CENTRO DE INGENIERIA Y DESARROLLO

CENTRO DE INGENIERIA Y DESARROLLO INDUSTRIAL

Proyecto Industrial Terminal

Instrumentación de un dispositivo para la reconstrucción tridimensional de objetos

PARA OBTENER LA ESPECILIDAD EN

"TECNÓLOGO EN MECATRÓNICA"

PRESENTA

Alumno: **ISC. Hugo Caudillo Reyes**

Tutor de Planta: Dr. Carlos Pedraza Ortega
Tutor Académico: Dr. Carlos Pedraza Ortega

QUERETARO, QRO. OCTUBRE 2007.

Resumen

El proyecto consiste en la instrumentación de un dispositivo que lleva a cabo la reconstrucción tridimensional de un objeto utilizando proyección de línea láser.

El dispositivo esta integrado por diferentes módulos: El modulo de adquisición de imágenes, el modulo de control de movimiento del objeto, el módulo de comunicación y la interfaz de usuario que integra y controla los diferentes módulos del sistema.

El módulo de adquisición de imágenes lo integran una cámara USB y un láser. El módulo de control de movimiento del objeto esta integrado por un motor, la base donde se pondrá el objeto y toda la instrumentación electrónica para el control del motor. El módulo de comunicación lo integra el cable para puerto paralelo así como su electrónica.

A través de la interfaz de usuario se pueden controlar y definir los parámetros necesarios para llevar a cabo la captura de las imágenes, el movimiento del objeto y el procesamiento de las imágenes para su posterior reconstrucción. Algunos de los parámetros que se configuran son los siguientes:

- Número de imágenes a adquirir.
- Velocidad del motor que controla el movimiento del objeto.
- Número de pasos que requiere el motor para dar una vuelta completa.
- Directorio y nombre de los archivos que tendrán las imágenes adquiridas.
- Tiempo de espera que tendrá el sistema entre el último paso dado y la toma de la imagen.
- Los puntos que definen el ROI de la imagen

El proceso consiste en rotar el objeto a través del movimiento del motor; El objeto es incidido por un haz de rayo láser. Mientras el objeto esta en rotación, se capturan las imágenes y se graban en archivos con formato JPEG para su posterior procesamiento y reconstrucción en 3D.

Agradecimientos

A Dios, por darme la vida y porque sin Èl no estaría aquí.

A mi familia, por todo el apoyo y confianza que me han brindado durante y en todos los proyectos que he emprendido. Especialmente a ti papa que me enseñaste que para alcanzar algo es necesario trabajar.

A mis Amigos y personas especiales, que a pesar de todo siempre han estado aquí, en mis éxitos y mis tropiezos, siempre con una frase de aliento y de confianza para no desistir. Gracias por su sincera amistad.

A las personas que directa e indirectamente me apoyaron para poder estudiar y concluir con la especialidad.

A los profesores de la especialidad, compañeros y amigos que enriquecieron mi conocimiento, me asesoraron y me resolvieron dudas.

Especialmente al Dr. J. Carlos Pedraza por darme la confianza, paciencia y tiempo para trabajar junto con él.

A mí, por el esfuerzo que fue necesario hacer para finalizar este trabajo y que finalmente lo que alcancé no es un documento más, sino una satisfacción que me alimenta el alma y que enriquece mi conocimiento.

Gracias Dios por haberme dado la vida… Esta vida.

Índice del Contenido

Capítulo 1. Antecedentes y Objetivo

1.1.-Reconstrucción en 3d

Las necesidades a las que intenta dar respuesta la digitalización 3D o tridimensional son antiguas: control de calidad industrial, verificación de piezas industriales, diseño de moldes, generación de modelo para simulación por elemento finito, prototipado rápido, etc. Las nuevas técnicas de digitalización 3D nos permiten capturar los puntos de medida con mayor precisión y velocidad, pudiéndose emplear sobre distintos tipos de objetos, de distintas dimensiones, geometrías y texturas. Existen distintos sistemas de digitalización 3D, que principalmente se pueden dividir en dos grandes grupos: sistemas con contacto o sin contacto con el objeto a digitalizar:

1.1.1.-Digitalización por contacto.

Estos sistemas de digitalización 3D son los más antiguos. Principalmente se emplean para la verificación dimensional de piezas industriales (control de calidad). Con estos sistemas se obtienen las coordenadas de los puntos gracias al desplazamiento de una punta sobre la superficie a digitalizar. En la actualidad existen cabezales de digitalización en continuo. La velocidad de adquisición de datos se incrementa notablemente respecto a los cabezales convencionales, ya que éstos no se separan de la superficie a digitalizar. Otro tipo de sistema de digitalización 3D con contacto son los brazos articulados de operación manual. Estos poseen una elevada precisión; pero por el contrario tienen una velocidad de adquisición de datos muy baja, ya que se necesita llevar manualmente la punta a cada punto que se quiera digitalizar, con lo cual la digitalización de un objeto de tamaño medio resulta muy costosa. Para emplear estos sistemas por contacto, se necesita que las piezas tengan la rigidez suficiente para que no se deformen con contacto de la punta y debido a la geometría de las puntas, es imposible digitalizar algunas ranuras y ángulos interiores; pero poseen una elevada resolución.

1.1.2.-Digitalización sin contacto

La principal ventaja de los digitalizadores 3D sin contacto es que tienen una velocidad de adquisición de datos muy superior a la de los digitalizadores con contacto. Podemos dividir las técnicas de digitalización sin contacto en dos grandes grupos:

a) Técnicas de visión pasiva. El sistema visual humano permite obtener información de profundidad mediante la fusión de dos escenas monoculares, que son las escenas que captan cada uno de nuestros ojos. Este sistema nos permite "ver en 3D" y es conocido como principio de visión estereoscópica. Así pues, dicho principio se basa en utilizar dos puntos de vista de un mismo objeto para encontrar las coordenadas tridimensionales. Para determinar la posición de un punto a partir de dos imágenes es necesario tener un modelo del sistema óptico utilizado. Este principio general puede mejorarse con modelos de cámaras más elaborados o utilizando más de dos cámaras, lo que se conoce con el nombre de fotogrametría. La visión estereoscópica presenta el interés de poder dar la posición en 3D de puntos sobre las superficies independientemente de la iluminación específica (visión pasiva). Algunos ejemplos conocidos son: paralelo del movimiento profundidad en base a sombra, estéreo visión, profundidad por foco (Depth From Focus o DFF), o profundidad de foco (Depth From Defocus o DFD).

b) Técnicas de visión activa. Estás técnicas son las que hacen intervenir una fuente de luz específica para determinar las coordenadas tridimensionales de los puntos de medida. Los sistemas ópticos se fundamentan en el cálculo de la profundidad.

Estos sistemas ópticos constan siempre como mínimo de un emisor de luz y un receptor.
Conociendo la dirección del rayo emitido y la del recibido se obtienen las dimensiones del triángulo formado y por lo tanto obtener la profundidad del punto inspeccionado. Existen tres tipos de sistemas ópticos:

I) Digitalización por láser. La fuente de luz en estos sistemas es un láser de diodos. Dicho láser proyecta una línea de luz sobre la superficie que vamos a digitalizar. La luz reflejada será detectada por una o dos células fotosensibles que se encuentran situadas a ambos lados del láser. Estos detectores leen el haz de luz reflejado y transmiten la información obtenida sobre el perfil de la pieza a un software. El resultado de la digitalización nos da la posibilidad de realizar una ingeniería inversa, es decir, obtener la geometría completa de la pieza prescindiendo de los planos.

II) Proyección de luz estructurada. En este sistema el emisor es un proyector de luz blanca y el receptor una cámara CCD. Cuando se inicia una digitalización el proyector lanza sobre el objeto una serie de franjas de luz verticales de claros y sombras alternadas, que son registradas por la cámara. Para obtener las coordenadas de los puntos digitalizados por triangulación, matemáticamente se podía decir que el sistema proyector es un emisor de planos de luz y la cámara CCD un receptor de líneas rectas. El cálculo de la profundidad consiste en resolver la intersección plano-recta.

III) Telemetría. La telemetría consiste en medir el tiempo de recorrido de un rayo luminoso (láser) hasta la superficie de medida. Se puede medir de dos formas (con la medida del tiempo de vuelo o por el cálculo por diferencia de fase). En el primer caso los datos se obtienen midiendo el tiempo entre la emisión del impulso luminoso y la observación del retorno. En el segundo se regula el impulso luminoso siguiendo una frecuencia terminada y se mide el desfase entre el rayo emitido y la luz retornada.

La clasificación de los Métodos de Reconstrucción 3D se puede observar en la figura 1.1.

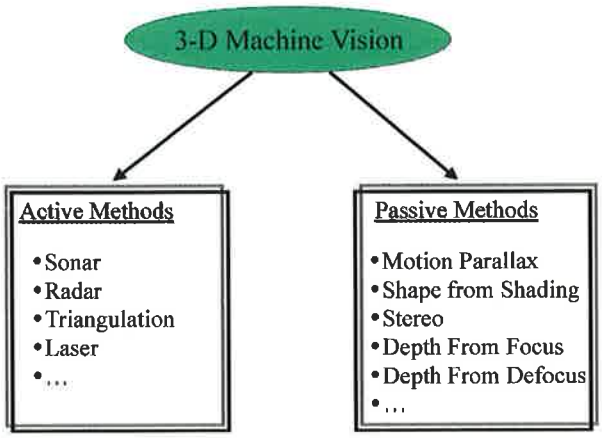


Figura 1.1.- Métodos de Reconstrucción 3D

En este proyecto estamos proponiendo la digitalización utilizando una técnica de proyección de línea láser.

1.2.-Objetivo

Llevar a cabo la instrumentación de un dispositivo que permita la reconstrucción tridimensional de objetos pequeños utilizando la proyección de rayo láser.

Para iniciar el proyecto, primero debemos de saber que tipo de motores existen y cual es el más conveniente en base a las necesidades que tenemos. Por lo que nos introduciremos al tema de los motores.

Capítulo 2. Motores

Esencialmente el objetivo del funcionamiento de un motor es el de convertir ó transformar la energía eléctrica en un movimiento mecánico, como se muestra en la figura 2.1.

Figura 2.1.- Proceso de transformación de la energía en un motor.

Debido a su construcción se pueden clasificar en diferentes tipos, los cuales se presentan a continuación.

2.1.-Motores DC

Los motores eléctricos con frecuencia se emplean como elemento de control final en los sistemas de control por posición o de velocidad. Los motores se puede clasificar en dos categorías principales: Motores de cd y motores de ca. La mayoría de los motores que se emplean en los sistemas de control modernos son motores de cd. Los principios básicos del funcionamiento de un motor son los siguientes:

1. Cuando en un campo magnético, una corriente pasa por un conductor, se ejerce una fuerza sobre el conductor. Para un conductor de longitud L que lleva una corriente I en una campo magnético que tiene una densidad de flujo B y es perpendicular al conductor, la fuerza ejercida F es igual a BIL.
2. Cuando un conductor se desplaza dentro de un campo magnético, sobre él se induce una f.e.m. La f.e.m. inducida, e, es igual a la velocidad con la que cambia el flujo magnético D(el flujo magnético es igual al producto de la densidad de flujo por el área) que cubre el conductor (ley de Faraday), es decir, $e = -dD/dt$. El signo menos indica que la dirección de la f.e.m. es en sentido opuesto al cambio que la

produce (Ley de Lenz); es decir, la dirección de la f.e.m. inducida es tal que produce una corriente que crea campos magnéticos que tienden a neutralizar el cambio en el flujo magnético asociado al devanado que produjo la f.e.m. Por ellos, con frecuencia se le conoce como fuerza contraelectromotriz.

2.1.1.-Motor de cd con devanados de campo

Los motores de cd con devanados de campo se dividen en: motores en serie, en paralelo, compuestos y de excitación independiente, dependiendo de la manera como se encuentran conectados los devanados de campo y los devanados de la armadura.

Motor (con excitación) en serie

En el motor en serie, los devanados de la armadura y de los campos están en serie. Este motor produce el par de rotación de arranque de mayor intensidad y alcanza la mayor velocidad sin carga. Con cargas ligeras existe el riesgo de que el motor alcance velocidades muy altas. La inversión de la polaridad de la alimentación eléctrica de los devanados no tiene efecto en la dirección de rotación del motor éste sigue girando en la misma dirección dado que tanto las corrientes de campo como de armadura quedaron invertidas.

Motor en derivación (en paralelo)

En éste tipo de motor los devanados de armadura y de campo están en paralelo, es decir, genera el par de rotación de menor intensidad, en el arranque tiene una velocidad sin carga mucho menor y permite una buena regulación de la velocidad. Debido a esta velocidad casi constante, independiente de la carga, estos motores se utilizan mucho. Para invertir la dirección de giro, hay que invertir la armadura o el campo. Por ello en este caso es preferible utilizar los devanados de excitación independiente.

Motor de excitación compuesta

Este motor tiene dos devanados de campo, uno en serie con la armadura y otro en paralelo. En estos motores se intenta conjuntar lo mejor del motor (excitado) en serie y del motor en

paralelo, es decir, un par de rotación de inicio de valor elevado y una buena regulación de la velocidad.

Motor de excitación independiente

En este motor el control de las corrientes de armadura y de campo es independiente y se le puede considerar un caso especial del motor en paralelo.

2.1.2.-Control de motores de cd

La velocidad que alcanza un motor de imán permanente depende de la magnitud de la corriente que pasa por el devanado de la armadura. En un motor con devanado de campo, la velocidad se modifica variando la corriente de la armadura, o la de campo; en general, es la primera la que se modifica. Por lo tanto, para controlar la velocidad se puede utilizar el control del voltaje que se aplica a la armadura. Sin embargo, dado que el empleo de fuentes de voltaje de valor fijo es frecuente, el voltaje variable se logra mediante un circuito electrónico.

En una fuente de corriente alterna, se utiliza el circuito de tiristor para controlar el voltaje promedio que se aplica a la armadura. Sin embargo, lo más común es que uno se ocupe de estas señales de control provenientes de microprocesadores. De ahí que lo más común es utilizar la técnica llamada *modulación por ancho de pulso* (PWM, por sus siglas en ingles), la cual utiliza una fuente de voltaje de cd constante y secciona su voltaje para que varíe su valor promedio.

2.1.3.-Motores de cd de imán permanente y sin escobillas

Un problema de los motores de cd es que requieren un colector y escobillas para invertir en forma periódica la corriente que pasa por cada uno de los devanados de la armadura. Las escobillas establecen contacto deslizante con el colector; las chispas que saltan entre ambos van desgastando las escobillas. Por ello, éstas deben ser reemplazadas de manera

periódica y volver a recubrir el colector. Para evitar estos problemas se diseñaron los motores sin escobillas.

En esencia, estos motores constan de una secuencia de devanados de estator y un rotor de imán permanente. Un conductor por el que pasa corriente eléctrica y se encuentra en medio de un campo magnético experimenta una fuerza; así mismo, como consecuencia de la tercera ley de Newton, el imán también experimenta una fuerza opuesta de igual magnitud. En el motor de cd convencional, el imán esta fijo y los conductores por los que pasa la corriente presentan movimientos. En el motor de cd de imán permanente y sin escobillas sucede lo contrario: los conductores por los que pasa corriente están fijos y es el imán el que se mueve. El rotor es un imán permanente de ferrita o cerámica; La corriente que llega a los devanados del estator conmuta en forma electrónica mediante transistores en secuencia alrededor de los devanados; la conmutación se controla con la posición del rotor, de manera que siempre haya fuerzas actuando en el imán y provoquen su rotación en la misma dirección. Los sensores de Hall por lo general se emplean para detectar la posición del rotor e iniciar la conmutación de los transistores; los sensores se colocan alrededor del estator.

Los motores de cd de imán permanente y sin escobillas se utilizan cada vez más cuando a la par se necesita un alto rendimiento, gran confiabilidad y poco mantenimiento. Gracias a que no tienen escobillas, estos motores no producen ruido y permiten alcanzar altas velocidades.

2.2.-Servo Motores

Los servos son un tipo especial de motor de c.c. que se caracterizan por su capacidad para posicionarse de forma inmediata en cualquier posición dentro de su intervalo de operación. Para ello, el servomotor espera un tren de pulsos que se corresponde con el movimiento a realizar. Están generalmente formados por un amplificador, un motor, un sistema reductor formado por ruedas dentadas y un circuito de realimentación, todo en una misma caja de pequeñas dimensiones. El resultado es un servo de posición con un margen de operación de 180° aproximadamente.

Se dice que el servo es un dispositivo con un eje de rendimiento controlado ya que puede ser llevado a posiciones angulares específicas al enviar una señal codificada. Con tal de que exista una señal codificada en la línea de entrada, el servo mantendrá la posición angular del engranaje. Cuando la señala codificada cambia, la posición angular de los piñones cambia. En la práctica, se usan servos para posicionar elementos de control como palancas, pequeños ascensores y timones. También se usan en radio-control, marionetas y, por supuesto, en robots. Los Servos son sumamente útiles en robótica. Los motores son pequeños.

En la figura 2.2 se muestra la composición interna de un servomotor. Se puede observar el motor, el circuito de control, un juego de piñones, y la caja. También se pueden ver los **3 cables de conexión externa**:

- Uno (rojo) es para **alimentación**, Vcc (~ +5volts);
- Otro (negro) para conexión a **tierra** (GND);
- El último (blanco o amarillo) es la línea de **control** por la que se le envía la señal codificada para comunicar el ángulo en el que se debe posicionar.

Figura 2.2.- Composición de un Servomotor

El motor del servo tiene algunos circuitos de control y un potenciómetro conectado al eje central del motor. En la figura superior se puede observar a la derecha. Este potenciómetro permite al circuito de control, supervisar el ángulo actual del servo motor. Si el eje está en el ángulo correcto, entonces el motor está apagado. Si el circuito detecta que el ángulo no

es correcto, el motor volverá a la dirección correcta, hasta llegar al ángulo que es correcto. El eje del servo es capaz de llegar alrededor de los 180 grados. Normalmente, en algunos llega a los 210 grados, pero varía según el fabricante.

Un servo normal se usa para controlar un movimiento angular de entre 0 y 180 grados. Un servo normal no es mecánicamente capaz de retornar a su lugar, si hay un mayor peso que el sugerido por las especificaciones del fabricante.

El voltaje aplicado al motor es proporcional a la distancia que éste necesita viajar. Así, si el eje necesita regresar una distancia grande, el motor regresará a toda velocidad. Si este necesita regresar sólo una pequeña cantidad, el motor girará a menor velocidad. A esto se le denomina **control proporcional**.

El sistema de control de un servo se limita a indicar en que posición se debe situar. Esto se lleva a cabo mediante una serie de pulsos tal que la duración del pulso indica el ángulo de giro del motor. Cada servo tiene sus márgenes de operación, que se corresponden con el ancho del pulso máximo y mínimo que el servo entiende. Los valores más generales se corresponden con pulsos de entre 1 ms y 2 ms de anchura, que dejarían al motor en ambos extremos (0° y 180°). El valor 1.5 ms indicaría la posición central o neutra (90°), mientras que otros valores del pulso lo dejan en posiciones intermedias. Estos valores suelen ser los recomendados, sin embargo, es posible emplear pulsos menores de 1 ms o mayores de 2 ms, pudiéndose conseguir ángulos mayores de 180°. Si se sobrepasan los límites de movimiento del servo, éste comenzará a emitir un zumbido, indicando que se debe cambiar la longitud del pulso. El factor limitante es el tope del potenciómetro y los límites mecánicos constructivos.

El período entre pulso y pulso (tiempo de OFF) no es crítico, e incluso puede ser distinto entre uno y otro pulso. Se suelen emplear valores ~ 20 ms (entre 10 ms y 30 ms). Si el intervalo entre pulso y pulso es inferior al mínimo, puede interferir con la temporización interna del servo, causando un zumbido, y la vibración del eje de salida. Si es mayor que el

máximo, entonces el servo pasará a estado dormido entre pulsos. Esto provoca que se mueva con intervalos pequeños.

Es importante destacar que para que un servo se mantenga en la misma posición durante un cierto tiempo, es necesario enviarle continuamente el pulso correspondiente. De este modo, si existe alguna fuerza que le obligue a abandonar esta posición, intentará resistirse. Si se deja de enviar pulsos (o el intervalo entre pulsos es mayor que el máximo) entonces el servo perderá fuerza y dejará de intentar mantener su posición, de modo que cualquier fuerza externa podría desplazarlo.

El diagrama de tiempos del tren de pulsos necesario para controlar un servomotor se puede observar en las figuras 2.3 y 2.4.

