1H equ 0x21
OHMS1L equ 0x22
OHMS2H equ 0x23
OHMS2L equ 0x24
OHMS3H equ 0x25
OHMS3L equ 0x26
OHMS4H equ 0x27
OHMS4L equ 0x28
OHMS5H equ 0x29
OHMS5L equ 0x30
OHMS6H equ 0x31
OHMS6L equ 0x32
portd equ 0x08
                org 0X00
                goto inicio

;CONFIGURACION DEL USART DE PIC TRABAJANDO A 9600 BAUDIOS

inicio

        clrf PORTC
        bsf STATUS,RP0          ;Selecciona al Banco 1
        movlw b'10111111'   ;Configura el puerto C como entradas y salidas
        movwf TRISC
        movlw b'00000000'  ;Configura el puerto D como entradas y salidas
        movwf TRISD
        movlw b'101'

                movwf TRISE
                movlw b'00111111'
                movwf TRISA

        movlw b'11000111'
        movwf OPTION_REG
        movlw  b'00101001' ;Configura los bit del registro TXSTA
        movwf TXSTA                     ;(Habilita transmisicion a una velocidad de 9600 BD
        movlw .31               ;Cosntante para generar la velocidad TX
        movwf SPBRG
        bcf STATUS, RP0         ;Selecciona al Banco 0
        bsf RCSTA, SPEN         ;Selecciona Tx como transmisor Asincrono activa a SPEN CON 1
        clrw


;TABLA DE BARRIDO DE LOS POTENCIOMETROS Y ENVIO DE LOS DATOS
Loop addwf PCL, 1
        goto checar1
        goto checar2
        goto checar3
        goto checar4
        goto checar5
        goto checar6
        goto datos_n
fin movlw 0
reinicio goto Loop
```

```
; FUNCION QUE ENVIA LOS DATOS AL PUERTO SERIE

datos_n

        movlw '1'
        call transmite
        movf OHMS1H, 0
        movwf portd
        call transmite

        movlw '2'
        call transmite
        movf OHMS2H, 0
        call transmite

        movlw '3'
        call transmite
        movf OHMS3H, 0
        call transmite

        movlw '4'
        call transmite
        movf OHMS4H, 0
        call transmite

        movlw '5'
        call transmite
        movf OHMS5H, 0
        call transmite

        movlw '6'
        call transmite
        movf OHMS6H, 0
        movwf portd
        call transmite
        goto fin

;TRANSMITE POR EL PUERTO SERIAL LO QUE TIENE W

transmite
                movwf TXREG
                bsf STATUS, RP0
manda           btfss TXSTA, TRMT
                goto manda
                bcf STATUS, RP0
                return

;FUNCIONES DE DONDE SE ANALIZAN LAS ENTRADAS ANALOGICAS Y SE PASAN A DIGITALES

;POTENCIOMETRO1 AN . 0
checar1     movlw b'10000001'
            call configura_a_dc
            movf ADRESH,0
            movwf OHMS1H
            bsf STATUS, RP0
            movf ADRESL,0
            movwf OHMS1L
            bcf STATUS,RP0
```

```
                movlw 1
                goto reinicio

;POTENCIOMETRO2 AN . 1
checar2         movlw b'10001001'
                call configura_a_dc
                movf ADRESH,0
                movwf OHMS2H
                bsf STATUS, RP0
                movf ADRESL,0
                movwf OHMS2L
                bcf STATUS,RP0
                movlw 2
                goto reinicio

;POTENCIOMETRO3 AN . 2
checar3         movlw b'10010001'
                call configura_a_dc
                movf ADRESH,0
                movwf OHMS3H
                bsf STATUS, RP0
                movf ADRESL,0
                movwf OHMS3L
                bcf STATUS,RP0
                movlw 3
                goto reinicio

;POTENCIOMETRO 4 AN . 4
checar4         movlw b'10100001'
                call configura_a_dc
                movf ADRESH,0
                movwf OHMS4H
                bsf STATUS, RP0
                movf ADRESL,0
                movwf OHMS4L
                bcf STATUS,RP0
                movlw 4
                goto reinicio

;POTENCIOMETRO5 AN . 5
checar5         movlw b'10101001'
                call configura_a_dc
                movf ADRESH,0
                movwf OHMS5H
                bsf STATUS, RP0
                movf ADRESL,0
                movwf OHMS5L
                bcf STATUS,RP0
                movlw 5
                goto reinicio