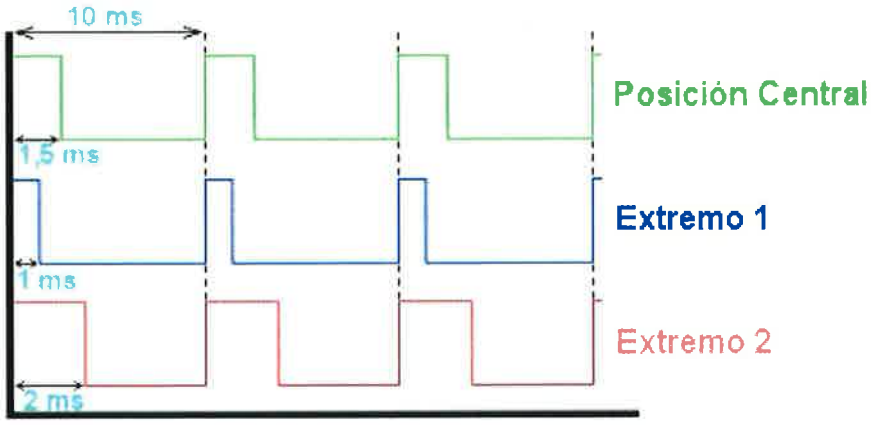


Figura 2.3.- Tren de pulsos para control del Servomotor

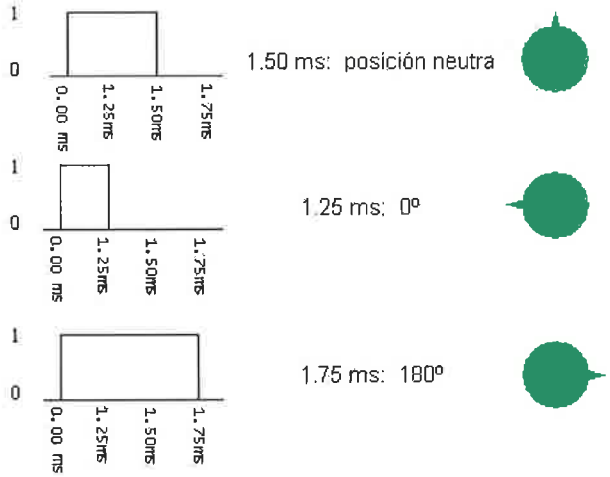


Figura 2.4.- Otra posibilidad de pulsos de Control

2.3.-Motores a Pasos

2.3.1.-Introducción

En numerosas ocasiones es necesario convertir la energía eléctrica en energía mecánica, esto se puede lograr, por ejemplo, usando los motores de corriente continua. Pero cuando lo deseado es posicionamiento con un elevado grado de exactitud y/o una muy buena regulación de la velocidad, se puede contar con una gran solución: utilizar un motor paso a paso.

Los motores paso a paso son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos.

La característica principal de estos motores es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8°, es decir, que se necesitarán 4 pasos en el primer caso (90°) y 200 para el segundo caso (1.8°), para completar un giro completo de 360°.

Estos motores poseen la habilidad de poder quedar enclavados en una posición o bien totalmente libres. Si una o más de sus bobinas están energizadas, el motor estará enclavado en la posición correspondiente y por el contrario quedará completamente libre si no circula corriente por ninguna de sus bobinas.

El motor paso a paso está constituido esencialmente por dos partes: a) Una fija llamada "estator" (ver figura 2.6), construida a base de cavidades en las que van depositadas las bobinas que excitadas convenientemente formarán los polos norte-sur de forma que se cree un campo magnético giratorio. b) Una móvil, llamada "rotor" (ver figura 2.5), construida mediante un imán permanente, con el mismo número de pares de polos, que el contenido en una sección de la bobina del estator; este conjunto va montado sobre un eje soportado por dos cojinetes que le permiten girar libremente.

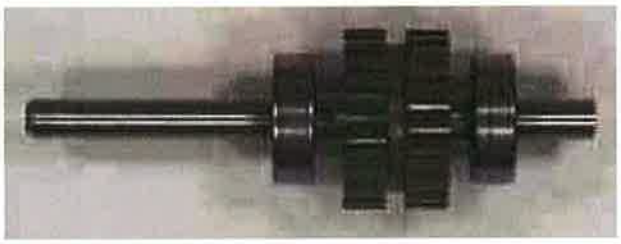


Figura 2.5.- Imagen del Rotor

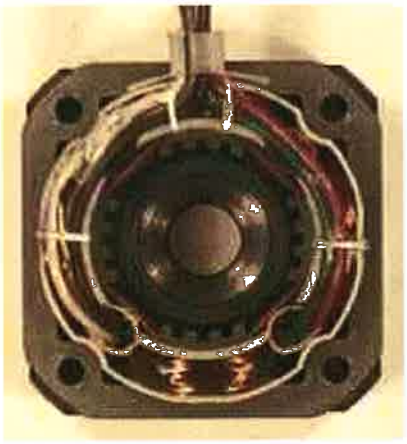


Figura 2.6.- Imagen de un Estator de 4 bobinas

Si por el medio que sea, conseguimos excitar el estator creando los polos N-S, y hacemos variar dicha excitación de modo que el campo magnético formado efectúe un movimiento giratorio, la respuesta del rotor será seguir el movimiento de dicho campo, produciéndose de este modo el giro del motor.

Puede decirse por tanto que un motor paso a paso es un elemento que transforma impulsos eléctricos en movimientos de giro controlados, ya que podremos hacer girar al motor en el sentido que deseemos y el número de vueltas y grados que necesitemos.

2.3.2.-Principio de funcionamiento

Los motores eléctricos, en general, basan su funcionamiento en las fuerzas ejercidas por un campo electromagnético y creadas al hacer circular una corriente eléctrica a través de una o varias bobinas. Si dicha bobina, generalmente circular y denominada estator, se mantiene en una posición mecánica fija y en su interior, bajo la influencia del campo

electromagnético, se coloca otra bobina, llamada rotor, recorrida por una corriente y capaz de girar sobre su eje, esta última tenderá a buscas la posición de equilibrio magnético, es decir, orientará sus polos NORTE-SUR hacia los polos SUR-NORTE del estator, respectivamente. Cuando el rotor alcanza esta posición de equilibrio, el estator cambia la orientación de sus polos, aquel tratará de buscar la nueva posición de equilibrio; manteniendo dicha situación de manera continuada, se conseguirá un movimiento giratorio y continuo del rotor y a la vez la transformación de una energía eléctrica en otra mecánica en forma de movimiento circular.

Aún basado en el mismo fenómeno, el principio de funcionamiento de los motores de corriente continua, los motores paso a paso son más sencillos si cabe, que cualquier otro tipo de motor eléctrico.

La figura 2.7 intenta ilustrar el modo de funcionamiento de un motor paso a paso, suponemos que las bobinas L1 como L2 poseen un núcleo de hierro dulce capaz de imantarse cuando dichas bobinas sean recorridas por una corriente eléctrica. Por otra parte el imán M puede girar libremente sobre el eje de sujeción central.

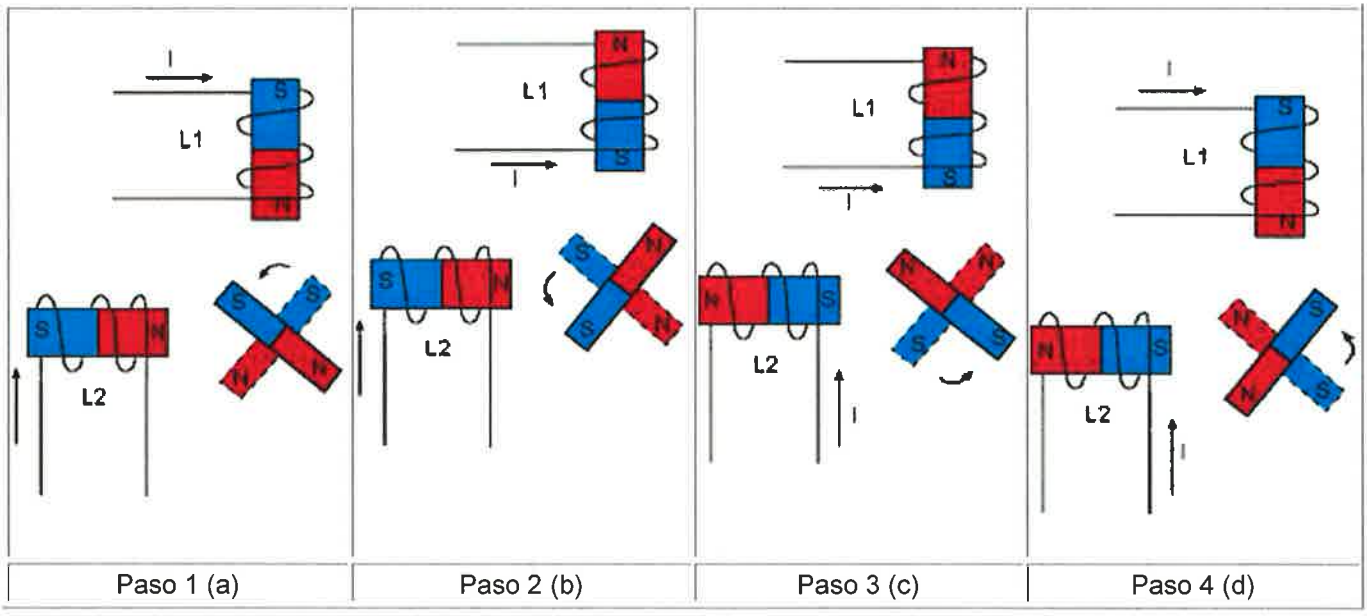


Figura 2.7.- Principio de funcionamiento de un motor paso a paso

Inicialmente, sin aplicar ninguna corriente a las bobinas (que también reciben el nombre de fases) y con M en una posición cualquiera, el imán permanecerá en reposo si no se somete a una fuerza externa.

Si se hace circula corriente por ambas fases como se muestra en la Figura 3.7(a), se crearán dos polos magnéticos NORTE en la parte interna, bajo cuya influencia M se desplazará hasta la posición indicada en dicha figura.

Si invertimos la polaridad de la corriente que circula por L1 se obtendrá la situación magnética indicada en la Figura 3.7(b) y M se verá desplazado hasta la nueva posición de equilibrio, es decir, ha girado 90 grados en sentido contrario a las agujas del reloj.
Invirtiendo ahora la polaridad de la corriente en L2, se llega a la situación de la Figura 3.7 (c) habiendo girado M otros 90 grados. Si, por fin, invertimos de nuevo el sentido de la corriente en L1, M girará otros 90 grados y se habrá obtenido una revolución completa de dicho imán en cuatro pasos de 90 grados.

Por tanto, si se mantiene la secuencia de excitación expuesta para L1 y L2 y dichas corrientes son aplicadas en forma de pulsos, el rotor avanzará pasos de 90 grados por cada pulso aplicado.
Por lo tanto se puede decir que un motor paso a paso es un dispositivo electromecánico que convierte impulsos eléctricos en un movimiento rotacional constante y finito dependiendo de las características propias del motor.
El modelo de motor paso a paso que hemos analizado, recibe el nombre de bipolar ya que, para obtener la secuencia completa, se requiere disponer de corrientes de dos polaridades, presentando tal circunstancia un inconveniente importante a la hora de diseñar el circuito que controle el motor. Una forma de solventar este inconveniente es la representada en la Figura 3.8, obteniéndose un motor unipolar de cuatro fases, puesto que la corriente circula por las bobinas en un único sentido.

Si inicialmente se aplica la corriente a L1 y L2 cerrando los interruptores S1 y S2, se generarán dos polos NORTE que atraerán al polo SUR de M hasta encontrar la posición de

equilibrio entre ambos como puede verse en la Figura 2.8(a). Si se abre posteriormente S1 y se cierra S3, por la nueva distribución de polos magnéticos, M evoluciona hasta la situación representada en la Figura 2.8(b).

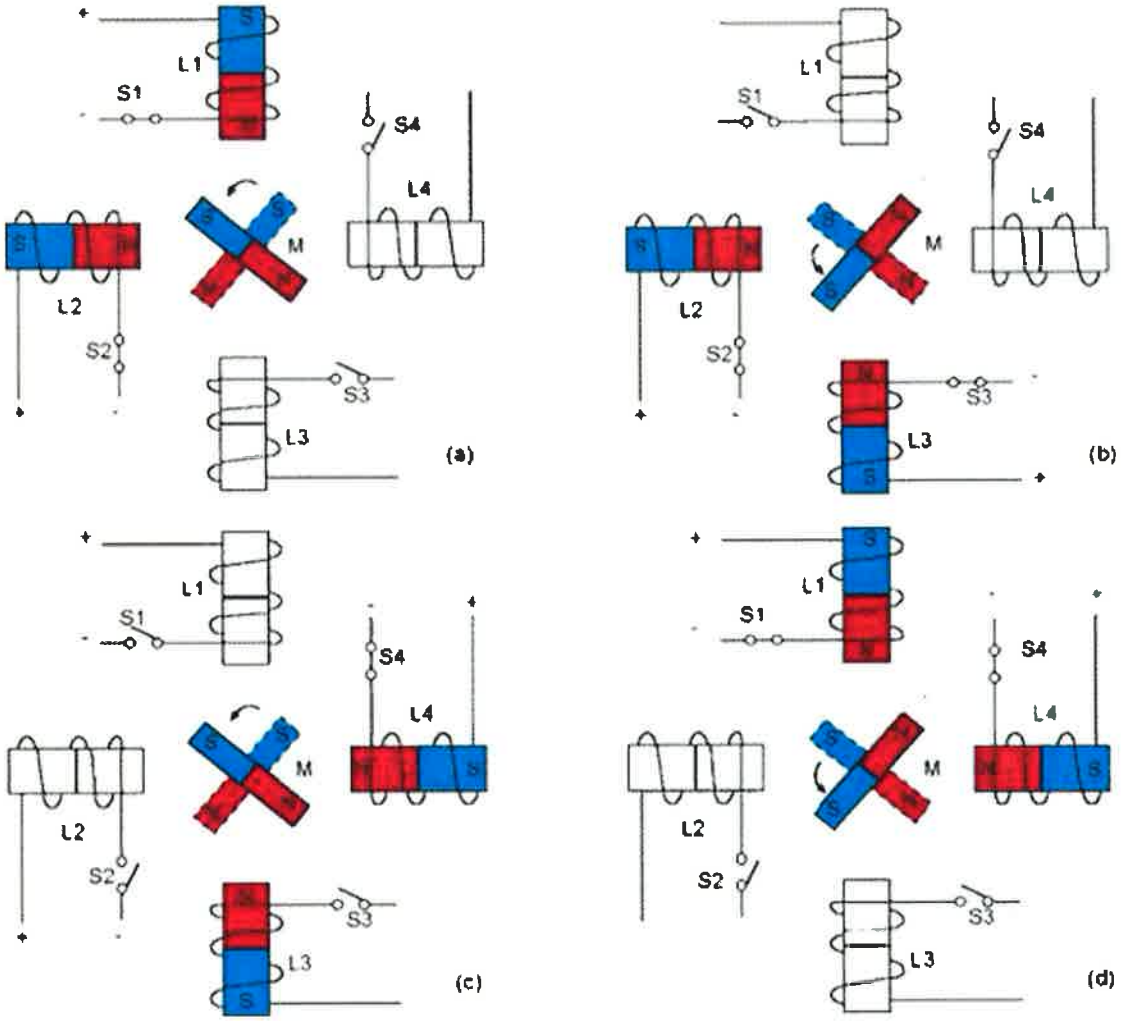


Figura 2.8.- Principio básico de un motor unipolar de cuatro fases

Siguiendo la secuencia representada en la Figuras 2.8(c) y 2.8(d), de la misma forma se obtienen avances del rotor de 90 grados habiendo conseguido, como en el motor bipolar de dos fases, hacer que el rotor avance pasos de 90 grados por la acción de impulsos eléctricos de excitación de cada una de las bobinas. En uno y otro caso, el movimiento obtenido ha sido en sentido contrario al de las agujas del reloj; ahora bien, si las secuencias de excitación se generan en orden inverso, el rotor girará en sentido contrario, por lo que

fácilmente podemos deducir que el sentido de giro en los motores paso a paso es reversible en función de la secuencia de excitación y, por tanto, se puede hacer avanzar o retroceder al motor un número determinado de pasos según las necesidades.

El modelo de motor paso a paso estudiado, salvo su valor didáctico, no ofrece mayor atractivo desde el punto de vista práctico, precisamente por la amplitud de sus avances angulares.

Una forma de conseguir motores Paso a Paso de paso mas reducido, es la de aumentar el número de bobinas del estator, pero ello llevaría a un aumento del coste y del volumen y a pérdidas muy considerable en el rendimiento del motor, por lo que esta situación no es viable. Hasta ahora y para conseguir la solución más idónea, se recurre a la mecanización de los núcleos de las bobinas y el rotor en forma de hendiduras o dientes, creándose así micropolos magnéticos, tantos como dientes y estableciendo las situaciones de equilibrio magnéticos con avances angulares mucho menores, siendo posible conseguir motores de hasta de 500 pasos.

2.3.3.-Parámetros de los motores paso a paso

Desde el punto de vista mecánico y eléctrico, es conveniente conocer el significado de algunas de las principales características y parámetros que se definen sobre un motor paso a paso:

- **Par dinámico de trabajo (*Working Torque*):** Depende de sus características dinámicas y es el momento máximo que el motor es capaz de desarrollar sin perder paso, es decir, sin dejar de responder a algún impulso de excitación del estator y dependiendo, evidentemente, de la carga.

  Generalmente se ofrecen, por parte del fabrican, curvas denominadas de arranque sin error (pull-in) y que relaciona el par en función el número de pasos.

  Hay que tener en cuenta que, cuando la velocidad de giro del motor aumenta, se produce un aumento de la f.c.e.m. en él generada y, por tanto, una disminución de la

corriente absorbida por los bobinados del estator, como consecuencia de todo ello, disminuye el par motor.

- **Par de mantenimiento (*Holding Torque*):** Es el par requerido para desviar, en régimen de excitación, un paso el rotor cuando la posición anterior es estable ; es mayor que el par dinámico y actúa como freno para mantener el rotor en una posición estable dada
- **Para de detención (*Detention Torque*):** Es una par de freno que siendo propio de los motores de imán permanente, es debida a la acción del rotor cuando los devanados del estator están desactivados.
- **Angulo de paso (*Step angle*):** Se define como el avance angular que se produce en el motor por cada impulso de excitación. Se mide en grados, siendo los pasos estándar más importantes los siguientes:

Grados por impulso de excitación	N° de pasos por vuelta
0,72°	500
1,8°	200
3,75°	96
7,5°	48
15°	24

- **Número de pasos por vuelta:** Es la cantidad de pasos que ha de efectuar el rotor para realizar una revolución completa; evidentemente es $NP = \frac{360}{\alpha}$

Donde **NP** es el número de pasos y **α** el ángulo de paso.

- **Frecuencia de paso máximo (*Maximum pull-in/out*):** Se define como el máximo número de pasos por segundo que puede recibir el motor funcionando adecuadamente.

- **Momento de inercia del rotor:** Es su momento de inercia asociado que se expresa en gramos por centímetro cuadrado.
- **Par de mantenimiento, de detención y dinámico:** Definidos anteriormente y expresados en miliNewton por metro.

3.3.4.-Control de los motores paso a paso

Para realizar el control de los motores paso a paso, es necesario generar una secuencia determinada de impulsos. Además es necesario que estos impulsos sean capaces de entregar la corriente necesaria para que las bobinas del motor se exciten, por lo general, el diagrama de bloques de un sistema con motores paso a paso es el que se muestra en la Figura 3.9.

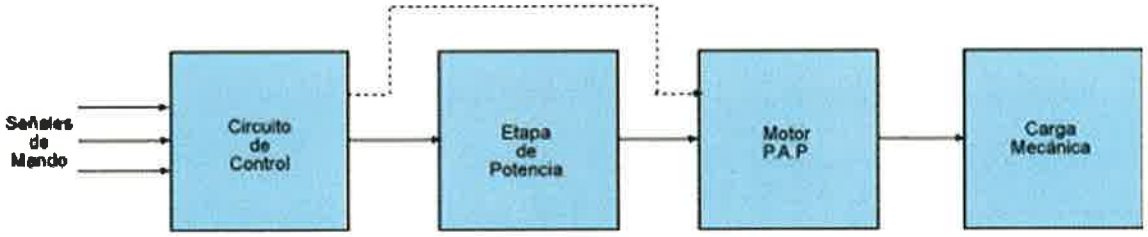


Figura 2.9.- Diagrama de bloques de un sistema con motor paso a paso

3.3.5.-Secuencia del circuito de control

Existen dos formas básicas de hacer funcional los motores paso a paso atendiendo al avance del rotor bajo cada impulso de excitación:

- Paso completo (*full step*): El rotor avanza un paso completo por cada pulso de excitación y para ello su secuencia ha de ser la correspondiente a la expuesta anteriormente, para un motor como el de la Figura 2, y que es presentada de forma resumida en la Tabla 1 para ambos sentidos de giro, las X indican los interruptores que deben estar cerrados (interruptores en ON), mientras que la ausencia de X indica interruptor abierto (interruptores en OFF).

Paso	S1	S2	S3	S4

Paso	S1	S2	S3	S4

1	X			X
2			X	X
3		X	X	
4	X	X		
1	X			X
Sentido horario (a)				

1	X	X		
2		X	X	
3			X	X
4	X			X
1	X	X		
Sentido antihorario (b)				

Tabla 1.- Secuencia de excitación de un motor paso a paso completo

- **Medio paso (*Half step*)**: Con este modo de funcionamiento el rotor avanza medio paso por cada pulso de excitación, presentando como principal ventaja una mayor resolución de paso, ya que disminuye el avance angular (la mitad que en el modo de paso completo). Para conseguir tal cometido, el modo de excitación consiste en hacerlo alternativamente sobre dos bobinas y sobre una sola de ellas, según se muestra en la Tabla 2 para ambos sentidos de giro.

Paso	Excitación de Bobinas			
	S1	S2	S3	S4
1	X			X
2				X
3			X	X
4			X	
5		X	X	
6		X		

Paso	Excitación de Bobinas			
	S1	S2	S3	S4
1	X	X		
2		X		
3		X	X	
4			X	
5			X	X
6				X

7	X	X			7	X		X
8	X				8	X		
1	X		X		1	X	X	
Sentido horario (a)					Sentido antihorario (b)			

Tabla 2.- Secuencia de excitación de un motor Paso a Paso en medio paso

Según la tabla 2 al excitar dos bobinas consecutivas del estator simultáneamente, el rotor se alinea con la bisectriz de ambos campos magnéticos; cuando desaparece la excitación de una de ellas, extinguiéndose el campo magnético inducido por dicha bobina, el rotor queda bajo la acción del único campo existente, dando lugar a un desplazamiento mitad.