;POTENCIOMETRO 6 AN . 6
checar6         movlw b'10110001'
                call configura_a_dc
                movf ADRESH,0
                movwf OHMS6H
                bsf STATUS, RP0
                movf ADRESL,0



                movwf OHMS6L
                bcf STATUS,RP0
                movlw 6
                goto reinicio

;FUNCION DEL CONVERTIDOR  ANALOGICO A DIGITAL

configura_a_dc:

                movwf ADCON0
                bsf STATUS, RP0

                movlw b'100000001'
                movwf ADCON1
                bcf STATUS, RP0
                bcf PIR1, ADIF
                nop
                bsf ADCON0, GO          ;Verifica si ha terminado la conversión
cambia  btfss PIR1,ADIF
                goto cambia
                return
; FIN
                end
```

Apéndice G

ESTRUCTURA DE UN PROGRAMA UTILIZANDO OpenGL

La estructura básica de un programa es muy sencilla, y siempre consta de las siguientes partes:

- Funciones de inicio de ventana.
- Arranque del Sistema Gráfico (con GLUT1).
- Creación e Inicialización de una ventana de trabajo.
- Funciones CallBack.
- Dibujar Objetos tridimensionales.
- Se posiciona en el espacio 3D
- Vista Tridimensional.

FUNCIONES DE INICIO DE VENTANA.

La primera rutina abre una ventana en la pantalla del ordenador. Esta función evidentemente no pertenece a OpenGL y por lo tanto, debe tomarse del gestor de ventanas que utilice cada máquina en concreto
int main(int argc, char **argv)

ARRANQUE DEL SISTEMA GRÁFICO (con GLUT1)

Incluir los archivos de cabecera necesarios en "main"():

En "main"inicializa de la librería GLUT . glutInit(). **glutInit** (int *argc, char **argv). Esta función inicializa GLUT y procesa la línea de comandos (para X-Windows). Además debe ser invocada antes que cualquier otra función de GLUT.

Activación del modo de dibujo . glutInitDisplayMode(). **glutInitDisplayMode** (unsigned int mode) Especifica que características se utilizaran a la hora de representar la imagen en pantalla, el buffer de profundidad, el formato del color, doble buffer, por ejemplo seria: ($GLUT\_DOUBLE \mid GLUT\_RGB \mid GLUT\_DEPTH$).

`GLUT_RGB`

(modelo de color con el que se dibujará -rojo-verde-azul)

`GLUT_DEPTH`

(incluir un buffer de profundidad)

`GLUT_DOUBLE`

(se debe utilizar un doble buffer).

CREACIÓN E INICIALIZACIÓN DE UNA VENTANA DE TRABAJO

Para construir un ambiente virtual con OpenGL es necesario tener un espacio de trabajo; una ventana donde se desarrollará la animación del ambiente virtual en este espacio de trabajo se definen algunas características tales como tamaño de la ventana, posición de la ventana, color del fondo (background), encabezado de la ventana etc.

1. **glutInitWindowPosition** (int xm int y) Especifica la posición de la ventana desde la esquina superior izquierda de la pantalla en pixels.
2. **glutInitWindowSize** (int winth, int size) Especifica el tamaño en pixels de la ventana.
3. **glutCreateWindow** (char *string) Esta función crea una ventana con un título. Esta función devuelve un único identificador por ventana, aunque la ventana no es presentada hasta que se ejecuta la función glutMainLoop ().

FUNCIONES CallBack

Es habitual que los programas OpenGL se ejecuten gracias a un "Bucle de Eventos". Tras la creación e inicialización de la ventana de trabajo, el programa entra en un bucle infinito, que acepta entradas y ejecuta los eventos asociados a las mismas. Estos eventos son ser pulsaciones de teclas, movimiento, pulsaciones o liberación del ratón, ocultación, muestra o cambio de tamaño de las ventanas, etc. Las instrucciones encargadas de instruir al programa sobre qué hacer en relación a cada evento son las Funciones CallBack, explicadas a continuación.