2.3.6.-Tipos de motores paso a paso

Hay dos tipos básicos de motores Paso a Paso, los BIPOLARES que se componen de dos bobinas y los UNIPOLARES que tienen cuatro bobinas. Externamente se diferencian entre sí por el número de cables. Los bipolares solo tienen cuatro conexiones dos para cada bobina y los unipolares que normalmente presentan seis cables, dos para cada bobina y otro para alimentación de cada par de éstas, aunque en algunos casos podemos encontrar motores unipolares con cinco cables, básicamente es lo mismo, solo que el cable de alimentación es común para los dos pares de bobinas.

2.3.7.-Motores a pasos Unipolares

En este tipo de motores, todas las bobinas del estator están conectadas en serie formando cuatro grupos. Esta a su vez, se conectan dos a dos, también en serie, y se montan sobre dos estatores diferentes, tal y como se aprecia en la siguiente Figura 2.10. Según puede apreciarse en dicha figura, del motor paso a paso salen dos grupos de tres cables, uno de los cuales es común a dos bobinados. Los seis terminales que parten del motor, deben ser conectados al circuito de control, el cual, se comporta como cuatro conmutadores

electrónicos que, al ser activados o desactivados, producen la alimentación de los cuatro grupos de bobinas con que está formado el estator. Si generamos una secuencia adecuada de funcionamiento de estos interruptores, se pueden producir saltos de un paso en el número y sentido que se desee.

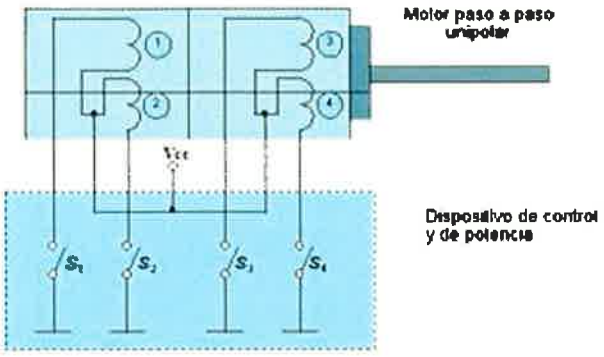


Figura 2.10.- Control de motor Unipolar

Los motores paso a paso unipolares se componen de 4 bobinas como se puede observar en la figura 2.11.

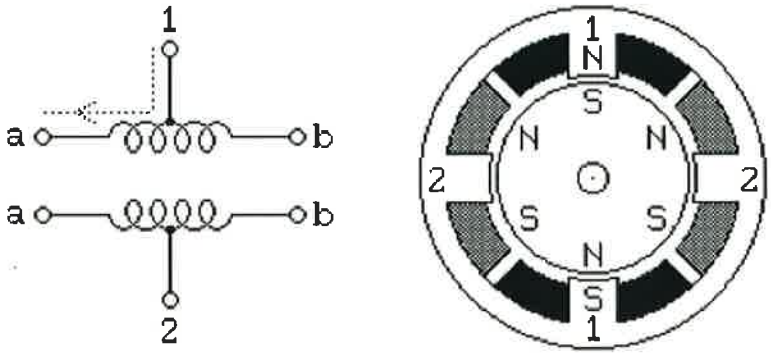


Figura 2.11.- Composición de un motor a pasos unipolar

2.3.8.-Motores a pasos Bipolares

En este tipo de motores las bobinas del estator se conectan en serie formando solamente dos grupos, que se montan sobre dos estatores, tal y como se muestra en la Figura 2.12.

Según se observa en el esquema de este motor salen cuatro hilos que se conectan, al circuito de control, que realiza la función de cuatro interruptores electrónicos dobles, que nos permiten variar la polaridad de la alimentación de las bobinas. Con la activación y desactivación adecuada de dichos interruptores dobles, podemos obtener las secuencias adecuadas para que el motor pueda girar en un sentido o en otro.

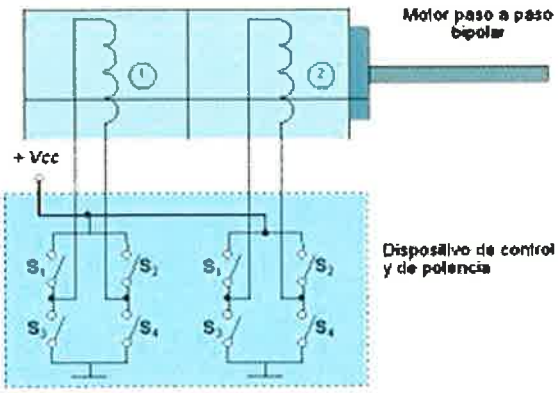
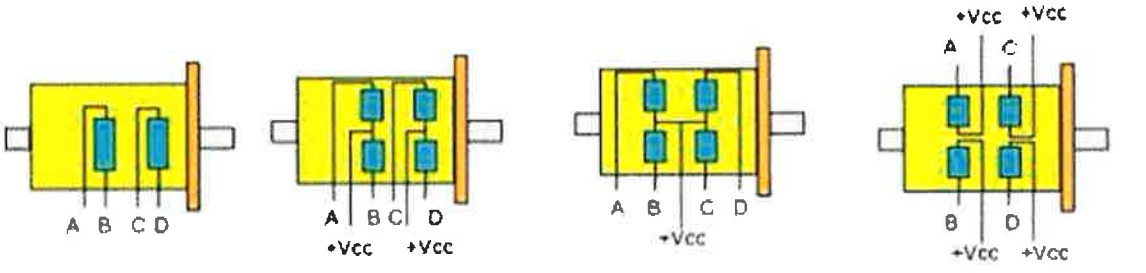


Figura 2.12.- Control de motor Bipolar

La existencia de varios bobinados en el estator de los motores de imán permanente, da lugar a varias formas de agrupar dichos bobinados, para que sean alimentados adecuadamente. Estas formas de conexión permiten clasificar los motores paso a paso en dos grandes grupos:



Como se puede observar en la figura 2.13, los motores paso a paso bipolares se componen de 2 bobinas.

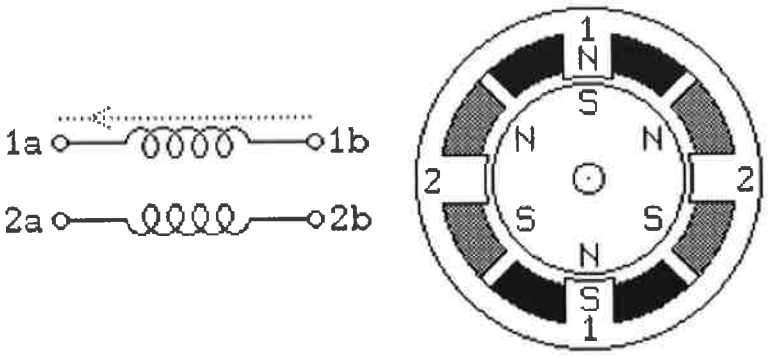
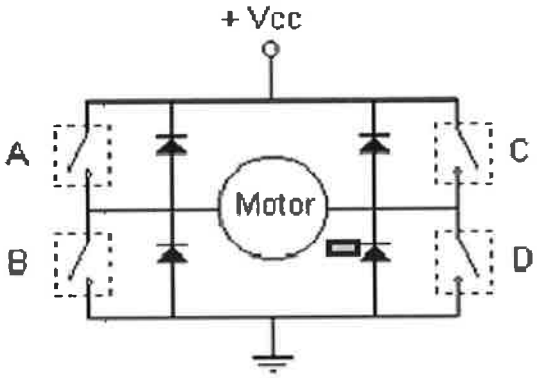


Figura 2.13.- Composición de un motor a pasos Bipolar

2.4.-Control de motores Bipolares

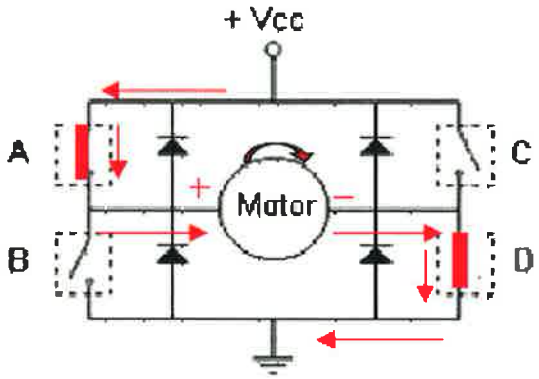
2.4.1.-Puente H

Un puente H es básicamente un arreglo de cuatro interruptores acomodados de la siguiente manera:

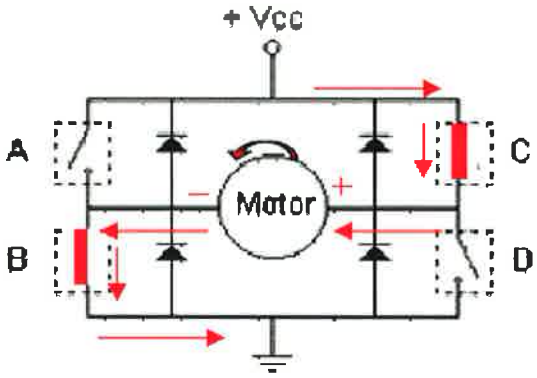


Estos interruptores (A, B, C y D) pueden ser de transistores bipolares, mosfets, jfets, relés o de cualquier combinación de elementos. El objetivo central es el de poder controlar el sentido de un motor de corriente continua sin la necesidad de aplicar voltaje negativo.

Si se cierran solamente los contactos A y D la corriente circulará en un sentido a través del motor o del elemento conectado en la parte central.



Y si se cierran solamente los contactos B y C la corriente circulará en sentido contrario.



Hay que observar también que un puente H necesita de cuatro diodos de protección para el motor. Un puente H tiene por lo general cuatro estados de operación:

Interruptores		Salida		Función
AD	CD	AB	CD	
0	0	0	0	Motor en libertad de acción
1	0	1	0	Motor gira en un sentido
0	1	0	1	Motor gira en el otro sentido
1	1	1	1	Motor se bloqueará y frenará

Donde un 0 corresponde a un interruptor abierto o una salida sin alimentación y un 1 corresponde a un interruptor cerrado o una salida con alimentación.

Por otro lado muchas veces es necesario controlar la velocidad de un motor de continua, una manera de hacerlo es variando es el voltaje aplicado mediante modulación por ancho de pulso o PWM. Este método consiste en aplicar un tren de pulsos cuadrados con un período fijo, en el cual se va variando el ancho del pulso. Este método también se puede entender como apagar y encender en motor a una tasa muy rápida de manera de lograr una velocidad menor. El período de la señal no debe ser muy largo ya que se quiere que este tenga un movimiento continuo y no avance a "tirones".

El transistor como interruptor

El transistor consta de una base un colector y un emisor, y se distribuye de la siguiente manera dependiendo si es PNP o un NPN, esto lo muestra la figura 2.14.

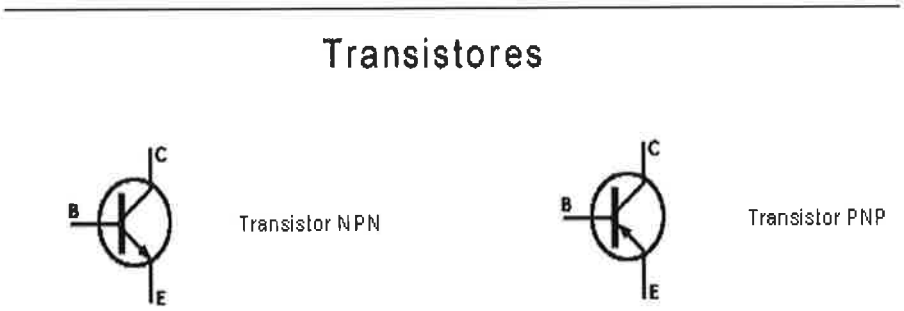


Figura 2.14.- Estructura de un transistor

Lo relevante de estos es que cuando ingresamos una corriente por la base, esta puede alterar de cierta forma la corriente que pasa entre el colector y el emisor o entre el emisor y el colector (por ejemplo amplificarla), lo que mas nos incumbe a nosotros es que cuando la

corriente o en su defecto el voltaje supera cierto nivel el transistor se satura, y esto produce que la corriente pase del colector al emisor o viceversa sin alteración alguna.

A continuación señalaremos un tipo de puente H que es muy efectivo, ya que se puede ocupar directamente un microcontrolador y tiene componentes que soportan altas corrientes. La lista de materiales que requiere el puente H de la figura 2.15 se encuentra en la sección de anexos. (ver anexo 3)

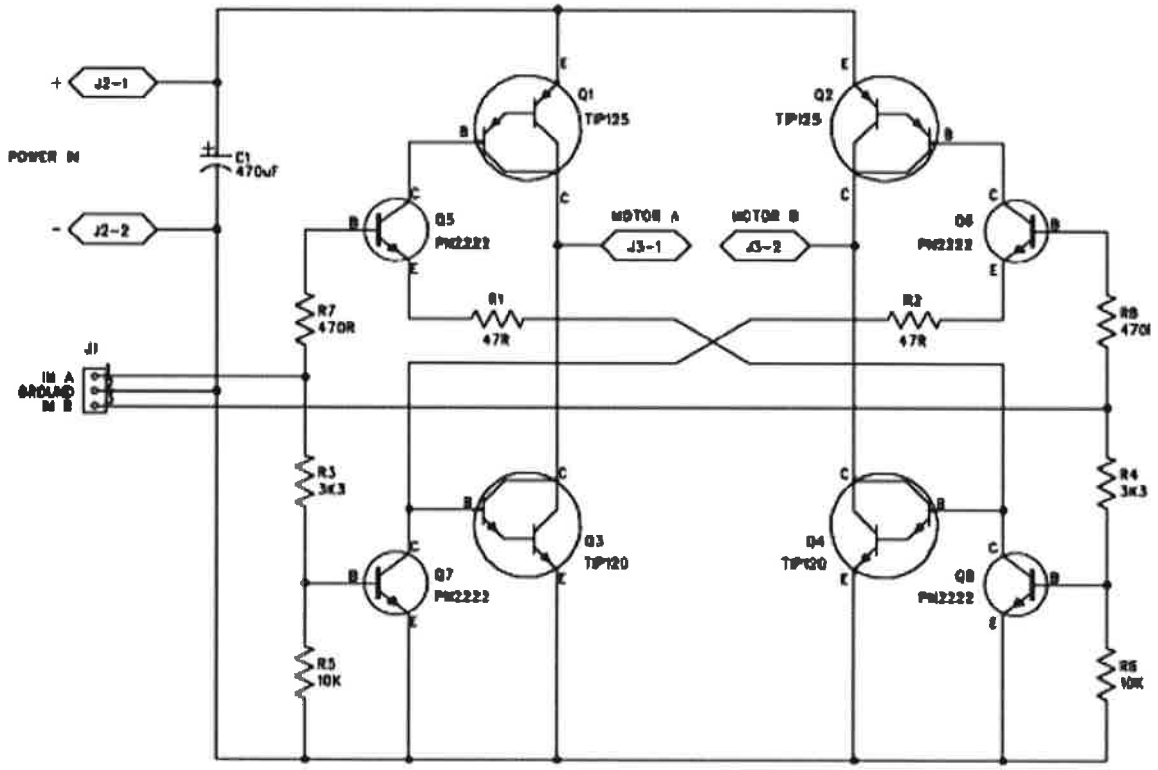
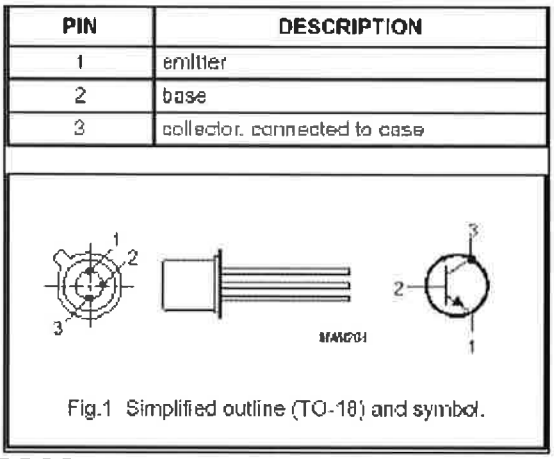


Figura 2.15.- Esquema de un circuito Puente H

Este esquema tiene como ventajas de que las señales de control requieren muy, pero muy poca corriente por lo que puede ser controlado desde cualquier circuito integrado o microcontrolador directamente. Además los transistores TIP125 y TIP120 (o TIP127 y TIP122) traen incorporados los diodos de protección. Si se van a cambiar por ejemplo por los transistores TIP31C y TIP32C se deben agregar los diodos. Aunque esto es posible no

es recomendable ya que estos últimos poseen una ganancia muy baja en comparación con los utilizados.

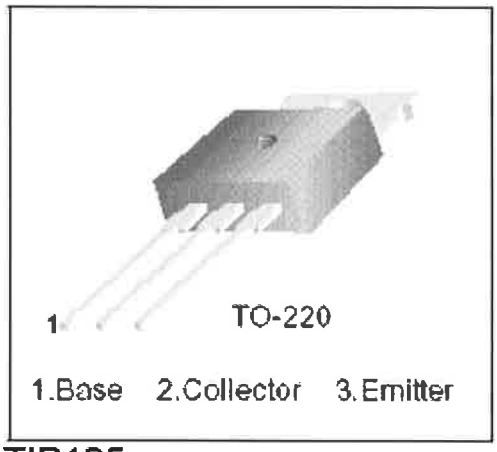
Aunque este esquemáticos puede parecerles un tanto confuso es muy fácil de implementar en el protoboard, donde ven motor A y motor B es donde va ubicado el motor, tienen que fijarse muy bien en los transistores y en como están ordenados la base, el emisor y el colector, aquí se muestran las imágenes del 2n2222 y de el tip125(los otros son iguales al TIP125).



PIN	DESCRIPTION
1	emitter
2	base
3	collector, connected to case

Fig.1 Simplified outline (TO-18) and symbol.

2n2222



TIP125

2.4.2.-Identificación de secuencia para motores bipolares

Para el caso de motores paso a paso bipolares (generalmente de 4 cables de salida), la identificación es sencilla. Simplemente tomando un tester en modo ohmetro (para medir resistencias), podemos hallar los pares de cables que corresponden a cada bobina, debido a que entre ellos deberá haber continuidad (en realidad una resistencia muy baja). Luego solo deberemos averiguar la polaridad de la misma, la cual se obtiene fácilmente probando. Es decir, si conectado de una manera no funciona, simplemente damos vuelta los cables de una de las bobinas y entonces ya debería funcionar correctamente. Si el sentido de giro es inverso a lo esperado, simplemente se deben invertir las conexiones de ambas bobinas y el H-Bridge.

En este momento ya conocemos las características de los motores y la forma de controlar cada uno de ellos. En base a ello, nuestra selección es utilizar motores a pasos y controlarlo mediante un circuito puente H.

Ahora nos toca revisar las herramientas de programación que nos pueden ayudar a diseñar una interfaz de usuario capaz de controlar dispositivos externos y poder hacer procesamiento de imágenes para poder llevar a cabo la reconstrucción en 3D.

Capítulo 3. Herramientas De Programación

3.1.-Labview

3.1.1.-¿Qué es LabVIEW?

LabVIEW es un revolucionario ambiente de desarrollo gráfico con funciones integradas para realizar adquisición de datos, control de instrumentos, análisis de mediciones y presentaciones de datos. LabVIEW le da la flexibilidad de un poderoso ambiente de programación sin la complejidad de los ambientes tradicionales.

A diferencia de los lenguajes de propósito general, LabVIEW provee funcionalidad específica para que pueda acelerar el desarrollo de aplicaciones de medición, control y automatización.

LabVIEW le entrega herramientas poderosas para crear aplicaciones sin líneas de texto de código. Con LabVIEW usted jala y coloca objetos ya construidos para rápidamente crear interfases de usuario. Después usted especifica la funcionalidad del sistema armando diagramas de bloques.

LabVIEW se puede conectar de manera transparente con virtualmente todo tipo de hardware incluyendo instrumentos de escritorio, tarjetas insertables, controladores de movimiento y controladores lógicos programables (PLCs).

Con LabVIEW usted se puede conectar con otras aplicaciones y compartir datos a través de ActiveX, la Web, DLLs, librerías compartidas, SQL, TCP/IP, XML, OPC y otros.

En muchas aplicaciones, la velocidad de ejecución es vital. Con un compilador incluido que genera código optimizado, sus aplicaciones en LabVIEW entregan velocidades de ejecución comparables con programas C compilados. Con LabVIEW puede desarrollar

sistemas que cumplan con sus requerimientos de desempeño a través de las plataformas incluyendo Windows, Macintosh, UNIX o sistemas de tiempo real.

3.1.2.-Beneficios

LabVIEW está altamente integrado con el hardware de medida, con lo que se puede configurar y usar rápidamente cualquier dispositivo de medida que se tenga. Con LabVIEW puede conectarse a miles de instrumentos de medida para construir sistemas de medida completos, incluyendo desde cualquier tipo de instrumento autónomo hasta dispositivos de adquisición de datos, controladores de movimiento y sistemas de adquisición de imagen. Además LabVIEW trabaja con más de 1000 librerías de instrumentos de cientos de fabricantes, y muchos fabricantes de dispositivos de medida incluyen también herramientas de LabVIEW con sus productos.

Con LabVIEW, puede construir sistemas definidos por el usuario mucho más rápidamente que con métodos tradicionales. Ya que las necesidades de las aplicaciones cambian, los sistemas definidos por el usuario de LabVIEW tienen la flexibilidad necesaria para poder modificarlos sin la necesidad de incorporar equipos nuevos. Utilizando un sistema basado en LabVIEW, tiene acceso a sistemas de instrumentación completos con costos mucho más bajos que un único instrumento comercial. National Instruments también asegura que los programas que desarrolla hoy pueden migrar para aprovechar las tecnologías del futuro.

LabVIEW está optimizado para el desarrollo de las aplicaciones de medida y automatización más exigentes. Debido a que la instrumentación virtual está basada en la tecnología informática estándar, usted puede disfrutar de un aumento exponencial en el rendimiento con un costo mucho más bajo que el de un nuevo instrumento de medida dedicado.

Además, LabVIEW se caracteriza por su compilador gráfico optimizado en multihilo para maximizar el rendimiento del sistema. Con LabVIEW puede desarrollar sistemas con el rendimiento necesario para las aplicaciones más exigentes.

Tanto en laboratorio como en producción, los sistemas de medida más rápidos significan un aumento de la producción. Con el poder de LabVIEW se pueden reducir los costos de cualquier prueba o llevar a cabo más experimentos de forma más rápida.

LabVIEW tiene extensas capacidades de adquisición, análisis y presentación disponibles en un sólo paquete, de tal forma que se puede crear una solución completa de manera única en la plataforma que ha elegido. Con LabVIEW puede publicar sus aplicaciones de datos en la Web muy fácilmente o conectarse a otras aplicaciones a través de una variedad de tecnologías estándar, como TCP/IP, DLLs y ActiveX.

LabVIEW simplifica el desarrollo de sistemas y produce un código reutilizable que se ejecuta a velocidades de código compilado. LabVIEW puede también crear ejecutables autónomos o librerías compartidas y DLLs para que pueda llamarlos desde otros entornos como Microsoft Visual Basic o Measurement Studio de National Instruments.

3.1.3.-¿Quiénes lo Usan?

Ingenieros, científicos y técnicos de todo el mundo utilizan LabVIEW para desarrollar soluciones que respondan a sus exigentes aplicaciones. LabVIEW es un revolucionario entorno gráfico de desarrollo para adquisición de datos, control de instrumentos, análisis de medidas y presentación de datos. LabVIEW le da la flexibilidad de un potente lenguaje de programación sin la complejidad típicamente asociada a estos lenguajes.

LabVIEW se ha convertido en una herramienta de desarrollo estándar de la industria para aplicaciones de prueba. LabVIEW combinado con el entorno ejecutor de pruebas, TestStand de National Instruments y la librería de controladores de instrumentos más amplia de la industria proporciona una plataforma de pruebas consistente e integrada para un sistema completo.

Puede utilizar LabVIEW para analizar y registrar resultados reales para aplicaciones en sectores como el automotriz, investigación de energía y muchos otros. Para las aplicaciones que requieren sonido y vibración, procesado de imagen, análisis de tiempo y frecuencia conjunta, wavelets y diseño de filtros digitales, LabVIEW ofrece software extra especialmente diseñado para aumentar la velocidad de desarrollo del sistema.

Puede utilizar LabVIEW para numerosas aplicaciones de control de procesos y automatización. Con LabVIEW puede realizar medidas y control de alta velocidad y con muchos canales. Para aplicaciones de automatización industrial complejas y a gran escala hemos diseñado el módulo de Datalogging and Supervisory Control, con el que se puede monitorear gran número de puntos de E/S, comunicarse con controladores industriales y redes y proporcionar control basado en PC.

LabVIEW es ideal para el monitoreo de maquinaria y para aplicaciones de mantenimiento predictivo que necesitan controles determinísticos, análisis de vibraciones, análisis de visión e imagen o control de movimiento. Con la familia de productos LabVIEW, incluyendo LabVIEW de Tiempo Real para control determinístico y confiable, se pueden crear potentes aplicaciones de monitoreo y control de maquinaria de manera rápida y precisa.

LabView es una herramienta de programación gráfica y modular utilizada para crear instrumentación virtual. Los programas creados mediante LabView son los "instrumentos virtuales" o VI (virtual instrument) que tienen un panel frontal y un diagrama de bloques (figura 1). La interfase de usuario del instrumento se realiza en la ventana del panel frontal y la funcionalidad del instrumento se programa en la ventana de diagrama de bloques. A través del panel frontal el usuario interactúa con el instrumento virtual mediante interruptores, controles deslizantes, gráficos y otros tipos de controles e indicadores proporcionados por LabView. Utiliza un lenguaje de programación gráfico, el lenguaje G, basado en bloques funcionales que se transfieren datos de distintos tipos. Los bloques se seleccionan de un menú tipo paleta, con funciones que van desde las aritméticas a funciones avanzadas de adquisición, control y rutinas de análisis. Incluye también

herramientas de depuración, de ayuda, ejecución resaltada, paso a paso, probetas y puntos de ruptura para trazar y supervisar la ejecución del flujo de datos. LabView permite la creación de aplicaciones profesionales con un mínimo de programación.

Las tarjetas de adquisición de datos suelen traer, además de los drivers, instrumentos virtuales para LabView que permiten hacer de interfase con el hardware.

3.1.4.-Panel frontal

Los elementos del panel frontal son los controles e indicadores. Los controles son las entradas del instrumento virtual, sobre las que actúa el usuario mediante el ratón, y los indicadores son las salidas del instrumento que permiten al usuario conocer los resultados en la pantalla del ordenador. En la figura 3.1 se puede observar tanto el panel frontal como el diagrama de bloques de un VI.

Los controles e indicadores se insertan en el panel frontal seleccionándolos desde la paleta de controles (Figura 3.2(a))

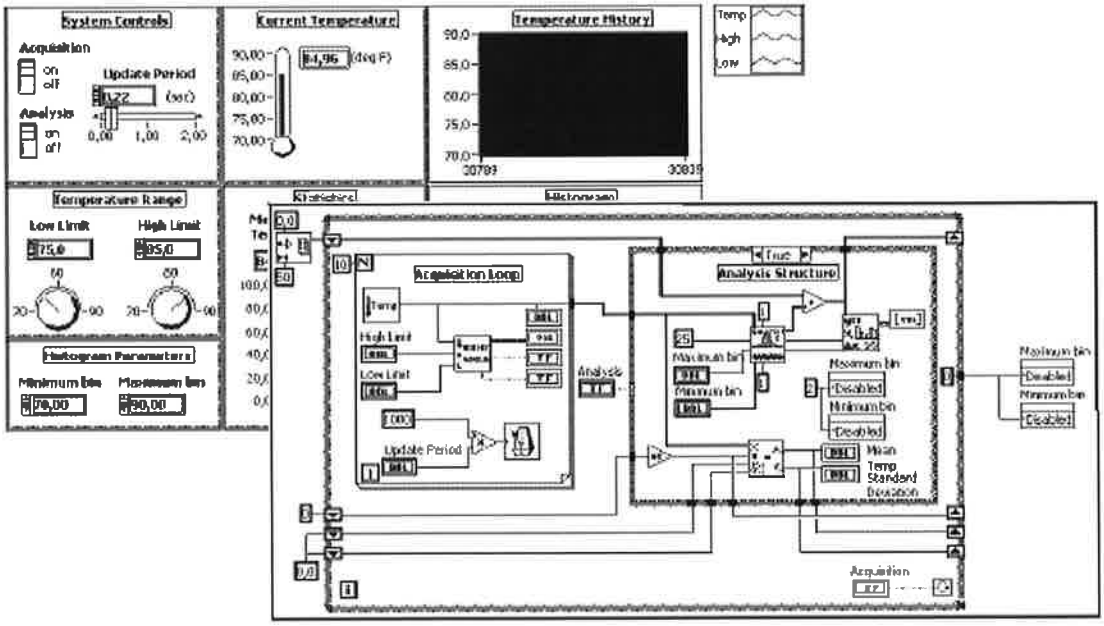


Figura 3.1- Panel frontal y diagrama de bloques de un instrumento virtual en LabView

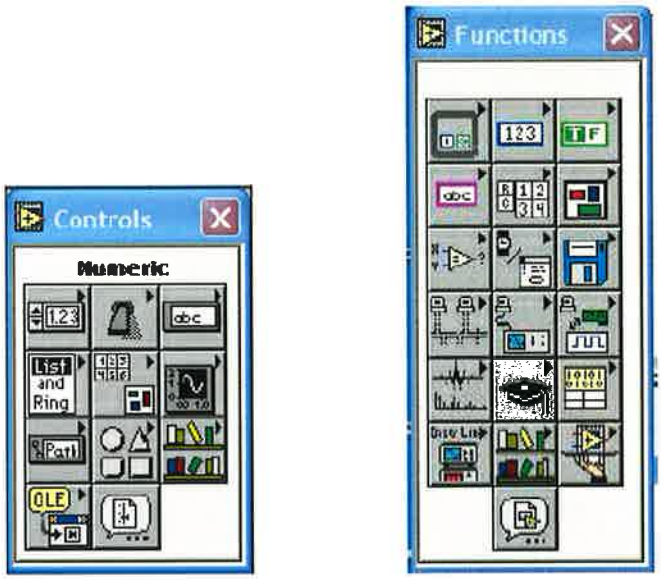


Figura 3.2.- Paleta de controles del panel frontal (a) y paleta de funciones del diagrama de bloques (b)

Nota: Si la paleta de controles no está visible se puede mostrar con Windows_Show Controls Palette. También se puede mostrar pulsando el botón derecho del ratón en una zona libre del panel frontal.

3.1.5.-Diagrama de bloques

El funcionamiento del instrumento virtual se programa en el diagrama de bloques con el lenguaje gráfico G. En el diagrama de bloques tendremos terminales y nodos conectados mediante cables.

Los terminales son en general puntos donde se puede conectar un cable. Dos tipos de bloques sólo pueden tener una terminal: Terminales correspondientes a los controles e indicadores del panel frontal. El usuario introduce valores en los controles y tras la ejecución del VI los resultados se muestran en los indicadores.

Constantes. Constantes creadas por el usuario o constantes predefinidas accesibles en la paleta de funciones.

Los nodos son los elementos que realizan la ejecución en el diagrama de bloques, pueden tener más de una terminal:

- Funciones

- SubVIs
- Estructuras
- Nodos de Fórmula
- Nodos de Interfase de Código
- Nodos de atributo

Los tres últimos son conceptos avanzados que no se van a explicar en este manual.

Los tres primeros se explican posteriormente.

Estos bloques se cablean virtualmente unos con otros para transferirse los datos. La ejecución del VI está dirigida por los datos, a medida que estos están disponibles los bloques funcionales realizan operaciones con los datos y proporcionan resultados para otros bloques.

3.1.6.-Tableta de herramientas

Esta tableta está disponible tanto en el panel frontal como en el diagrama de bloques, las distintas herramientas que se indican en la figura 4.3 son las siguientes:

Figura 4.3.- Paleta de herramientas

Operación: Cambia el valor de un control o selecciona el texto de un control.

Posicionado/tamaño/selección: Selecciona objetos y permite cambiar su posición o tamaño.

Edición de texto: Para editar texto o crear etiquetas

Cableado: Para conectar terminales y nodos en el diagrama de bloques.

Menú contextual: Se obtiene un menú personalizado para cada objeto. Equivalente a botón derecho.

Desplazamiento: Desplaza la ventana como con las barras de desplazamiento.

Puntos de ruptura: Pone puntos de ruptura en VIs, funciones y estructuras.

Probeta de datos: Puntos de prueba en cables.

Toma de color: Toma el color de un objeto para pegarlo con la herramienta de selección de color.

Selección de color: Asigna el color del fondo y el primer plano de los objetos.

3.1.7.-Controles e indicadores del panel frontal

Los controles permiten al usuario, en tiempo de ejecución del VI, introducir datos, parámetros, opciones, etc., son pues las entradas al VI. Los indicadores permiten conocer resultados de la ejecución del VI ya sea de forma numérica, gráfica, textual, etc., son las salidas del VI.

Estos objetos tienen un terminal correspondiente en el diagrama de bloques. La etiqueta propia (label) es la misma en el terminal que en su objeto.

Nota: Si se pincha fuera de la etiqueta antes de introducir el texto, ésta desaparece. Se la vuelve a mostrar pulsando el botón derecho dentro del control y seleccionando Show_Label.

Se pueden tener controles e indicadores de todos los tipos que aparecen en la paleta de controles, entre ellos vamos a ver los siguientes:

- Numéricos
- Boléanos
- De cadena de caracteres (string)
- Enumerados
- Arrays
- Clusters
- Gráficos

Controles numéricos

Entre otros, los controles e indicadores numéricos (Figura 3.4) pueden ser digitales, de cursor deslizante, rotatorio y enumerado. Estos últimos son un caso especial y se ven en un apartado aparte.

Se puede cambiar entre otros aspectos la representación y el formato en el menú contextual del objeto.

Representación: Los datos numéricos, en general, pueden ser enteros, de coma flotante y complejos. Los datos enteros pueden ser con o sin signo y de 8, 16 y 32 bits (byte, word y long). Los datos en coma flotante y los complejos pueden ser de precisión simple, doble y extendida.

Formato: Se puede elegir la presentación numérica o temporal, el número de dígitos decimales y el tipo de notación.

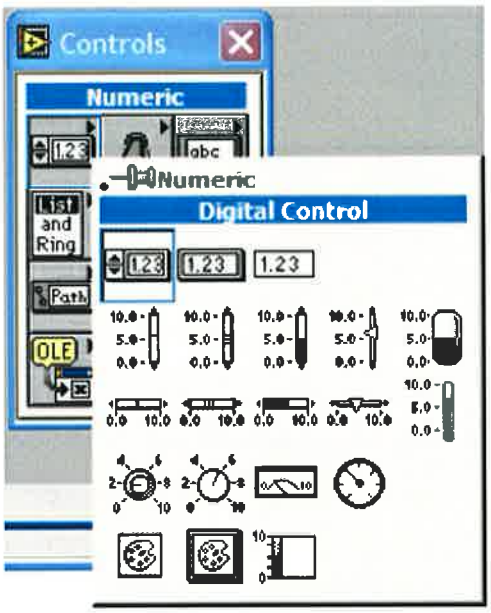


Figura 3.4.- Controles numéricos

Controles boléanos

Son botones, LEDs e interruptores. El tipo de datos de estos controles es boleano, sólo tiene dos valores: verdadero (true, on) y falso (false, off). El valor de los botones sin pulsar y de los interruptores abajo es falso (off).

Configuración de la acción mecánica de los controles boléanos

Se refiere al comportamiento del control booleano ante la acción del ratón. El comportamiento puede ser el típico de un interruptor, o el de un pulsador con o sin enclavamiento.

La acción, además, puede ser debida a la pulsación del botón o a su liberación. Hay 6 tipos de acciones mecánicas posibles para los controles boléanos del panel frontal. Estas acciones se seleccionan mediante el menú contextual.

En los esquemas de los iconos de la figura 3.5 la primera línea (m) expresa la acción del botón primario del ratón según que se pulse o se libere, la segunda línea (v) representa el cambio en el valor booleano del control, y la tercera línea ilustra el comportamiento del control. El comportamiento de los pulsadores con enclavamiento está relacionado con el funcionamiento de LabView, así el enclavamiento permite asegurar que el valor del control es "leído" (RD de read) antes de que el pulsador vuelva a su posición inicial.

Leído significa que el valor pasa al terminal o nodo que sigue.

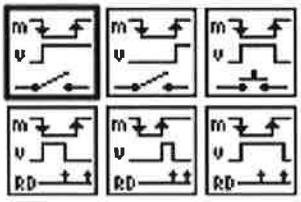


Figura 3.5.- Acción mecánica de los controles boléanos

Siguiendo el orden de los iconos, las acciones mecánicas son las siguientes:

Switch When Pressed: Se trata de un interruptor cuyo valor cambia cada vez que se pulse el botón del ratón.

Switch When Released: En este interruptor la acción se produce cuando se libera el botón del ratón.

Switch Until Released: El comportamiento es el de un pulsador, cambiando el valor desde que se pulsa el botón del ratón hasta que se suelta.

Latch When Pressed: Es un pulsador con enclavamiento, cuando se pulsa el botón del ratón el valor cambia pero no vuelve hasta que el valor del control es "leído" por primera vez desde su enclavamiento.

Latch When Released: En este pulsador el enclavamiento se produce al soltar el botón del ratón. También vuelve a su valor por defecto cuando es "leído" por primera vez desde su enclavamiento.

Latch Until Released: El pulsador se enclava al pulsarse el botón del ratón, pero se libera con la primera "lectura" después de soltar el botón del ratón.

Controles de cadena de caracteres

Las cadenas de caracteres (string) son colecciones de caracteres en código ASCII.

Controles enumerados

Se trata de listas enumeradas. Cada posición de la lista se rellena con texto y le corresponde un número entero, empezando por 0. Este valor numérico se puede ver mostrando el display digital (Men´u contextual_Show_Digital Display). Dentro del diagrama de bloques el terminal del control es un número entero con el valor de la posición seleccionada desde el panel frontal.

Arrays

Un array es una colección ordenada de datos, todos ellos del mismo tipo. El número de datos es variable. Puede tener una o más dimensiones y hasta $231 - 1$ elementos por dimensión (si hay memoria). Para acceder a cada dato se utiliza un índice por cada dimensión. Los índices empiezan en 0.

En el panel frontal se puede utilizar un array como control o como indicador. Primero se inserta el array (ver Figura 3.6) y, a continuación, se inserta un control o un indicador del tipo deseado. Todos los elementos del array serán controles o todos indicadores.

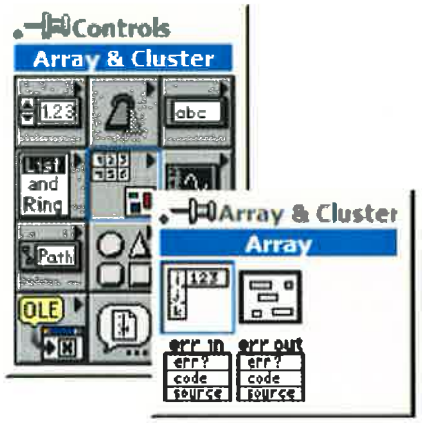


Figura 3.6.- Arrays en el panel frontal

Clusters

Colección de datos heterogénea. Los elementos pueden ser de distinto tipo pero todos deben ser controles o todos indicadores. Un cluster de control o indicador se inserta en el panel frontal de forma similar a un array.

Gestión de errores

Un tipo de cluster predefinido se utiliza para el manejo de errores. Los cluster error in y error out permiten transmitir los posibles errores de un subVI a otro. Estos cluster están constituidos por tres componentes:

- Un valor boleano indica la presencia o ausencia de error.
- Un valor numérico entero se corresponde con el número de error.
- Una cadena de caracteres permite conocer el origen del error.

Indicadores gráficos

De los cinco tipos de gráficos se explican los gráficos Waveform Chart y Waveform Graph.

Waveform Chart

Permite representar gráficos de datos numéricos equiespaciados. El gráfico se representa punto por punto. Los números escalares que llegan al indicador son la coordenada Y del gráfico mientras que la coordenada X de cada punto se calcula automáticamente desde un valor inicial X0 (por defecto 0) y se incrementa con dX (por defecto 1).

El menú contextual del gráfico permite auto escalar las coordenadas X e Y , modificar los valores por defecto de X0 y dX, cambiar el formato y precisión del gráfico y las opciones de la rejilla del gráfico (Figura 3.7).

Waveform Graph

Se trata de un indicador gráfico de datos numéricos equiespaciados. A diferencia del Waveform Chart, este no tiene memoria, no se representa punto por punto sino como un bloque.

Para representar un gráfico el dato que llega debe ser un array. Cada elemento del array será la coordenada Y del punto. Los puntos quedan representados en el mismo orden que están en el array empezando por el de menor índice.

La coordenada X del primer punto (X0) se incrementa desde el valor del primer elemento del array con cada punto en la cantidad dX.
El menú contextual del Waveform Graph permite las mismas opciones que el Waveform Chart.
Para representar más de un gráfico, en lugar de un array unidimensional, se debe enviar un array bidimensional donde cada fila es un gráfico distinto, como en la Figura 3.8.

3.1.8.-Programación del diagrama de bloques

En la ventana de diagrama de bloques se insertan nodos, terminales y constantes, y se cablean mediante la herramienta de cableado.