Son un interfaz de programación para entradas orientadas a eventos. Se define una de estas funciones para cada evento previsto en el programa. La función particular, definida por el usuario, se ejecuta cuando se produce el evento. En todo programa glut debe haber al menos una función callback que dibuja la pantalla. Diferentes funciones callback:

1. **Display:** Esta función es invocada cuando el sistema determina que el contenido de la ventana tiene que ser redibujado, por ejemplo cuando la ventana se abre, o arrastramos otra ventana por delante de esta. La función de glut encargada de registrar este evento es

`glutDisplayFunc()`.

La función que registramos no tiene ningún argumento de entrada.

2. **Reshape:** La ventada de la aplicación puede cambiar de tamaño, normalmente porque el usuario arrastre alguno de los bordes de la ventana con el ratón. La función que registra este evento es

   `glutReshapeFunc()`.

   Y la función que nosotros registremos para manejar el evento debe tener dos parámetros enteros. Cuando se ejecute esta función de redimensión estos dos parámetros contendrán los nuevos valores del tamaño de la ventana.

3. **Mouse:** Esta función va a manejar los eventos que sean producidos por el ratón. Por ejemplo, que uno de los botones se presione o se suelte. La función que registra este evento es

   `glutMouseFunc()`.

   Al igual que en el caso anterior, esta función, al ser invocada pasa argumentos a la función de callback. En este caso estos argumentos describen la posición del ratón y el evento que se acaba de producir.

4. **Keyboard:** Esta función es muy parecida a la anterior. En este caso en lugar del ratón se trata de un evento de teclado. La función de glut que registra el callback es

   `glutKeyboardFunc()`.

   Y como viene siendo normal, a la función de callback también se le pasa información sobre la tecla en la que se ha producido el evento. Como caso excepcional, al callback también se le suministra información sobre la posición del ratón en el momento que se pulsó la tecla.

Estas son las cuatro funciones de callback más importantes, y las que más se usan, pero glut incluye muchas otras. Si bien no se suelen usar mucho merece la pena conocer: Idle, Timer, Special y Joystick.

Por otra parte hay que comentar que no todos los callbacks están disponibles en todas las versiones de glut. Según esta ha ido avanzando se han introducido nuevas funciones. Por ejemplo

`glutJoystickFunc()`,

esta función se añadió en la versión 4 de glut. Si escribimos un programa que haga uso de ella, va a ser necesario que se ejecute siempre sobre glut 4 o superior.

DIBUJAR OBJETOS TRIDIMENSIONALES

En esta sección se explican una serie de funciones básicas que nos permiten empezar hacer pruebas. En primer lugar veremos como limpiar la ventana y dejarla preparada para empezar a dibujar, como dar un color a los objetos y hacer un dibujo completo, basado en una serie de primitivas.

Limpiar la Ventana

Cuando abrimos una ventana nueva para dibujar sobre ella, la zona de memoria sobre la que se define, puede no estar limpia, sino tener almacenado parte del último dibujo que se realizó o bien basura. Por tanto en general será necesario limpiar esa ventana y darle el color de fondo (background) adecuado.

Existen comandos específicos para limpiar las ventanas sobre las que se va a trabajar y poner el color de fondo adecuado, sobre todo por motivos de simplicidad y de eficiencia.

```
void glClearColor (GLclampf red, GLclampf green, GLclampf blue,
GLclampf alpha);
```

Activa el color de fondo actual, siguiendo el modelo de representación RGB. Cada argumento representa la cantidad del color indicado que tiene el color de fondo, y toma valores en el intervalo [0, 1]. El valor por defecto es el color NEGRO que se representa por el valor: (0, 0, 0, 0).

Especifica el color

En OpenGL la descripción geométrica de un objeto es independiente de su apariencia externa, es decir del color. OpenGL ofrece varios esquemas de color con los que se puede trabajar en función de la complejidad y de la calidad que se quiera dar a las imágenes. A cada uno de los modelos posibles se les denomina esquemas de color.

La función glColor3f(), que asigna a la variable de estado correspondiente el color actual, en el que se van a dibujar todos los objetos. Esta función toma tres parámetros que corresponden como en el caso de la función glClear(), con las cantidades de rojo, verde y azul, que componen el color seleccionado.

```
glColor3f (1.0, 1.0, 1.0); blanco
```

Dibujar Solidos

Por ejemplo, para dibujar una esfera se emplea la función:

```
gluSphere(quadratic, radio, param1, param2);
```

A este objeto se le proporcionan una característica principalmente el color.

Definición de vértices

En OpenGL la geometría de todos los objetos está definida en último término como un conjunto de vértices ordenados. Para especificar estos vértices utilizamos la función glVertex ().

```
void glVertex{234}{sifd}{v} (TYPE coords);
```

Especifica las coordenadas de un vértice que puede utilizarse para describir la geometría de un objeto. Como se ve en la definición pueden especificarse 2 ó 3 coordenadas, y en varios formatos distintos. Las llamadas a esta función deben ejecutarse entre un par de llamadas glBegin() y glEnd().

Primitivas Geométricas

Una vez que hemos visto como se define un vértice vamos a ver como se pueden definir puntos, líneas y polígonos, a partir de esos vértices. Para crear cualquier figura geométrica en OpenGL hay que definir los vértices entre un par de llamadas a las funciones glBegin() y glEnd(). A la función glBegin se le puede pasar un argumento que determina qué tipo de figura geométrica se va construir. Después se definen los vértices mediante llamadas a la función glVertex.

```
void glBegin (GLenum mode);
```

Marca el principio de una lista de vértices que describen una primitiva geométrica. El parámetro mode índica el tipo de primitiva, que puede ser en este caso GL_QUADS Cada cuátrupla determina un cuadrado.

SE POSICIONAN EN EL ESPACIO 3D

Vista Tridimensional

En esta sección se vera como ubicar los objetos y el punto de vista del observador dentro de un espacio tridimensional, para obtener la vista de la escena que deseemos.

Para crear una imagen plana es decir bidimensional sobre la pantalla del ordenador, a partir de una descripción tridimensional de la escena, tenemos que realizar una serie de operaciones, que incluyen la proyección y el recorte de los objetos sobre un volumen de vista específico. Basándose en OpenGL estas operaciones son las siguientes:

- Transformaciones, que se utilizan para operaciones de modelado, vista y proyección.
- Recorte de objetos o porciones de objetos no visibles.
- Correspondencia entre coordenadas 2D y pixeles de la pantalla.

Transformación de propósito general

```
void glMatrixMode (GLenum mode)
```

Especifica cual de las matrices se va a modificar según el argumento mode. Las matrices que se pueden modificar son la matriz de modelo y vista combinadas, la matriz de proyección y la matriz de textura, que no utilizaremos en este curso.

Los parámetros que se utilizan para cada caso se reflejan en la tabla G.1.

Tabla 5.1: Comandos.

Argumento	**Matriz**
GL_MODELVIEW	Matriz de modelo y vista
GL_PROJECTION	Matriz de proyección
GL_TEXTURE	Matriz de Textura

Los siguientes comandos de transformación afectan a la matriz especificada por esta función. Sólo puede modificarse una matriz en cada operación, y estará activa la última indicada por la función anterior. Por defecto se modifica la matriz de modelo y vista (ModelView).

Al comienzo todas las matrices están inicializadas a la matriz identidad.

```
void glLoadIdentity (void)
```

Inicializa la matriz actual, con los valores de la matriz identidad de 4x4. Se utiliza para limpiar la matriz actual, al comienzo o bien para volver a la posición inicial del objeto.

```
void glLoadMatrix{fd} (const TYPE *m)
```

Especifica una matriz a partir del puntero ŞmŤ para cargarla en la matriz actual. El argumento m es un vector de 16 (4x4) componentes.

Transformaciones de Módulo

Estas operaciones sirven para transformar la posición, orientación y tamaño del modelo o modelos que forman la escena. Las operaciones básicas que tenemos son las ya vistas en clase como:

- Trasladar..
- Rotar.
- Escalar.

Las funciones de OpenGL para realizar estas operaciones son las siguientes:

```
void glTraslate{fd} (TYPE x, TYPE y, TYPE z)
```

Multiplica la matriz actual por una matriz de traslación, que desplaza un objeto por las distancias de traslación x, y, z. Este desplazamiento se realiza desde la posición actual del objeto hasta alcanzar la posición final, es decir utiliza desplazamientos relativos.