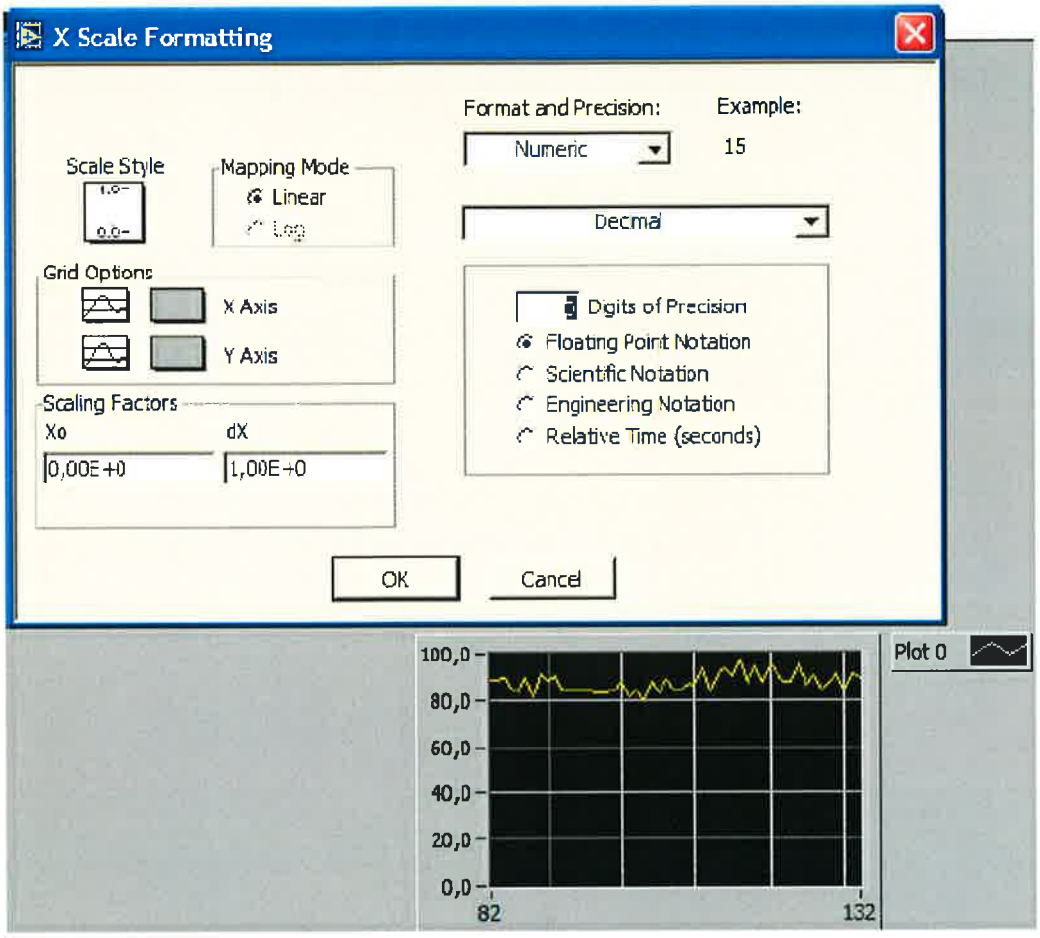


Figura 3.7.- Formato de un gráfico Waveform Chart

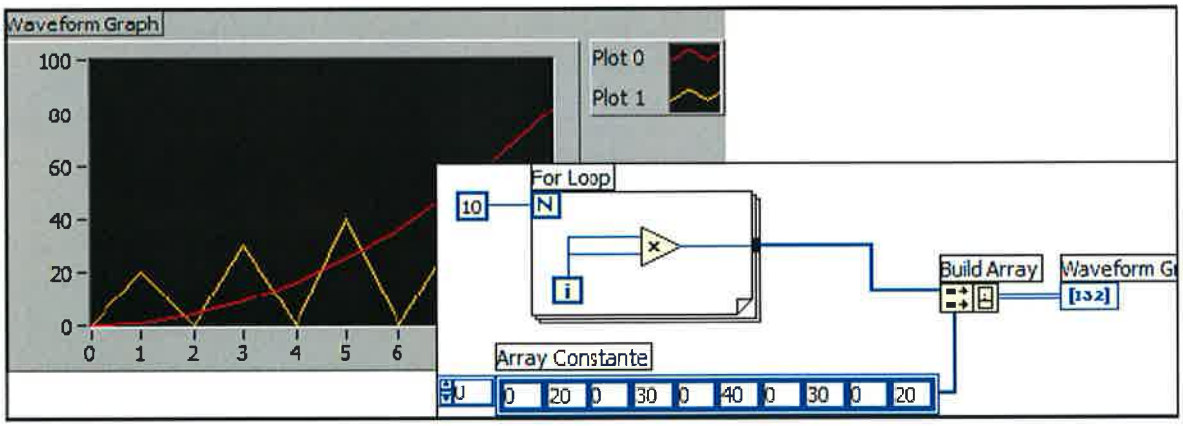


Figura 3.8.- Dos gráficas en un Waveform Graph

Los terminales aparecen automáticamente al insertar un control o un indicador en el panel frontal. Los nodos y constantes se insertan de la paleta de funciones. Los nodos pueden ser estructuras, funciones y subVIs.

Las estructuras tienen una funcionalidad similar a la de un lenguaje textual de alto nivel. Entre estas estructuras están los bucles for y while y las estructuras case. En este apartado se ven también algunas de las funciones disponibles que realizan operaciones con los distintos tipos de datos.

La programación modular en LabView se realiza mediante los SubVIs. Estos se ven mas adelante.

Para la programación se tiene en cuenta el funcionamiento de la ejecución en LabView:

- Un nodo sólo se ejecuta cuando tiene todos los datos disponibles.
- Durante la ejecución de un nodo el cambio en sus entradas no le afecta.
- Un nodo no proporciona datos a sus salidas hasta que no termina su ejecución.

Bucle While

Estructura repetitiva de bucle While (While Loop) que se ejecuta mientras la condición de alida sea cierta. Tiene dos terminales, el terminal de condición y el terminal de iteración. El subdiagrama del bucle While se sigue ejecutando mientras el valor que le llega al terminal de condición sea cierto. El terminal de iteración indica el número de iteración en la que se encuentra, empezando en 0.

Un bucle While inicia su ejecución cuando todos los datos de entrada están disponibles. Estos datos entran a través de túneles en la primera iteración. Cuando el valor que llega al terminal de iteración es falso termina el bucle y se proporcionan resultados a su salida.

Nota: Como todo Nodo del diagrama de bloques su ejecución se produce cuando los datos a su entrada están disponibles. Durante su ejecución aunque cambien la entradas estas no tienen efecto. Los resultados de salida sólo se proporcionan después de la última iteración, durante la ejecución no proporciona resultados fuera del bucle While. Aplicación: Un botón de Stop debe estar dentro del bucle While lo mismo que un indicador de valores para cada iteración.

Bucle For

Una estructura de bucle For (For Loop) ejecuta el subdiagrama que se encuentra en su interior un número de veces predeterminado.

El bucle For tiene dos terminales:
El terminal N (terminal de cuenta) indica el número de veces que se repetirá la ejecución. Se puede cablear un dato de tipo entero desde el exterior de la estructura hasta este terminal o bien utilizar las características de auto indexado.

El terminal de iteración "i" proporciona un valor entero con el número de iteración actual que se está ejecutando. Va desde 0 hasta N − 1. Además de estos terminales, datos externos pueden entrar a través de túneles (siempre antes de la primera ejecución) y salir a través de túneles después de que se ha ejecutado N veces el bucle.

Auto indexado

Son dos características conjuntas de un bucle que permiten indexar y acumular arrays automáticamente en sus límites. El auto indexado se aplica a los túneles de entrada y salida del bucle.
En los túneles de entrada permite extraer de uno en uno los elementos de un array (empezando por el de menor índice). En este caso no es necesario cablear el terminal N.
En los túneles de salida es posible acumular valores para formar un array que se proporciona al final de la última iteración. Los elementos escalares se acumulan secuencialmente en arrays de dimensión 1, los arrays de dimensión 1 se acumulan en arrays de dimensión 2, etcétera.

Mediante el menú contextual de un túnel se puede activar o desactivar su auto indexado. Esto es independiente para cada túnel. Si el túnel de salida tiene el auto indexado desactivado sólo sale del bucle el valor de la última iteración.
Esta característica es propia también de los bucles While. En los bucles For el auto indexado está activado por defecto mientras que en los bucles While está desactivado.

Nota: El número de veces que se ejecuta un bucle será el menor de los posibles: el que indica el terminal de cuenta, o el del menor array que entra por un túnel con el auto indexado activado.

Registros de desplazamiento

Los registros de desplazamiento (Shift Register) se pueden utilizar en los bucles while y for. Tienen utilidad para calcular valores medios y construir filtros digitales o máquinas de estados.

Para añadir un registro de desplazamiento se selecciona con el menú contextual del bucle en los lados izquierdo o derecho.

El valor de la iteración actual se puede guardar conectándolo a la entrada del registro en el lado derecho del bucle. Los valores de iteraciones previas se obtienen de las salidas del registro en el lado izquierdo del bucle, tal como se ilustra en la figura 3.9.

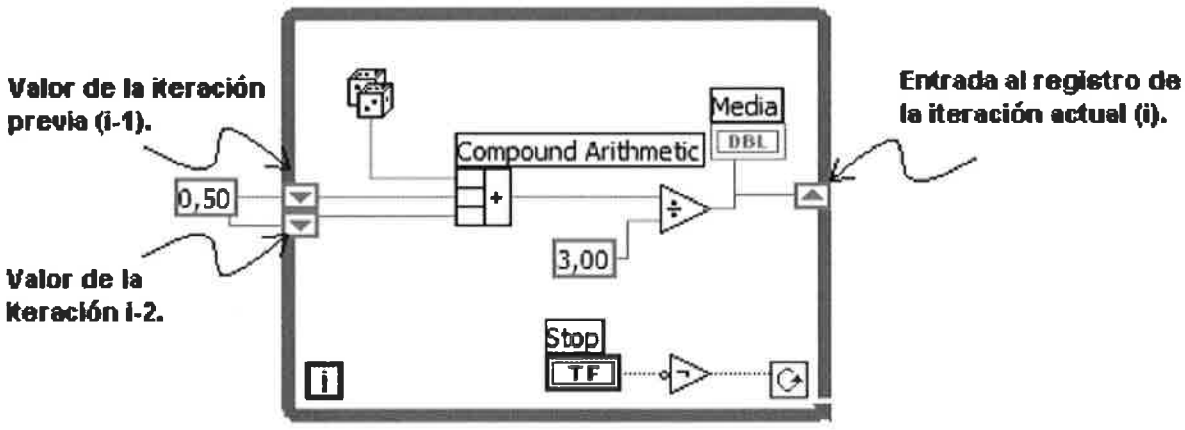


Figura 3.9.- Valor medio de la iteración actual y las dos anteriores

Estructura case

Una estructura case permite ejecutar condicionalmente uno de entre varios subdiagramas. El diagrama que se ejecuta corresponderá al valor que llega al terminal selector. El tipo de datos que puede llegar al selector puede ser boleano, entero, enumerado o de cadena de caracteres.

Por defecto el tipo de datos del selector es boleano y sólo se puede seleccionar entre dos subdiagramas. Si se quiere seleccionar entre mas de dos subdiagramas se puede, por ejemplo, cambiar los valores del selector a datos enteros. Para conseguir esto se cablea un valor entero al selector, el valor falso se convierte en 0 y valor por defecto, y el valor verdadero pasa a ser 1. Para añadir mas casos se selecciona un valor (preferiblemente el mayor disponible) y se pincha en "Add case" del menú contextual.

Túneles

Los datos de entrada pasan a cada subdiagrama a través de túneles. Estos datos se toman antes de la ejecución del subdiagrama seleccionado. Los datos de salida del caso seleccionado salen a través de túneles una vez que se ha ejecutado.

Los túneles de entrada no necesitan estar conectados dentro de cada subdiagrama pero sí los túneles de salida.

Funciones y constantes numéricas

Se incluyen constantes, definibles por el usuario, de tipo numérico y enumerado y constantes predefinidas en la subpaleta Additional Numeric Constant.

Polimorfismo de las funciones

La misma función numérica actúa sobre distintos tipos, dimensiones o representaciones:

Sobre números escalares (enteros, de coma flotante y complejos).

Arrays de tipo numérico.

Cluster de tipo numérico.

Cuando los datos de entrada son de distinto tipo, en general, el resultado tiene el tipo de mayor anchura.

Funciones Booleanas

Operaciones lógicas sobre datos boléanos, entre otras (Figura 3.10) están las operaciones de and, or, or exclusivo y negación lógica.

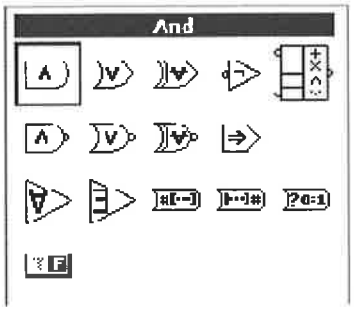


Figura 3.10.- Funciones boleanas

Funciones de cadenas de caracteres

Operaciones sobre cadenas de caracteres.

Arrays en el diagrama de bloques

Los arrays se han definido como colecciones de datos ordenadas, todos del mismo tipo.

Se pueden crear arrays directamente en el diagrama de bloques:

- Como un array constante.
- Mediante la función de inicialización de array (Initialize Array)que permite inicializar
- un array con todos los elementos iguales al de entrada.
- Mediante la función Build Array.
- Mediante un bucle for (o uno while) aprovechando las características de auto indexado.

Además de las funciones numéricas que, debido a las características de polimorfismo, pueden operar sobre datos de tipo array, existen otras funciones específicas que operan sobre todo el array.

Por ejemplo:

Array Size proporciona el número de elementos en cada dimensión de un array.

Build Array concatena elementos o arrays para construir un array.

Transpose 2D array permite pasar filas a columnas transponiendo así un array bidimensional.

Funciones sobre clusters

Los clusters son colecciones heterogéneas de datos. Si todos los datos son del mismo tipo se pueden utilizar las funciones lo mismo que con arrays.

Además existen funciones específicas:
Las funciones Bundle y Bundle By Name sirven para crear cluster a partir de sus componentes. Por el contrario las funciones Unbundle y Unbundle By Name proporcionan los componentes individuales de un cluster. Hay además dos funciones que permiten convertir entre array y cluster (para cluster con todos los elementos del mismo tipo).

También existe la posibilidad de crear un cluster constante insertando dentro del marco las distintas constantes componentes del cluster, como en la figura 3.11.

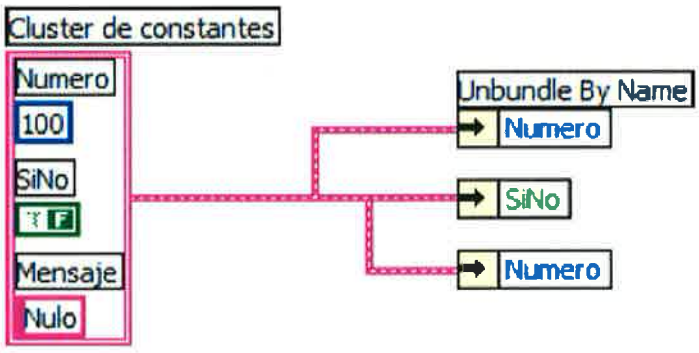


Figura 3.11.- Cluster de constantes

Funciones de tiempo y diálogo

Entre otras están la función que produce un retraso de tiempo y las de gestión de error .

3.1.9.-SubVIs

La programación modular en LabView se realiza mediante el uso de SubVIs. Un SubVI es un instrumento virtual con su propio panel frontal y su propio diagrama de bloques que está preparado para utilizarse dentro del diagrama de bloques de otro VI.

Para poder usar un VI dentro de otro VI, es necesario definir las entradas y salidas del SubVI. Las entradas serán controles y las salidas indicadores. Entradas y salidas se deben asignar a terminales del conector del icono (ver Figura 3.12).

Nota: Los controles o indicadores que no se asignen no serán entradas o salidas del SubVI.

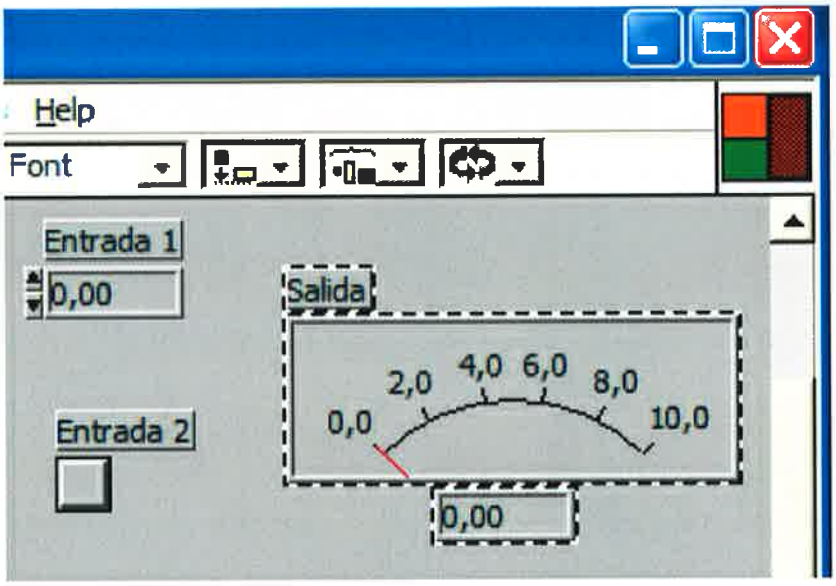


Figura 3.12.- Asignación de controles e indicadores al conector

El icono se puede diseñar para que sea más fácil su identificación en el diagrama de bloques del VI donde se inserte el SubVI. Se utiliza el editor de iconos pinchando dos veces desde el panel frontal. Este editor dispone de algunas de las herramientas típicas de los programas de diseño gráfico.

Nota: Conviene conservar el borde exterior del icono para ver más fácilmente su delimitación cuando se utilice en el diagrama de bloques.

Nota: para cambiar las propiedades de la letra se hace doble clic en el icono A de la paleta de herramientas tal como se indica en la figura 3.13:

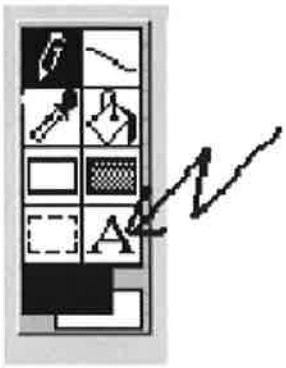


Figura 3.13.- Paleta de herramientas del Editor de Iconos

Un subVI se utiliza como un nodo más dentro de un diagrama de bloques. Se inserta desde la paleta de funciones seleccionándolo mediante el diálogo de fichero que se abre.
Para guiarse en el cableado es conveniente abrir la ayuda (Show Help del menú Help).

3.2.-Matlab

3.2.1.-Introducción al Matlab

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. MATLAB integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional.

MATLAB dispone también en la actualidad de un amplio abanico de programas de apoyo especializado, denominado Toolboxes, que extienden significativamente el número de funciones incorporadas en el programa principal. Estos Toolboxes cubren en la actualidad prácticamente casi todas las áreas principales en el mundo de la ingeniería y la simulación, destacando entre ellos el 'toolbox' de proceso de imágenes, señal, control robusto, estadística, análisis financiero, matemáticas simbólicas, redes neuronales, lógica difusa, identificación de sistemas, simulación de sistemas dinámicos, etc. es un entorno de cálculo técnico, que se ha convertido en estándar de la industria, con capacidades no superadas en computación y visualización numérica.

De forma coherente y sin ningún tipo de fisuras, integra los requisitos claves de un sistema de computación técnico: cálculo numérico, gráficos, herramientas para aplicaciones específicas y capacidad de ejecución en múltiples plataformas. Esta familia de productos proporciona al estudiante un medio de carácter único, para resolver los problemas más complejos y difíciles.

3.2.2.-Características del entorno

Características de MATLAB :

- Cálculos intensivos desde un punto de vista numérico.

- Gráficos y visualización avanzada.
- Lenguaje de alto nivel basado en vectores, arrays y matrices.
- Colección muy útil de funciones de aplicación.

Las poderosas capacidades de cálculo técnico de MATLAB se ponen a la disposición de los estudiantes, aunque limita el tamaño de las matrices a 8192 elementos, la edición de estudiante mantiene toda la potencia de la versión profesional de MATLAB 4.0, en una forma diseñada para que los estudiantes puedan ejecutarlo en sus propios ordenadores personales bajo Windows.

Toolbox especiales :

Se incluyen el Toolbox de señales y Sistemas (un conjunto de herramientas para el procesamiento de señal y para el análisis de sistemas de cuadro) y el Toolbox Symbolyc Math (herramienta de cálculo simbólico basada en Maple V).

A continuación presentamos la interfase de usuario de MATLAB 4.0 con el despliegue de una aplicación con grafica en 3D correspondiente al modelo Z=x^y-y^x su tabla de calculo y el análisis de la función.

Salidas o presentaciones

MATLAB provee acceso inmediato a las características gráficas especializadas requeridas en ingeniería y ciencias. Potente graficación orientada a objetos gráficos le permite graficar los resultados de su análisis, incorporar gráficos en sus modelos de sistemas, rápidamente presentar complejos 3-D objetos, y crear resultados de presentación, entre lo cual se destaca:

- Representaciones 2-D y 3-D, incluyendo datos triangulados y reticulados
- Representaciones 3-D quiver, ribbon, y stem
- Control de fuentes, letras Griegas, símbolos, subíndices y superíndices
- Selección expandida de símbolos marcadores de curvas
- Gráficos de torta, de barras 3-D y gráficos de barras horizontales

- y filtrando

Numerosas operaciones para manipular arreglos multidimensionales, incluyendo reticulación e interpolación de datos, están también disponibles.

Descriptivos Gráficos Para Explorar y Presentar Sus Datos

Gráficos de propósitos generales y de aplicación específica le permiten visualizar al instante señales, superficies paramétricas, imágenes y más. Todos los atributos de los gráficos de MATLAB son personalizables, desde los rótulos de ejes al ángulo de la fuente de luz en las superficies 3-D . Los gráficos están integrados con las capacidades de análisis, de modo que usted puede mostrar gráficamente cualquier conjunto de datos sin editar, ecuación o resultado funcional.

I/O Directo de Datos

Usted puede ingresar y sacar datos de f MATLAB rápidamente. Las funciones están disponibles para leer y escribir archivos de datos formateados en MATLAB, llamados archivos MAT. Funciones adicionales ejecutan programas ASCII e I/O binario de bajo nivel desde los archivos de programas M, C, y Fortran, permitiéndole trabajar con todos los formatos de datos. MATLAB también incluye soporte incorporado para formatos populares de archivos estándar.

Computación Simbólica Integrada

Integrando el motor simbólico Maple V® con MATLAB, los Symbolic Math Toolboxes le permiten mezclar libremente computación simbólica y numérica una sintaxis simple e intuitiva.

Análisis de Datos Confiable, Rápido y Exacto

Los métodos usados comúnmente para análisis de datos multidimensional generalizados 1-D, 2-D están incorporados en MATLAB. Interfaces gráficas fáciles de usar, específicas para aplicaciones, la línea de comando interactiva y herramientas de programación estructuradas le permiten elegir el mejor camino para sus tareas de análisis.

Análisis de Datos para DSP

MATLAB ofrece muchas herramientas para realizar la funcionalidad indispensable en procesamiento de señales, tales como Transformadas Rápidas Fourier y Transformadas Rápidas Inversas de Fourier. La visualización de datos de procesamiento de señales está soportada por funciones tales como gráficos stem y periodogramas. El lenguaje de MATLAB, inherentemente orientado a matrices hace que la expresión de coeficientes de filtros y demoras de buffers sean muy simples de expresar y comprender.