```
void glRotate{fd} (TYPE angle, TYPE x, TYPE y, TYPE z)
```

Multiplica la matriz actual por una matriz de rotación que gira el objeto en el sentido contrario al del giro de las agujas del reloj, sobre el eje que va desde el origen de coordenadas al punto (x, y, z) especificado como parámetro, un ángulo igual al especificado como argumento.

```
void glScale{fd} (TYPE x, TYPE y, TYPE z)
```

Multiplica la matriz actual por un matriz de escalado, que escala el objeto en los tres ejes de coordenadas X, Y, Z, por los factores de escala x, y, z. Cada coordenada de un punto (x, y, z) se multiplica por los factores de escala especificados como parámetros.

Transformaciones de vista

Las transformaciones de vista cambian la posición y orientación del punto de vista del observador. En general se componen de operaciones básicas de transformación, sobre todo traslaciones y rotaciones.

Las transformaciones de vista deben realizarse antes de cualquier transformación de modelo.

La ubicación por defecto del observador es el origen de coordenadas, y orientado hacia el eje Z negativo, es decir mira hacia el interior de la pantalla.

Existen varias formas de ubicar la posición final del observador en un programa realizado con OpenGL. A continuación señalamos algunas de ellas:

1. Utilizando las rutinas de traslación y rotación. De esta forma a partir de la posición inicial ya descrita podemos mover la cámara a la posición adecuada por medio de estas rutinas.

2. Utilizar la rutina gluLookAt () que pertenece a las librerías auxiliares de OpenGL, para indicar hacia donde mira la cámara.

3. Crear rutinas de utilidad propias, que calculen todos los movimientos necesarios, según los parámetros que queramos darles.

A continuación veremos las dos primeras de estas formas, pues son las que se utilizaron en este trabajo.

Rutina glTraslate()

Como ya hemos mencionado la cámara está inicialmente colocada en el origen de coordenadas, mirando hacia el eje Z-negativo. El caso más simple es el de moverla hacia atrás, es decir alejarla, de los objetos que componen la escena. Para esto se utiliza la llamada a la función

`glTraslate ()`.

Utilización de la Rutina gluLookAt ().

```
void gluLookAt (GLdouble eyex, GLdouble eyey, GLdouble eyez,
GLdouble centrox, GLdouble centroy, GLdouble centroz, GLdouble
upx, GLdouble upy, GLdouble upz)
```

Define una matriz de vista y la postmultiplica por la matriz actual.

La ubicación viene indicada por los puntos eyex, eyey, eyez.

Los parámetros centrox, centroy, centroz, determinan el punto al cual mira la cámara, que en general es el centro de la escena.

Los último los argumentos upx, upy, upz indican la dirección de abajo arriba del volumen de vista.

Esta función engloba varias rutinas básicas de OpenGL, sobre todo rutinas de traslación y rotación.

Proyección Perspectiva.

Este método de proyección se utiliza en aplicaciones de simulación y en cualquier otra aplicación que requiera realismo, ya que la imagen es similar a la que capta el ojo humano.

La función que define la pirámide es la siguiente:

```
void gluPerspective (GLdouble fovy, GLdouble aspect, GLdouble
zNear, GLdouble zFar)
```

Esta función crea una matriz de proyección que define una vista simétrica y la multiplica por la matriz actual. El argumento fovy determina el ángulo del campo de vista, en el plano X-Z y su rango es $[0°, 180°]$. El parámetro aspect es el cociente entre el ancho de la pirámide dividido por el alto de la base de la pirámide. Los valores zNear y zFar son las distancias desde el punto de vista a los planos de recorte. Deben ser siempre positivos.

Apéndice H

```
#include <windows.h>                /* MS Windows requiere que este header se incluya antes de gl.h y glu.h */
//Librerias utilizadas para el desarrollo de la interfaz
//#include "stdafx.h"
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>

#include "xyz.h"
#include "VariablesRS232.h"
//#include "stdafx.h>"

#include <gl\glut.h>
#include <gl\gl.h>
#include <gl\glu.h>

//Variables globales utilizadas para la manipulacion de la iluminacion
float LightPos0[] = { 0.75f, 1.0f, 1.0f, 0.0f};
float LightAmb0[] = { 0.75f, 0.75f, 0.75f, 1.0f};
float LightDif0[] = { 1.0f,1.0f, 1.0f, 1.0f};
float LightSpc0[] = { 1.0f, 1.0f, 1.0f, 1.0f};

//Variables globales utilizadas para la manipulacion de la camara
static float angulo=0.0,razon;


static float angulo1=0.0,angulo2=0.0,angulo3=0.0,ratio;
static float x=0.0f,y=1.0f,z=5.0f;
static float lx=0.0f,ly=0.0f,lz=-1.0f;
//static float Ux=0.0f,Uy=1.0f,Uz=0.0f;
static float vx=0.0f,vy=1.0f,vz=0.0f;

static GLint muneco_de_nieve;
float a=0,b=0,c=0;
bool conectar;

void EscalaVentana(int w, int h)
        {
        // Evita una división por cero
        if(h == 0)
                h = 1;
        ratio = 1.0f * w / h;
        // Reinicia el sistema de coordenadas
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();

        // Ajusta la vista a las dimensiones de la ventana
    glViewport(0, 0, w, h);

        // Establece el volumen de trabajo
        gluPerspective(45,ratio,1,100);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        gluLookAt(x ,y , z,
                x + lx,y + ly,z + lz,
                        vx,vy,vz);
        }
```

```
void Dibujar_hombre_de_nieve() {
	glColor3f(1.0f, 1.0f, 1.0f);

// Dibujar cuerpo
	glTranslatef(0.0f ,0.75f, 0.0f);
	glutSolidSphere(0.75f,20,20);


// Dibujar cabeza
	glTranslatef(0.0f, 1.0f, 0.0f);
	glutSolidSphere(0.25f,20,20);

// Dibuajr ojos
	glPushMatrix();
	glColor3f(0.0f,0.0f,0.0f);
	glTranslatef(0.05f, 0.10f, 0.18f);
	glutSolidSphere(0.05f,10,10);
	glTranslatef(-0.1f, 0.0f, 0.0f);
	glutSolidSphere(0.05f,10,10);
	glPopMatrix();

// Dibujar nariz
	glColor3f(1.0f, 0.5f , 0.5f);
	glRotatef(0.0f,1.0f, 0.0f, 0.0f);
	glutSolidCone(0.08f,0.5f,10,2);
}


//Funcion encargada de generar un lista de objetos y ponerlos en pantalla
GLuint createDL() {
	GLuint snowManDL;

	// Crea el id de la lista
	snowManDL = glGenLists(1);

	// inicia la lista
	glNewList(snowManDL,GL_COMPILE);

	// llama a la funcion con las rutinas encargadas del render
		Dibujar_hombre_de_nieve();

	// Fin de la lista
	glEndList();

	return(snowManDL);
}

void Conectar(){
		conectar=true;
		while (true)
 {
 // Abrir el puerto
 hCom=CreateFile("COM1",
     GENERIC_READ|GENERIC_WRITE,
     0,0,OPEN_EXISTING,
```

```
    FILE_ATTRIBUTE_NORMAL,0);
 // Si error finalizar
 if (hCom==INVALID_HANDLE_VALUE) break;

 // Leer configuración actual
 if (!GetCommState(hCom,&sComCfg)) break;

 // Cambiar configuración
 sComCfg.BaudRate=CBR_9600;
 sComCfg.ByteSize=8;
 sComCfg.Parity=NOPARITY;
 sComCfg.StopBits=ONESTOPBIT;
 sComCfg.fRtsControl=RTS_CONTROL_ENABLE;

 // Escribir nueva configuración
 if (!SetCommState(hCom,&sComCfg)) break;

 // Configurar TIMEOUTS
 // Para que la operación de lectura
 // finalice inmediatamente
 sTimOut.ReadIntervalTimeout=MAXDWORD;
 sTimOut.ReadTotalTimeoutMultiplier=0;
 sTimOut.ReadTotalTimeoutConstant=0;
 sTimOut.WriteTotalTimeoutMultiplier=0;
 sTimOut.WriteTotalTimeoutConstant=0;
 sTimOut.WriteTotalTimeoutConstant=0;
        if (!SetCommTimeouts(hCom,&sTimOut)) break;

        // Contruimos la cadena a enviar
        //  strcpy(acBuf,"Hola Mundo");
        dwLen=10;
 // Transmitimos la cadena
 // Salir del bucle principal
  break;
 }