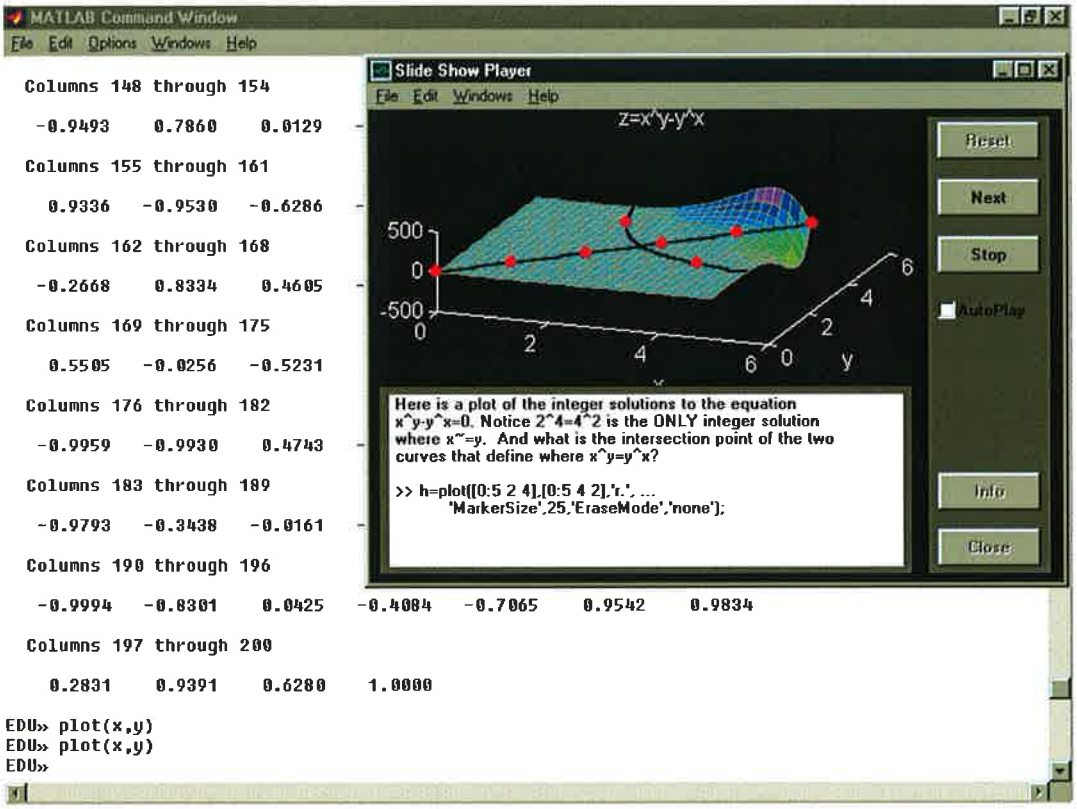
Análisis de Datos en Aplicaciones de Imágenes

MATLAB y la Image Processing Toolbox ofrece un amplio conjunto de herramientas que le permite fácilmente manipular, procesar y analizar datos de imágenes, interactivamente mostrar pantallas de imágenes 2-D o 3-D, visualizar datos temporarios cuando es necesario, y comentar sus resultados para publicaciones técnicas. La orientación basada en matrices del lenguaje de MATLAB le permite expresar en forma compacta operaciones matemáticas de forma similar a cómo las expresaría sobre papel. Como resultado, es fácil e intuitivo efectuar procesamiento de imágenes y operaciones de análisis tales como FFTs, filtrado 2-D, morfología binaria, manipulación geométrica, conversión de espacios de colores, compresión, análisis de componentes conectados y más.

Algorithm Development (Desarrollo de Algoritmos) Sea que usted esté usando los algoritmos del sistema o esté inventando los suyos propios, MATLAB le provee un ambiente en el que usted puede experimentar. A diferencia de C y C++, MATLAB le permite desarrollar algoritmos desde cero o trabajar con interfaces complicadas a bibliotecas externas. La poderosa fundación de computación, el lenguaje técnico, y cientos de funciones en cajas de herramientas (toolboxes) convierten a MATLAB en lo más adecuado para aplicaciones matemáticamente intensivas que requieran análisis de datos,

procesamiento de señales e imágenes, modelado de sistemas o técnicas numéricas avanzadas.

3.2.3.-Ventanas



Como vemos, en la figura anterior, la interfase de usuario de MATLAB no es muy distinta a la de otras aplicaciones a las cuales estamos acostumbrados, pero la verdadera diferencia consiste en la utilidad que presta como aplicación para la investigación y el desarrollo de modelos matemáticos y estadísticos los cuales son tratados de forma interactiva, y con superposición de ventanas en un entorno de fácil comprensión e interpretación de los datos arrojados como resultados de los distintos rangos de calculo que se pueden proporcionar a cada modelo de tal forma que podemos hacer estudios de comportamiento y tratar de determinar como se comportará una determinada variable a través de una serie de experimentación en tiempo real.

Las ventanas de despliegue grafico son muy similares, en las cuales el énfasis de la presentación se pone en la grafica generada y no en el entorno de trabajo, es por esta razón

que puede parecer que el diseño de esta aplicación es escueto, pero debemos recordar que como todo este tipo de aplicaciones su desarrollo está orientado al logro de un objetivo especifico como es el resolver modelos matemáticos.

Operaciones con vectores y matrices

Definiendo Matrices y Vectores

El entorno de desarrollo nos permite resolver problemas de calculo complejo y es asi como en el calculo matricial y vectorial se puede hacer buen uso de MATLAB, a continuación se ejemplifica el uso del mismo, tengamos en cuenta que una matriz es un arreglo vectorial, por lo tanto el uso de las formas matriciales son aplicables a las formas vectoriales. Si queremos definir la siguiente matriz en MATLAB:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

Entonces escribimos:

»A=[1 2 3 4;5 6 7 8;9 10 11 12;13,14,15,16];

(El simbolo "»" denota el *prompt* de MATLAB y no se escribe al entrar instrucciones). El ";" al final de la instrucción omite el "eco" o salida a la pantalla. La instrucción

»x=4:-1:1

General el vector *fila* x=[4,3,2,1]. La instrucción

»C=A(3:4,1:3);

Se refiere a la submatriz

$$C = \begin{pmatrix} 9 & 10 & 11 \\ 13 & 14 & 15 \end{pmatrix}$$

de A. También D=A([1,3],3:4) genera

$$D = \begin{pmatrix} 3 & 4 \\ 11 & 12 \end{pmatrix}$$

Matrices Especiales

En MATLAB podemos generar *matrices especiales* con las siguientes instrucciones:

rand(n,m) - matriz n□m de entradas aleatorias entre 0 y uno.
eye(n) - matriz identidad n□n.
zeros(n,m) - matriz cero de tamaño n□m.
ones(n,m) - matriz n□m con todas las entradas uno.

Combinando estas instrucciones podemos generar matrices bastante complicadas. Por ejemplo, la instrucción

»E=[eye(2),ones(2,3);zeros(2),[1:3;3:-1:1]]

Genera la matriz

$$E = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 3 & 2 & 1 \end{pmatrix}$$

La instrucción round(x) redondea "x" al entero más cercano a "x". Podemos combinar funciones en MATLAB. Por ejemplo, round(10*rand(4)) genera una matriz con entradas aleatorias entre 0 y 10.

Aritmética de Matrices

Considere las siguientes matrices:

$$A = \begin{pmatrix} 4 & 5 \\ 2 & 3 \end{pmatrix} \quad , \quad B = \begin{pmatrix} -1 & 1 & 2 \\ 4 & 3 & 1 \end{pmatrix} \quad , \quad C = \begin{pmatrix} -1 & 2 \\ 2 & 4 \end{pmatrix}$$

Entonces las operaciones A*B (producto matricial de A con B), A+B (suma de A mas B), 3*A (multiplicación escalar de 3 por A) tienen los siguientes resultados:

»A*B

ans =

16 19 13
10 11 7

»A+B

??? Error using ==> +
Matrix dimensions must agree.

»3*A

ans =

12 15
 6 9

Note que MATLAB "anuncia" que A+B no se puede calcular. Las operaciones A' (transpuesto de A), inv(A) (inversa de A), y A^3 (esto es A*A*A) tienen como resultados:

»A'

ans =

```
4 2
5 3

»inv(A)

ans =

1.5000 -2.5000
-1.0000 2.0000

»A^3

ans =

174 235
 94 127
```

Si precedemos las operaciones matriciales "*", "^" con el punto ".", entonces estas se hacen termino a termino. Por ejemplo A.*C y A.^2 generan:

```
» A.*C

ans =

-4 10
 4 12

» A.^2

ans =

16 25
 4  9
```

Solución de Sistemas Lineales

Considere le sistema lineal

$$\begin{cases} x_1 - 2x_2 + 3x_3 = 1 \\ 4x_1 + x_2 - 2x_3 = -1 \\ 2x_1 - x_2 + 4x_3 = 2 \end{cases}$$

Definimos la matriz de coeficientes y el lado derecho por las instrucciones:

```
»A=[1 -2 3;4 1 -2;2 -1 4];
»b=[1 -1 2]';
```

Note el transpuesto en b para hacerlo un vector columna. Vamos a resolver este sistema por tres métodos:

- eliminación Gaussiana
- forma echelon reducida o método de Gauss-Jordan
- método de la inversa

En el método de Gauss-Jordan, luego de obtener la forma echelon de la matriz de coeficientes aumentada, eliminamos también la parte de arriba de la matriz hasta producir una matriz donde las columnas con unos, solo tienen un uno. Esto se conoce como la forma *echelon reducida* (ver texto). Para comparar los tres métodos utilizamos la instrucción flops de MATLAB que estima el número de operaciones de punto flotante entre dos llamadas sucesivas a flops. Una llamada de la forma flops(0) inicializa el contador de operaciones a cero. La sucesión de instrucciones:

```
» flops(0)
» x=A\b

x =
```

-0.0417
 0.4167
 0.6250

» flops

Lleva a cabo eliminación Gaussiana en el sistema de arriba y produce como resultado:

ans =

73

Esto es, se necesitaron aproximadamente 73 operaciones de punto flotante (sumas, restas, multiplicaciones ó divisiones) para resolver el sistema con eliminación Gaussiana. Para el método de Gauss-Jordan tenemos:

» flops(0)
» rref([A b])

ans =

1.0000 0 0 -0.0417
0 1.0000 0 0.4167
0 0 1.0000 0.6250

» flops

ans =

483

el cual requiere 483 operaciones de punto flotante. Finalmente el método de la inversa se realiza con la siguiente secuencia de instrucciones:

```
» flops(0)
» x=inv(A)*b

x =

-0.0417
 0.4167
 0.6250

» flops

ans =

108
```

el cual toma 108 operaciones. Vemos pues que eliminación Gaussiana es el mejor de los tres métodos lo cual es cierto en general.

Usando MATLAB podemos estudiar la relación entre la solubilidad del sistema Ax=b y la no singularidad de la matriz de coeficientes A. En clase vimos que el sistema Ax=b tiene solución única para cualquier lado derecho b si y solo si la matriz A es no singular. ¿Qué sucede si A es singular? ¿Entonces Ax=b no tiene solución? Si A es singular el sistema Ax=b puede tener solución para algunos b's pero de seguro hay al menos un b* para el cual Ax=b* no tiene solución. Vamos a genera una matriz singular con MATLAB:

```
» A=round(10*rand(6));
» A(:,3)=A(:,1:2)*[4 3]'

A =

2 5 23 9 7 3
0 8 24 8 9 6
7 0 28 5 8 8
7 1 31 1 3 10
```

9 5 51 7 0 4
4 7 37 4 7 2

(Como usamos la instrucción rand, el resultado de esta y cualquier secuencia de instrucciones que use esta función de MATLAB, no siempre será el mismo). La primera instrucción genera una matriz aleatoria con entradas enteras entre 0 y 10, y con la segunda instrucción remplazamos la tercera columna de A con cuatro veces la primera columna más tres veces la segunda columna. ¡La matriz resultante es singular! (Explique esto sin calcular el determinante). Generamos ahora un lado derecho arbitrario mediante la instrucción:

» b=round(20*(rand(6,1)-0.5))

b =

10
4
5
3
-9
3

Esto genera una matriz 6□1 aleatoria con entradas enteras entre -10 y 10. Resolvemos el sistema Ax=b calculando la forma echelon reducida de la matriz de coeficientes aumentada [A b]:

» rref([A b])

ans =

1 0 4 0 0 0 0
0 1 3 0 0 0 0
0 0 0 1 0 0 0
0 0 0 0 1 0 0

0 0 0 0 0 1 0

0 0 0 0 0 0 1

Como la última fila es de la forma $(0 \cdots 0 | 1)$ el sistema es inconsistente, i.e., no tiene solución. ¡Recuerde que A es singular! Esto no quiere decir que Ax=b nunca tenga solución. Si definimos c=A*b, con el b de arriba digamos, el sistema Ax=c tiene solución x=b (¿por qué?). De hecho si calculamos la forma echelon reducida de [A c] tenemos:

» c=A*b;

» rref([A c])

ans =

1 0 4 0 0 0 30

0 1 3 0 0 0 19

0 0 0 1 0 0 3

0 0 0 0 1 0 -9

0 0 0 0 0 1 3

0 0 0 0 0 0 0

el cual denota un sistema consistente dependiente con soluciones:

$$(x_1, x_2, x_3, x_4, x_5, x_6) = (30 - 4x_3, 19 - 3x_3, x_3, 3, -9, 3)$$

donde x3 es arbitrario.

Funciones de Matrices

MATLAB posee una gran cantidad de funciones matriciales. De las más comunes tenenmos:

- min(A), max(A) - dan el mínimo y máximo respectivamente por columnas de A
- sum(A), prod(A) - producen la suma y producto respectivamente por columnas de A

- norm(A,p) - norma p de la matriz A donde p=1,2, ó inf
- eig(A) - vector cuyos componentes son los valores propios de A
- det(A) - el determinante de A
- inv(A) - la matriz inversa de A

3.2.4.-Gráficas

MATLAB provee excelentes funciones para gráficas en dos, tres y cuatro dimensiones. Veamos un par de ejemplos sencillos. Suponga que queremos trazar la gráfica de la función

$$f(x) = x^2 e^{-x^2} \quad , \quad x \in [-5,5]$$

Esto lo podemos lograr con las instrucciones:

```
» x=-5:.1:5;
» y=x.^2.*exp(-x.^2);
» plot(x,y)
```

El resultado gráfico de estas instrucciones se pueden observar en la figura 3.14.

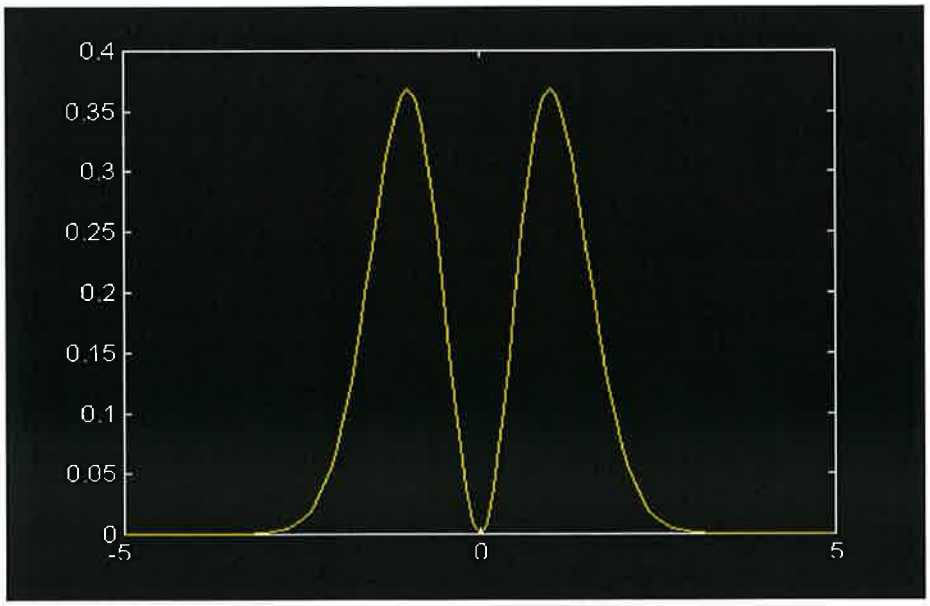


Figura 3.14.- Gráfica en MatLab

La primera instrucción divide el intervalo [-5,5] en subintervalos de largo 0.1, la segunda instrucción evalúa la función en los puntos de la partición, y finalmente graficamos los resultados con plot. La instrucción plot tiene opciones para cambiar patrones del trazado, poner títulos, etc.

Supongamos ahora que queremos dibujar la superficie:

$$f(x,y) = x^2 e^{-y^2} \quad , \quad (x,y) \in [-5,5] \times [-5,5]$$

Esto lo hacemos con la secuencia de instrucciones:

```
» x=-5:.4:5;
» y=x;
» [X,Y]=meshgrid(x,y);
» Z=X.^2.*exp(-Y.^2);
» surf(X,Y,Z)
```

El resultado gráfico de estas instrucciones se puede observar en la figura 4.15.

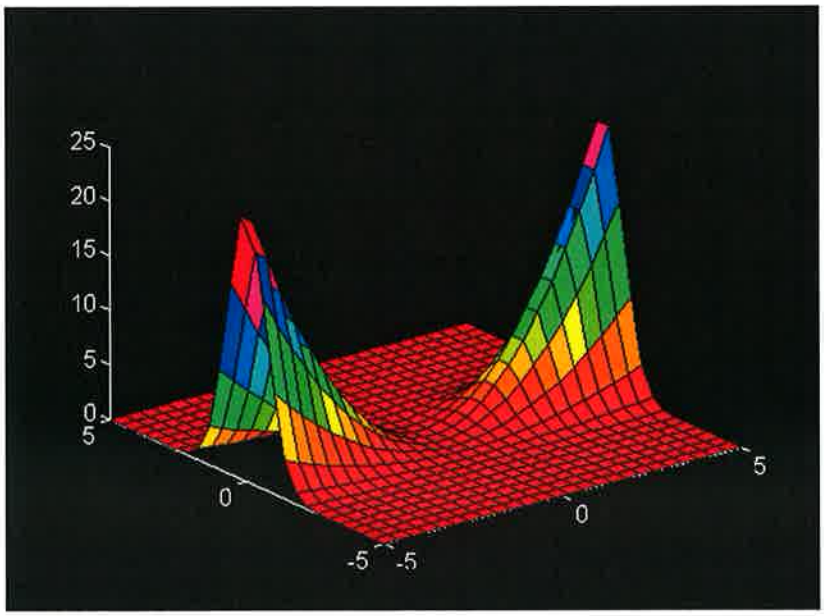


Figura 3.15.- Superficie en MatLab

Las primeras dos instrucciones dividen los ejes de "x" y "y" en subintervalos de largo 0.4; la tercera instrucción genera una rejilla en el conjunto [-5,5]□[-5,5] con cuadraditos de lados 0.4 como se ilustra en la siguiente figura:

La cuarta instrucción evalúa la función en los puntos de la rejilla, y finalmente trazamos la superficie con surf.

3.2.5.-Funciones especiales

Lista parcial de funciones

Funciones matemáticas

Funcionales especiales y elementales

- Funciones gamma, beta y elípticas.

- Transformación de sistemas de coordenadas.
- Matriz identidad y otras matrices elementales.
- Matrices de Hilbert, Toeplitz, Vandermonde, Hadamard, etc.
- Partes reales, imaginarias y complejas conjugadas.
- Funciones trigonométricas y de potencias.

Algebra lineal numérica

- Valores propios y descomposición de matrices.
- Funciones generales de evaluación de matrices.
- Determinantes, normas, rangos, etc.
- Matrices inversas y factorización de matrices.
- Matriz exponencial, logarítmica y raíces cuadradas.

Polinomios e interpolación

- Interpolación 1-D y 2-D.
- Construcción polinomial.
- Interpolación por splines cúbicos.
- Diferenciación de polinomios.
- Evaluación de polinomios.
- Multiplicación y división de polinomios.
- Residuos de polinomios y residuos.

Métodos numéricos no lineales

- Búsqueda de ceros en funciones de una única variable.
- Minimización de funciones de una o más variables.
- Resolución numérica de integrales.

- Solución numérica de ecuaciones diferenciales ordinarias.

Estadística y análisis de Fourier

- Convolución 1-D y 2-D.
- Filtros digitales 1-D y 2-D.
- Transformadas de Fourier 1-D y 2-D y su inversa.
- Coeficientes de correlación y matrices de co-varianza.
- Deconvolución.
- Magnitudes y ángulos de fase.
- Funciones max, min, sum, mean y otras funciones de estadística básica.

Operaciones algebraicas y lógicas

- Suma, resta, multiplicación, división y potencias de matrices.
- Matrix traspuesta.
- Operadores lógicos AND, OR, NOT y XOR.

Utilidades

- Gestión y mantenimiento de errores.
- Conversión de tipos de datos Fortran.
- Funciones de fecha y hora.
- Clasificación de matrices.
- Conversión de números a cadenas y viceversa

Ahora ya conocemos las funcionalidades y herramientas que nos proporcionan estos dos lenguajes de programación: LabView y MatLab. Con ellos podemos fácilmente crear una interfaz de usuario potente que nos ayude a controlar el motor, adquirir imágenes para su posterior procesamiento y finalmente llevar a cabo la reconstrucción en 3D.

Lo que nos corresponde ahora es aprender respecto al procesamiento de imágenes.

Capítulo 4. Procesamiento De Imágenes

4.1.-Representación de Imágenes Digitales

Una imagen digital es una imagen que ha sido digitalizada tanto espacialmente como en el valor del brillo. La representación de una imagen de intensidad se lleva a cabo por una matriz g(x,y), que representa la distribución de intensidades en el espacio (x,y), donde el valor de g en un determinado punto es proporcional al brillo (nivel de gris). Los elementos de una imagen digital son comúnmente llamados píxeles.

$$g(x,y) = \begin{bmatrix} g(1,1) & g(1,2) & \dots & g(1,N) \\ g(2,1) & g(2,2) & \dots & g(2,N) \\ \cdot & \cdot & \cdot & \cdot \\ g(M,1) & g(M,2) & \dots & g(M,N) \end{bmatrix}$$

A la digitalización espacial de una imagen se le llama comúnmente muestreo de imagen, y a la digitalización del brillo se le llama cuantización de nivel de gris. Si representamos una imagen digital por una matriz $g(x,y)$ donde los valores de x e y están igualmente espaciados en un arreglo de M x N elementos, y los valores del nivel de gris están acotados, esto es, $0 <= g(x,y) <= K$; en el proceso de digitalización es importante determinar el valor de M, N y K. La resolución o grado de detalle discernible de una imagen depende fuertemente de estos valores. En cuanto más grandes sean estos valores, la imagen digital se parecerá mas a la imagen original. Un tamaño típico de imagen digital es el de un arreglo de 512 x 512 pixeles con 256 niveles de gris. En este caso, el valor cero para un píxel representa el color negro, y el valor 255 representa el color blanco, esto quiere decir que los valores oscilan entre 0 y 255. La Figura 5.1 muestra una imagen en niveles de gris en donde se indican las coordenadas y una barra indicando la cuantización del nivel de gris.

Figura 4.1.- Imagen de intensidad con 256 niveles de gris

4.2.-Procesamiento Digital de Imágenes

A continuación se describirá los pasos fundamentales del procesamiento digital de imágenes. Dependiendo del tipo de aplicación, todos o algunos de los siguientes pasos se llevan a cabo:

- Captura de Imagen
- Pre-procesamiento de la imagen
- Segmentación de Objetos
- Procesamiento de las características del objeto

4.2.1.-Captura de la imagen

Este paso consiste en la captura o adquisición de la imagen digital. Para hacer esto, se requiere un sensor de imágenes y un dispositivo capaz de digitalizar la señal producida por el sensor. El sensor puede ser una cámara analógica con puerto USB o firewire.

4.2.2.-Pre-procesamiento

Después de la adquisición de la imagen el preproceso comúnmente es muy importante para mejorar las expectativas de éxito del procesamiento principal de la imagen para obtener la información de interés. En este paso, las tareas típicas son el mejoramiento de contraste, la eliminación del ruido, y la definición de las regiones de interés.

4.2.3.-Segmentación de objetos

Este paso consiste en el aislamiento de los objetos a procesar en la imagen. En general, el proceso de segmentación de forma automática es un trabajo difícil y de gran importancia en procesamiento digital de imágenes para la solución exitosa del problema global. En la segmentación es común llevara a cabo la operación de binarización de imagen; esto es, convertir una imagen normal en una imagen binaria. Una imagen binaria es una imagen en la cual se asume que cada píxel puede tener únicamente dos valores. Esencialmente, estos dos valores corresponden a 1 y 0. Mirando una imagen de esta manera se hace más fácil de distinguir sus características estructurales, por ejemplo, es más fácil distinguir objetos, del fondo de la imagen.

4.3.-Elementos de los Sistemas de Procesamiento de Imágenes

Un sistema de procesamiento digital de imágenes de propósito general consiste de varios elementos o dispositivos para llevar a cabo las operaciones. Estos elementos o dispositivos generalmente llevan a cabo los siguientes trabajos.

- Adquisición de la imagen
- Almacenamiento de la imagen
- Procesamiento de la imagen
- Despliegue de la imagen

La cantidad de datos asociada con información visual (imágenes) es comúnmente grande. El almacenamiento de este tipo de información se mide en Kbytes (Mil bytes), Mbytes (millón de Bytes), Gbytes (billón de Bytes) o Tbytes (Trillón de Bytes). Por ejemplo, una imagen de intensidad de 512 por 512 pixeles que requiere de un Byte de memoria por elemento, requiere de 0.27 Mbytes para ser almacenada.

El método más común para proveer almacenamiento de imágenes es la memoria de computadora (RAM). Oros dispositivos como los discos magnéticos y ópticos son también usados para el almacenamiento de imágenes digitales.

4.3.3.-Elementos para el procesamiento de la imagen

La parte medular en un sistema de procesamiento de imágenes es el procesamiento de la información visual que usualmente se expresa en forma de algoritmos computacionales, lo cual involucra la utilización e implementación de software especializado y el uso de computadoras personales o especializadas para llevara a cabo tareas específicas. Los fundamentos y tópicos esenciales para la implementación de algoritmos de procesado de imágenes son uno de los temas principales de la estancia técnica.

4.3.4.-Elementos de despliegue de imágenes

Los monitores de TV o de computadora son los dispositivos mayormente usados para el despliegue de imágenes en sistemas de procesamiento. En los primeros años del procesamiento de imágenes el despliegue de imágenes también se llevó a cabo usado impresoras, principalmente para trabajos de procesamiento de baja resolución. El uso de caracteres de impresión se fue para simular los diferentes niveles de gris sobre la imagen.

4.4.-Mejoramiento de Imágenes

Esta tarea se aplica principalmente para mejorar la calidad de la imagen para poder llevar a cabo de una manera exitosa la recuperación de la información que nos interesa contenida en la imagen. Las tareas mas comunes son las de eliminación de ruido a través de filtrado, y el mejoramiento del contraste. El tipo de técnica de mejoramiento que se aplica a las imágenes depende fuertemente de la aplicación en específico en la que se esté trabajando.

Los tipos de tratamientos para el mejoramiento de imágenes se dividen principalmente en: métodos en el *dominio espacial* y métodos en el *dominio frecuencial*. El primero de ellos se refiere al tratamiento directo de la imagen que consiste en la manipulación de los píxeles. El segundo de ellos se basa en la modificación de las características de la imagen en el dominio de Fourier.

4.5.-Mejoramiento por procesamiento puntual

El mejoramiento por procesamiento puntual se basa en el procesado de la imagen a través del análisis de la intensidad de cada píxel. Este tipo de técnica se considera una de las más sencillas para el mejoramiento de imágenes.

4.6.-Negativo de imagen

El negativo de una imagen digital se obtiene invirtiendo los valores de nivel de gris, esto es, aplicando la función de transformación $s=T(r)$ donde r y s representan el valor del nivel de gris antes y después de la transformación, y L el número de niveles de gris, como se muestra en la Figura 5.3.

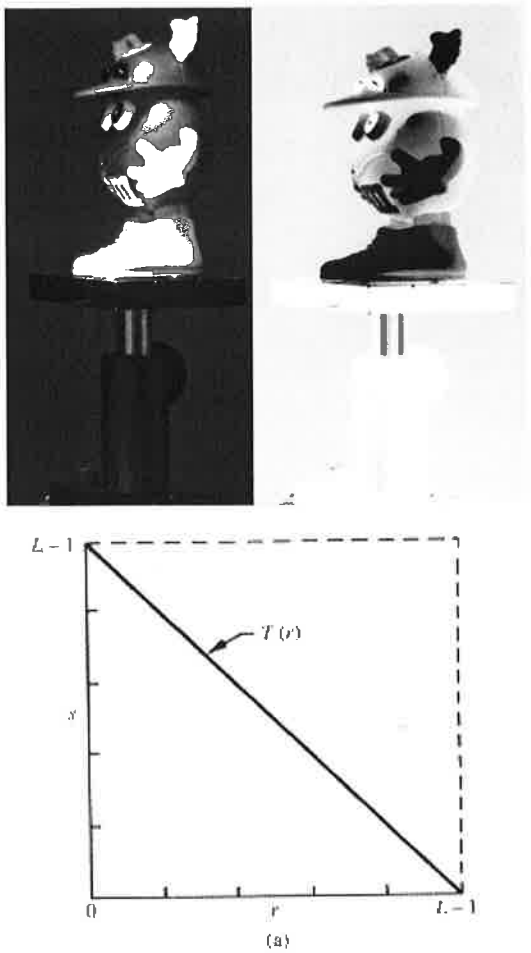


Figura 4.3.- Obtención del negativo de una imagen y su función de transformación

4.7.-Estrechamiento de contraste

Las imágenes que tienen poco contraste son debidas a la baja iluminación a la hora de la captura, al rango dinámico pequeño de sensibilidad que en ocasiones tienen los censores. En estos casos, el contraste puede ser mejorado digitalmente incrementando el rango dinámico de los niveles de gris contenidos en la imagen. La Figura 5.3 muestra la función de transformación y el resultado de un algoritmo implementado en LabView. Éstas imágenes es el resultado de una transformación típica usada para estrechamiento de contraste. Los puntos (r_1,s_1) y (r_2,s_2) controlan la forma de la función de transformación. Por ejemplo, si $r_1=s_1$ y $r_2=s_2$ la transformación es una función lineal que no produce cambios en la escala de grises. Si r1=r2, s1=0 y s2= L-1, la función de transformación representa una función de binarización que produce una imagen binaria. En general, se

asume que $r_1 \leq r_2$ y $s_1 \leq s_2$, esto representa una simple función que crece monotónicamente.

Finalmente contamos con todas las herramientas necesarias para poder integrar todos los módulos que formarán nuestro sistema y que nos permitirán buscar y lograr el objetivo propuesto.

Capítulo 5. Metodología

5.1.-Descripción de la Metodología

Inicialmente el proyecto estaba definido como el desarrollo de una mesa X-Y para la reconstrucción de objetos 3D. Para ello se inicio por diseñar la electrónica que permitiera el mejor control de los motores, por lo que antes que nada habría que definir el tipo de motores a utilizar, ya que de ello dependía la estrategia a seguir, principalmente para llevar a cabo el control de la velocidad y movimiento de los motores a utilizar.

Debido a que se requería tener un control muy preciso de la velocidad del movimiento del motor tanto en el arranque como durante el movimiento, se opto por utilizar motores a pasos.

Un motor a pasos unipolar es considerablemente más fácil de controlar e identificar su secuencia de excitación que un motor a pasos bipolar, además, como ya se tenía experiencia en el manejo de estos, no generaba ningún aprendizaje nuevo; debido a esto, se opto por utilizar motores a pasos bipolares.

Debido a la estructura y funcionamiento de los motores a pasos bipolares, la mejor forma de controlar sus movimientos es utilizando el denominado Puente H, que no es mas que un arreglo de cuatro interruptores que permiten circular la corriente en un sentido u otro a través de las bobinas del motor, permitiendo así el control de movimiento y giro del mismo.

En el mercado hay diferentes circuitos integrados que tienen la funcionalidad de un Puente H y que manejan diferentes corrientes dependiendo del consumo requerido por el motor. Pero, utilizar estos hubiera sido muy fácil su implementación, por lo que se opto por crear el circuito del puente H mediante transistores.

Una vez implementado el circuito del puente H se valido su funcionamiento utilizando un motor de cd. En este tipo de motores se cambia la dirección del giro del motor cambiando

el sentido de circulación de la corriente. Por lo que basto con conectar los cables correspondientes al positivo y negativo del motor a las salidas del puente H que controlaría una bobina y alimentar el puente H con un "1" y un "0" o con un "0" y un "1" para hacer que el motor cambiara de sentido de giro.

El paso siguiente fue identificar los cables que pertenecían a cada una de las bobinas del motor de pasos bipolar para, posteriormente, identificar las secuencias de excitación que permitirían el movimiento del motor.

Mediante un multímetro se identificaron los cables que mostraban tener continuidad entre ellos, con esto se identificaron los cables de cada bobina.

En la sección 3.3.5 se describe el funcionamiento y control de los motores a pasos bipolares además viene definida la secuencia de alimentación que hay que enviar a través de las bobinas del motor para provocar el movimiento del mismo. En ellas viene descrito el envió de los valores por cada uno de los 4 cables de un motor a pasos bipolar, identificando estos cables como A,B,C y D. Lo difícil es identificar que cable corresponde al A, cual al B cual al C y cual al D. Pero una vez identificadas los cables de las dos bobinas se reduce el margen de búsqueda, ya que los cables A y B pertenecen a una bobina y los cables C y D a la otra.

Nuestra búsqueda se disminuye un poco, aunque finalmente dicha búsqueda se tuvo que llevar a cabo a prueba y error, enviando valores entre los cables de una bobina y los cables de otra.

Una vez que se tenia funcionando el circuito del puente H y el motor a pasos bipolar a través del mismo, el paso siguiente fue implementar la electrónica necesaria para integrar el sensor o detector del rayo láser así como el rayo láser mismo.

El resultado de la integración de los módulos que permitieran llevar a cabo el control del motor, adquisición y procesamiento de las imágenes fue lo que finalmente se convertiría en la interfaz del usuario. En ella se fueron integrando poco a poco distintos controles e indicadores que permitieron parametrizar, controlar y definir algunas variables además de mostrar cierta información que facilitaba llevar a cabo las pruebas del proceso completo de lo que ahora es nuestro sistema.

A estas alturas, el programa en LabView que se había convertido en la interfaz del usuario había crecido considerablemente ocasionando una complejidad tanto para su seguimiento como para su interpretación. Para eliminar esta complejidad se pensó en crear un SubVI que permitiera hacer más legible el seguimiento del diagrama en bloques de la interfaz del usuario.

Finalmente se creo un SubVI con la funcionalidad que se pudo "encapsular" en el mismo (Figura 6.3), y a pesar de que fue mucha la programación que se envió a éste, el diagrama de bloques de la interfaz del usuario siguió con cierta complejidad en su interpretación.

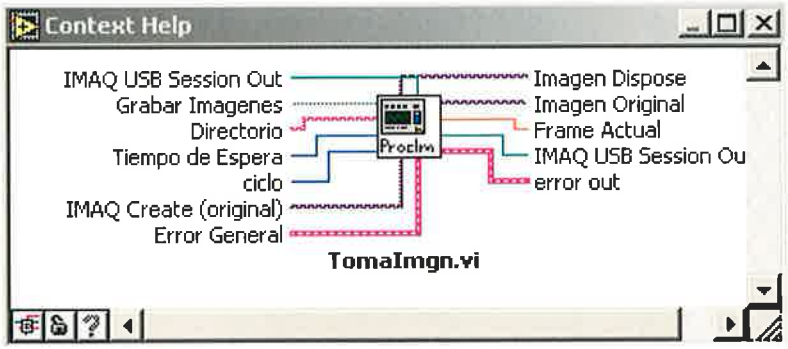


Figura 5.3.- SubVI creado para integrar funcionalidad

Cuando se iniciaron las pruebas para evaluar el funcionamiento y sincronización entre el movimiento del motor y la toma de las imágenes, nos percatamos que los movimientos del motor que estábamos utilizando eran muy bruscos y al momento de hacer las tomas ocasionaba que las imágenes salieran un poco movidas.

Por lo que decidimos cambiar el motor por otro de mayor ángulo y que finalmente trabajando con el a medios pasos nos dio como resultado que los movimientos eran muy suaves y esto nos repercutió en una mejor calidad en las imágenes.

Los avances se iban dando pero para lograr concluir el procesamiento de imágenes en LabView capaz de llevar a cabo la reconstrucción en 3D se requería de más tiempo, pero el tiempo se nos agoto, por lo que se decidió replantear nuevamente el alcance del proyecto.

Desafortunadamente el procesamiento de las imágenes para su reconstrucción en 3D no se logro concluir en LabView y para solventar este problema se integro al LabView una función en MatLab. La función en MatLab pertenece al Dr. Carlos Pedraza y solo se adapto un poco para lograr su funcionamiento integrado al programa en LabView.

Esta función requería de dos cosas: la primera, que se le enviara como parámetros de entrada 4 valores que correspondían al área a procesar dentro de la imagen, esto es, las coordenadas x1, y1, x2, y2 que la función tomaría como región de interés (ROI) dentro de la imagen. Y la segunda, la función necesitaba que las imágenes fueran grabadas como archivos sin ningún procesamiento en formato JPEG.

Finalmente se concluyo la instrumentación, el desarrollo e implementación de todos los componentes necesarios del sistema que nos ayudó a lograr el objetivo buscado: Llevar a cabo la reconstrucción en 3D de objetos pequeños utilizando proyección de una línea láser.

5.2.-Diagrama de Bloques del Sistema

El sistema esta compuesto de varios módulos que interactúan y se ejecutan de tal manera que llevan a cabo el funcionamiento y objetivo del mismo, la reconstrucción en 3D de un objeto.

El sistema esta integrado por los siguientes módulos (ver Figura 5.4):

1. Interfaz del usuario
2. Control de movimiento del motor
3. Adquisición de imágenes
4. Reconstrucción del objeto

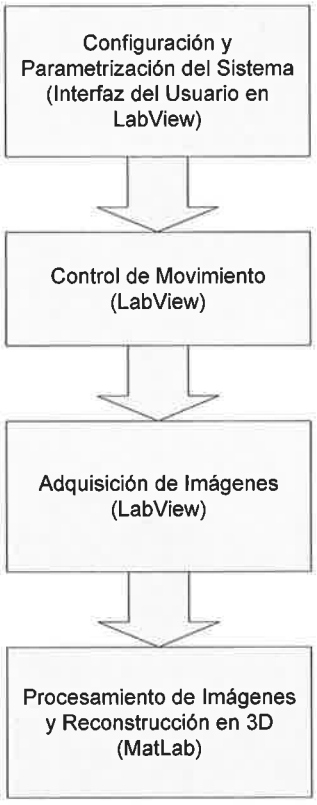
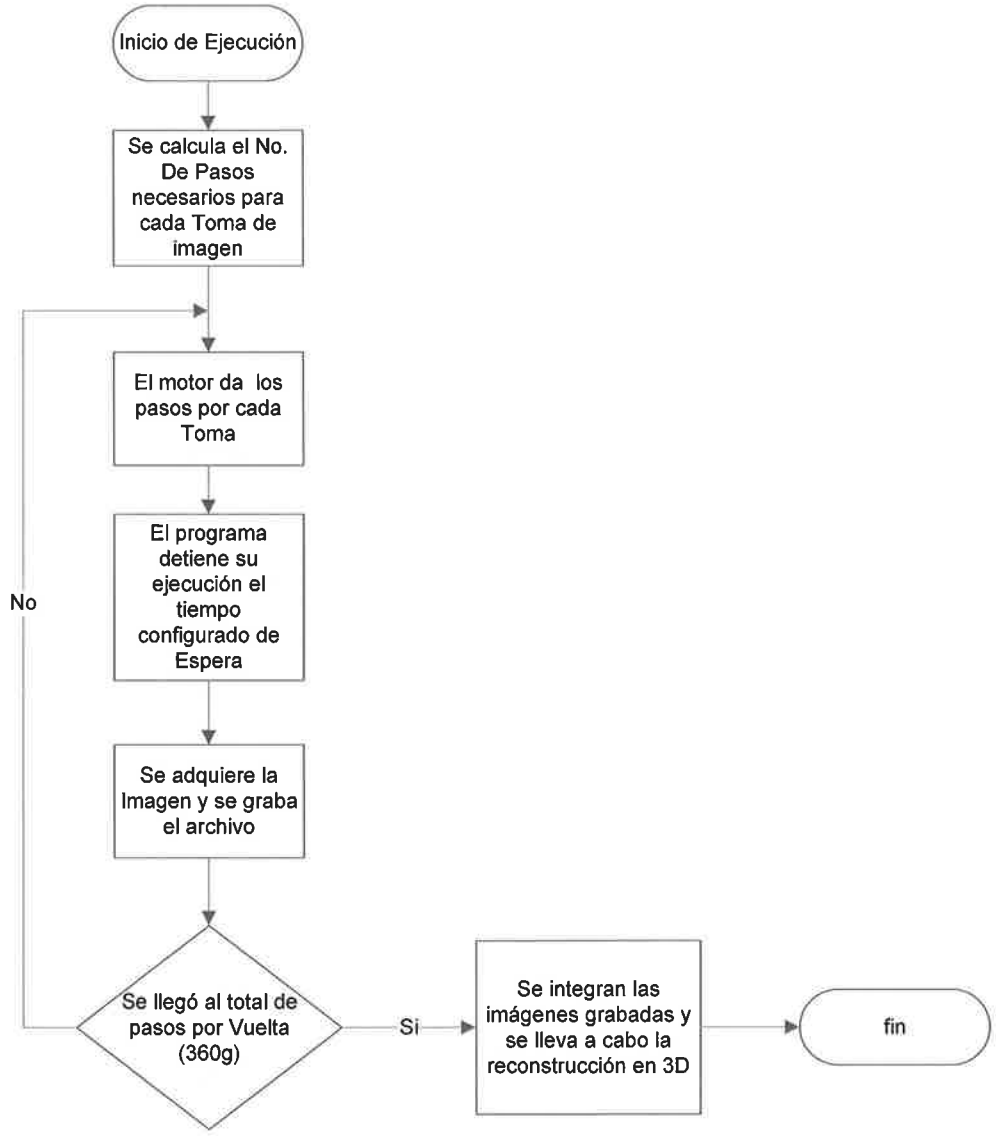


Figura 5.4.- Estructura modular del sistema

5.3.-Diagrama de Flujo del Sistema

El funcionamiento del sistema se lleva a cabo mediante el siguiente diagrama de flujo:



5.4.-Descripción del Sistema

La Figura 5.5 muestra la descripción del sistema real con todos sus componentes.