}

void Desconectar(){

if(hCom!=INVALID_HANDLE_VALUE)
CloseHandle(hCom);
}

void procesaTecladoNormal(unsigned char key, int x, int y) {

        switch(key){
        case 27:
                exit(0);
                Desconectar();
                break;
        case 'C':
                Conectar();
                break;
        case 'D':
                Desconectar();
                break;
        }
```

```
}

//Funcion encargada de Inicializar la Ventana generada por la Glut
void InicioEscena()
	//{
//
//	glEnable(GL_DEPTH_TEST);
//	muneco_de_nieve = createDL();
	//}
{
	glClearColor(0.0, 0.0, 0.0, 0.0);
	glShadeModel(GL_SMOOTH);
	glCullFace(GL_BACK);
	glEnable(GL_DEPTH_TEST);
	glEnable(GL_CULL_FACE);
	glLightfv(GL_LIGHT0, GL_POSITION, LightPos0);
	glLightfv(GL_LIGHT0, GL_AMBIENT, LightAmb0);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, LightDif0);
	glLightfv(GL_LIGHT0, GL_SPECULAR, LightSpc0);
	glEnable(GL_LIGHT0);
	glEnable(GL_LIGHTING);
	glEnable(GL_DEPTH_TEST);
	muneco_de_nieve = createDL();
}


void DibujarEscena(void) {

	{
	 glClearColor (0., 0., 1., 1.);			// Pone el color de fondo a azul
    glClear(GL_COLOR_BUFFER_BIT);			// Borra la ventana
    glFlush();						// forzamos el dibujo
	}

	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

// Draw ground "suelo"

	glColor3f(0.9f, 0.9f, 0.9f);
	glBegin(GL_QUADS);
		glVertex3f(-100.0f, 0.0f, -100.0f);
		glVertex3f(-100.0f, 0.0f,  100.0f);
		glVertex3f( 100.0f, 0.0f,  100.0f);
		glVertex3f( 100.0f, 0.0f, -100.0f);
	glEnd();

 //Dibujar 36 muñecos de nieve

	for(int i = -3; i < 3; i++)
		for(int j=-3; j < 3; j++) {
			glPushMatrix();
			glTranslatef(i*10.0,0,j * 10.0);
			glCallList(muneco_de_nieve);
			glPopMatrix();
		}

	glutSwapBuffers();   //se encarga de intercambiar el buffer posterior con el buffer anterior
}
```

```
void Yaw(float ang1) {
        vy = cos(ang1);
        vx = sin(ang1);
        glLoadIdentity();
        gluLookAt(x, y, z,
                x + lx,y+ly ,z + lz ,
                    vx,vy,vz );
}
void Pitch(float ang2) {
        ly = sin(ang2);
        lz = -cos(ang2);
        glLoadIdentity();
        gluLookAt(x, y, z,
                lx+x , y+ly ,lz+z  ,
                    vx,vy,vz );
}
void Roll(float ang3) {
        lx = sin(ang3);
        lz = -cos(ang3);
        glLoadIdentity();
        gluLookAt(x, y, z,
                x + lx,y + ly,z + lz ,
                    vx,vy,vz );
}
void moverx(float i) {
        x = x + i*(1)*0.01;
        glLoadIdentity();
        gluLookAt(x, y, z,
                x+lx ,y+ly ,z+lz ,
                    vx,vy,vz );
}
void movery(float j) {
        y = y + j*(1)*0.01;
        glLoadIdentity();
        gluLookAt(x, y, z,
                x + lx,y + ly,z + lz,
                    vx,vy,vz );
}
void moverz(float k) {
        y = y + k*(ly)*0.1;
        z = z + k*(lz)*0.1;
        glLoadIdentity();
        gluLookAt(x, y, z,
                x + lx,y + ly,z + lz,

                    vx,vy,vz );
}


void DawnRecibe()
{
BYTE  bDatRx;   // Buffer para lo
int i;
bool uno=false;
bool dos=false;
bool tres=false;
bool cuatro=false;
bool cinco=false;
```

```
bool seis=false;

for(i=0;i<200;i++){
        if(ReadFile(hCom,&bDatRx,1,&dwBytRea,0)){
          if(uno==true){
                    vector[0]=calcu(120,255,bDatRx);
                    uno=false;
                }
                if(dos==true){
                vector[1]=calcu(120,255,bDatRx);
                    dos=false;
                }
                if(tres==true){
         vector[2]=calcu(120,255,bDatRx);
                    tres=false;
                }
                if(cuatro==true){
                     vector[3]=calcu(120,255,bDatRx);
                     cuatro=false;
                }
                if(cinco==true){
                     vector[4]=calcu(120,255,bDatRx);
                     cinco=false;
                }
                if(seis==true){
                     vector[5]=calcu(120,255,bDatRx);
                     seis=false;
                }

               if(bDatRx=='1')
               uno=true;

               if(bDatRx=='2')
               dos=true;

               if(bDatRx=='3')
               tres=true;

               if(bDatRx=='4')
               cuatro=true;

               if(bDatRx=='5')
               cinco=true;

               if(bDatRx=='6')
               seis=true;

      }
}

if(conectar==true){
///////Roll
        if(vector[0]<50)
        angulo -= 0.00915f;Roll(angulo);
        if(vector[0]>310)
        angulo += 0.0095f;Roll(angulo);
        if(vector[0]>50&&vector[0]<310)
        angulo =angulo;Roll(angulo);
```

```
///////Y
        if(vector[2]<50)
        movery(-0.051f);
        if(vector[2]>310)
        movery(0.051f);
        if(vector[2]>50&&vector[2]<310)
        movery(0.000f);

///////X
        if(vector[1]<50)
        moverx(-.0022f);
        if(vector[1]>310)
        moverx(.0022f);
        if(vector[1]>50&&vector[1]<310)
        moverx(0.000f);
///////Z
        if(vector[3]<50)
        moverz(-.0022f);
        if(vector[1]>310)
        moverz(.0022f);
        if(vector[1]>50&&vector[1]<310)
        moverz(0.000f);
///////Yaw
        if(vector[1]<50)
        Yaw(-.0022f);
        if(vector[1]>310)
        Yaw(.0022f);
        if(vector[1]>50&&vector[1]<310)
        Yaw(0.000f);
/////// Pitch
        if(vector[1]<50)
        Pitch(-.0022f);
        if(vector[1]>310)
        Pitch(.0022f);
        if(vector[1]>50&&vector[1]<310)
        Pitch(0.000f);
}
PurgeComm(hCom,PURGE_RXCLEAR);

}

void Teclas(unsigned char key, int x, int y) {
switch (key) {
                case '1' : angulo1 += 3.1416f/180;Yaw(angulo1);break;
                case '2' : angulo1 -= 3.1416f/180;Yaw(angulo1);break;
                case '3' : angulo2 -= 3.1416f/180;Pitch(angulo2);break;
                case '4' : angulo2 += 3.1416f/180;Pitch(angulo2);break;
                case '5' : angulo3 -= 3.1416f/180;Roll(angulo3);break;
                case '6' : angulo3 += 3.1416f/180;Roll(angulo3);break;
                case '7' : moverx(10);break;
                case '8' : moverx(-10);break;
                case '9' : movery(10);break;
                case '0' : movery(-10);break;
                case 'q' : moverz(10);break;
                case 'w' : moverz(-10);break;
                case 'a' : glutFullScreen ();break;
}
```

```
        if (key == 27)
        exit(0);
}
void menu(int item)
{
procesaTecladoNormal((unsigned char)item,0,0);
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);

        // Funciones GLUT para inicializar la ventana
        glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);


        glutInitWindowPosition(100,100);
        glutInitWindowSize(640,360);
        glutCreateWindow("Ambiente Virtual");

                InicioEscena();

                // Indicamos la función para el evento 'Pintar'
                glutKeyboardFunc(Teclas);
                glutDisplayFunc(DibujarEscena);
                glutIdleFunc(DibujarEscena);

                // Indicamos la función para el evento cambiar tamaño
                glutReshapeFunc(EscalaVentana);

        glutCreateMenu(menu);
        glutAddMenuEntry("C--->Conectar DawnWay  ", 'C');
        glutAddMenuEntry("D--->Desconectar DawnWay ", 'D');
        glutAddMenuEntry("ESC->Salir", 27);
        glutAttachMenu(GLUT_RIGHT_BUTTON);


        // Lanzamos el bucle indefinido de evento
        glutMainLoop();
        return(0);
}
```