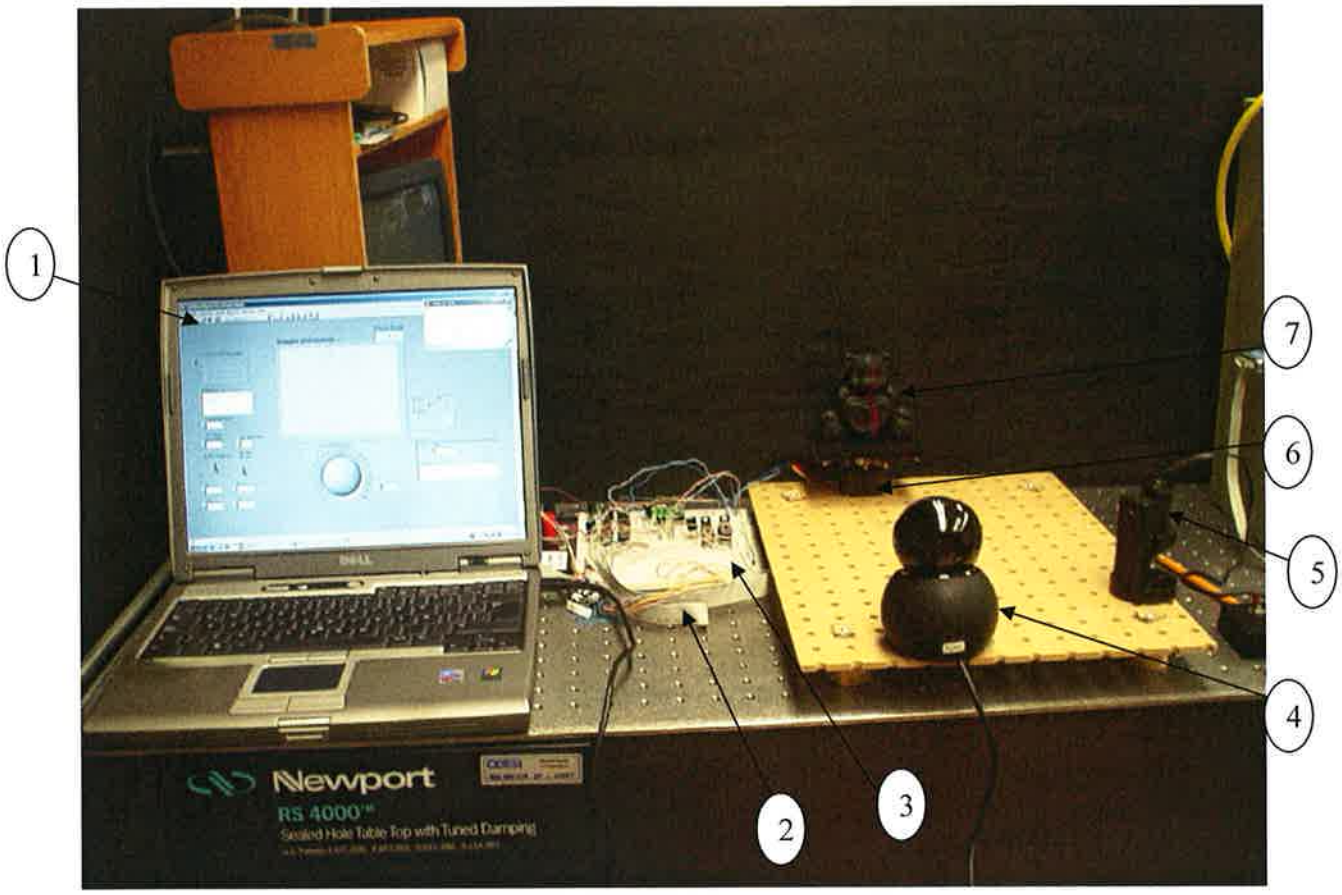


Figura 5.5.- Sistema y sus componentes

1. Interfaz del usuario
2. Cable Paralelo que comunica la salida de la PC con el control del motor
3. Circuito que recibe la información de la PC a través del puerto paralelo y controla el motor
4. Cámara USB que adquiere las imágenes
5. Láser que lleva a cabo la proyección de la línea en el objeto
6. Motor que es controlado mediante la PC y su circuito eléctrico correspondiente
7. Objeto al que se le hará la reconstrucción 3D

5.5.-Módulo de Control de Movimiento

El módulo de control de movimiento es el encargado de dirigir el movimiento y la velocidad del motor. Éste esta formado por:

- El motor.
- El circuito eléctrico que constituye el puente H para el control del motor.
- El cable paralelo, que sirve de interfaz de comunicación entre el circuito y la PC.

5.6.-Módulo de adquisición de imágenes

El módulo de adquisición de imágenes es el encargado, como su nombre lo indica, de adquirir las imágenes a través de la cámara USB.

Aunque no participa en la adquisición de imágenes, el láser que se proyecta en el objeto también pertenece a este módulo, ya que es el que origina o genera la imagen que nos interesa adquirir.

5.7.-Interfaz de usuario

La interfaz del usuario fue desarrollada en LabVIEW 7.1 y esta integrada por los siguientes controles e indicadores (ver figura 5.6).

5.7.1.-Controles de Entrada:

1. ***Path destino:*** Define el directorio y el nombre del archivo con el que se guardaran las imágenes. Al nombre del archivo se le agrega el consecutivo de imagen correspondiente y estas imágenes son grabadas en formato JPEG.
2. ***Tiempo de espera entre cada toma:*** Define el tiempo de espera entre el momento que termina de dar el paso el motor y el momento en el que la cámara adquiere la imagen. Este tiempo nos permite adquirir una imagen mas nítida, ya que sin este tiempo, el movimiento mismo del motor hace que el objeto vibre un poco y la imagen que se adquiere resulta un poco borrosa.

3. ***No. De pasos por vuelta del motor:*** Define el número de pasos que requiere el motor para dar una vuelta.
4. ***Grabar imágenes:*** Define si en la ejecución se grabaran las imágenes que se adquieran o no.
5. ***Total de imágenes a tomar:*** Define el número de imágenes a adquirir. En base a este dato se calcula cada cuantos pasos se tomará una imagen de tal manera que se tomen todas las imágenes durante una vuelta del objeto.
6. ***Velocidad del motor:*** Define la velocidad en que el motor dará los pasos.
7. ***Sentido de giro del motor:*** Define el sentido de giro del motor.

5.7.2.-Indicadores de Salida

8. ***No. De Imagen Actual:*** Muestra el numero de imagen que se esta adquiriendo actualmente.
9. ***Cámaras USB disponibles:*** Muestra la lista de cámaras USB detectadas y disponibles para su uso.
10. ***Imagen Adquirida:*** Muestra la imagen que se esta adquiriendo actualmente.
11. ***Estatus del proceso de toma de imágenes:*** Muestra el estatus del proceso de adquisición de imágenes, en caso de algún error se muestra tanto el código como la descripción del error en este indicador.
12. ***Estatus del proceso de reconstrucción 3D:*** Muestra el estatus del proceso de reconstrucción en 3D, en caso de algún error se muestra tanto el código como la descripción del error en este indicador.
13. ***Dirección de salida utilizada para el puerto paralelo:*** Muestra la dirección utilizada para el envió de información al puerto paralelo.

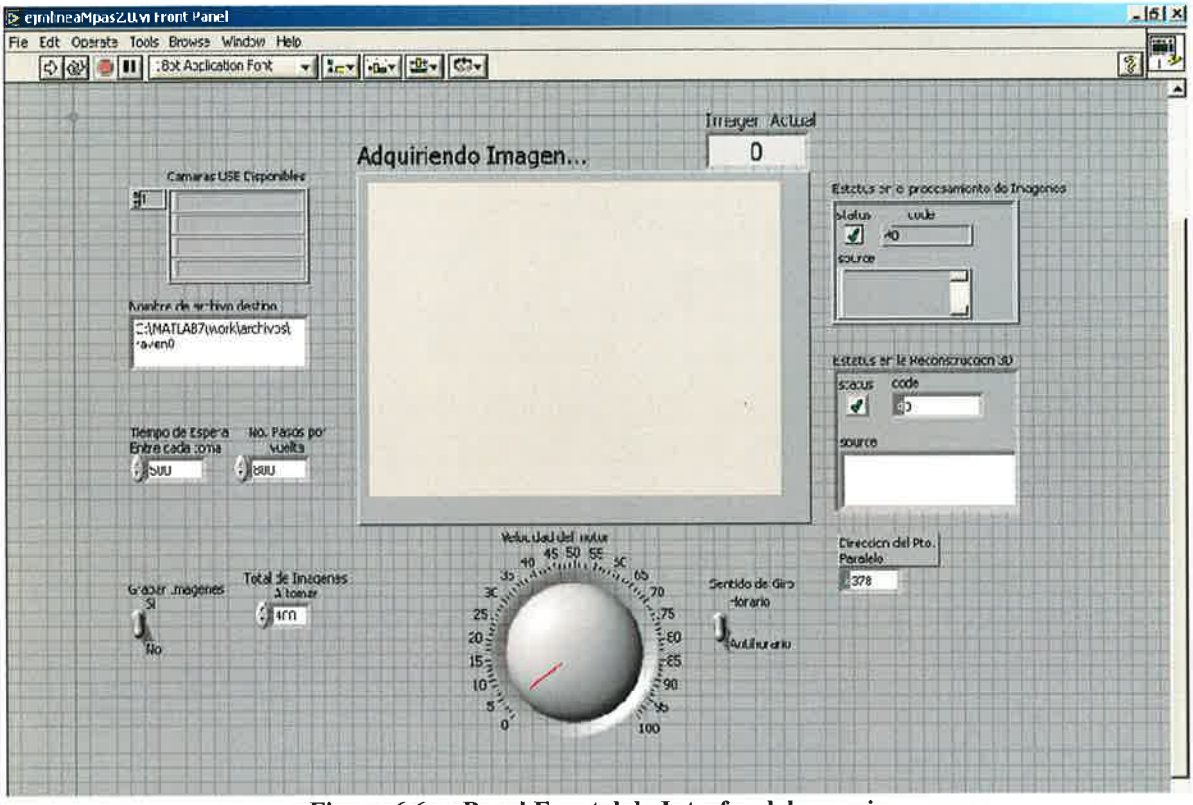


Figura 6.6.- Panel Frontal de Interfaz del usuario

En la Figura 5.7 y 5.8, se muestra los componentes y diseño de la interfaz del usuario en el diagrama de bloques.

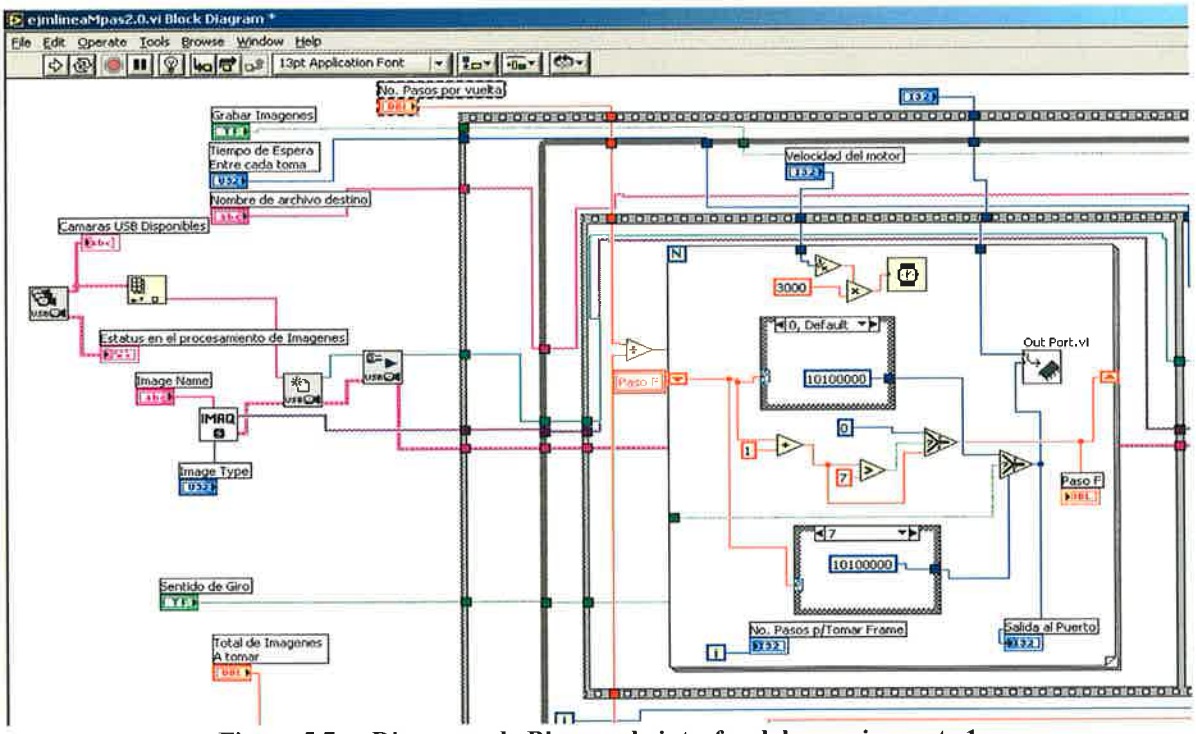


Figura 5.7.- Diagrama de Bloques de interfaz del usuario, parte 1

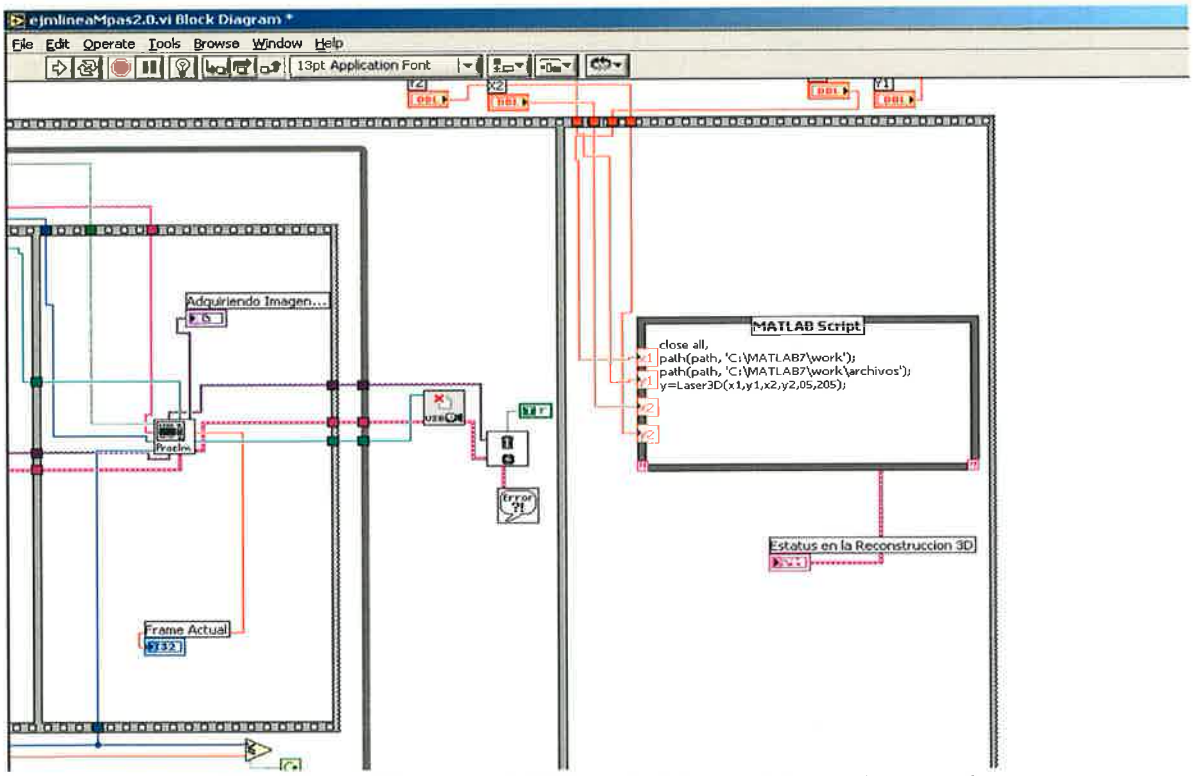


Figura 5.8.- Diagrama de Bloques de interfaz del usuario, parte 2

Capítulo 6. Resultados

Una de las interrogantes que se nos presentaron durante el proceso de las pruebas fue cual era la posición en la que se debía ubicar el dispositivo láser de tal manera que la información que pudieran proporcionarnos al adquirir las imágenes fuera de tal calidad que la reconstrucción del objeto en 3D fuera la mas aproximada.

Se llevaron a cabo varias pruebas ubicando el dispositivo láser en diferentes posiciones. La posición del dispositivo láser que hizo que las imágenes que se adquirieran generarán una mayor aproximación en la reconstrucción del objeto fue a un ángulo de aproximadamente 40 grados con respecto a la línea vertical imaginaria que se pudiera dibujar uniendo el objeto y la cámara USB.

En seguida se presentan algunas imágenes tomadas ubicando el dispositivo láser en diferentes posiciones.

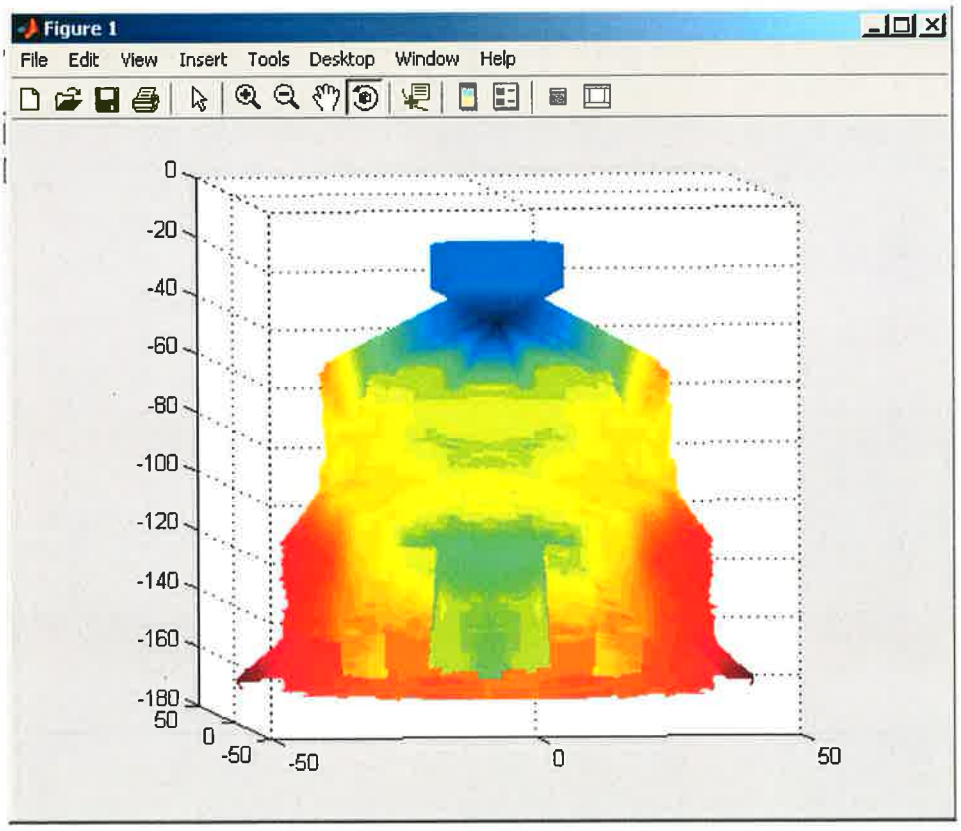


Figura 6.1.- Reconstrucción del objeto ubicando el dispositivo láser a 5 grados aprox.

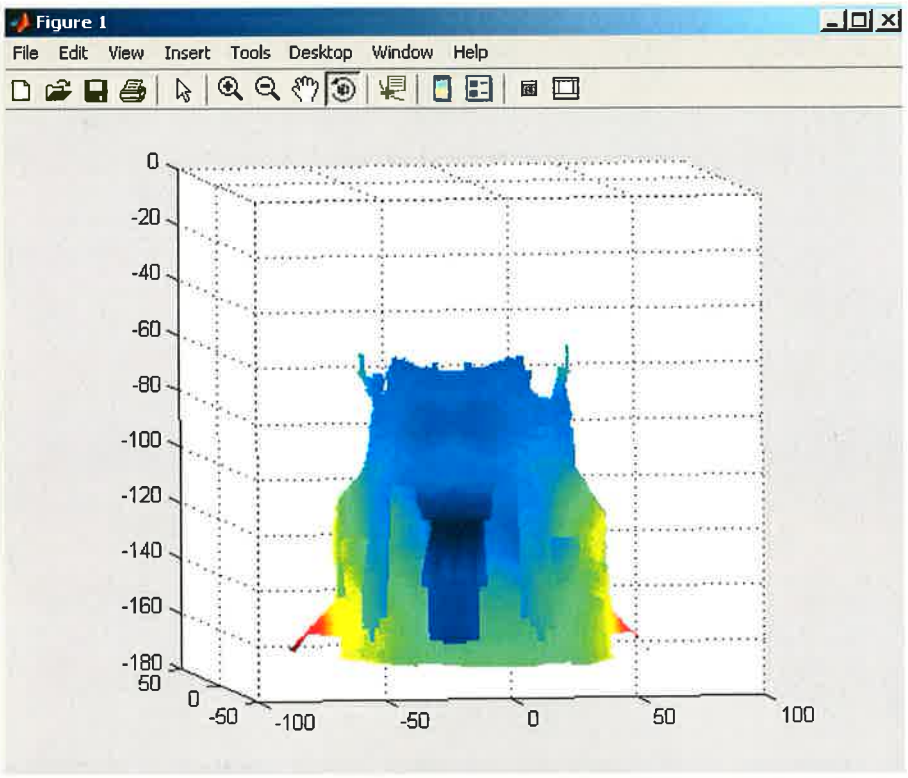


Figura 6.2.- Reconstrucción del objeto ubicando el dispositivo láser a 25 grados aprox.

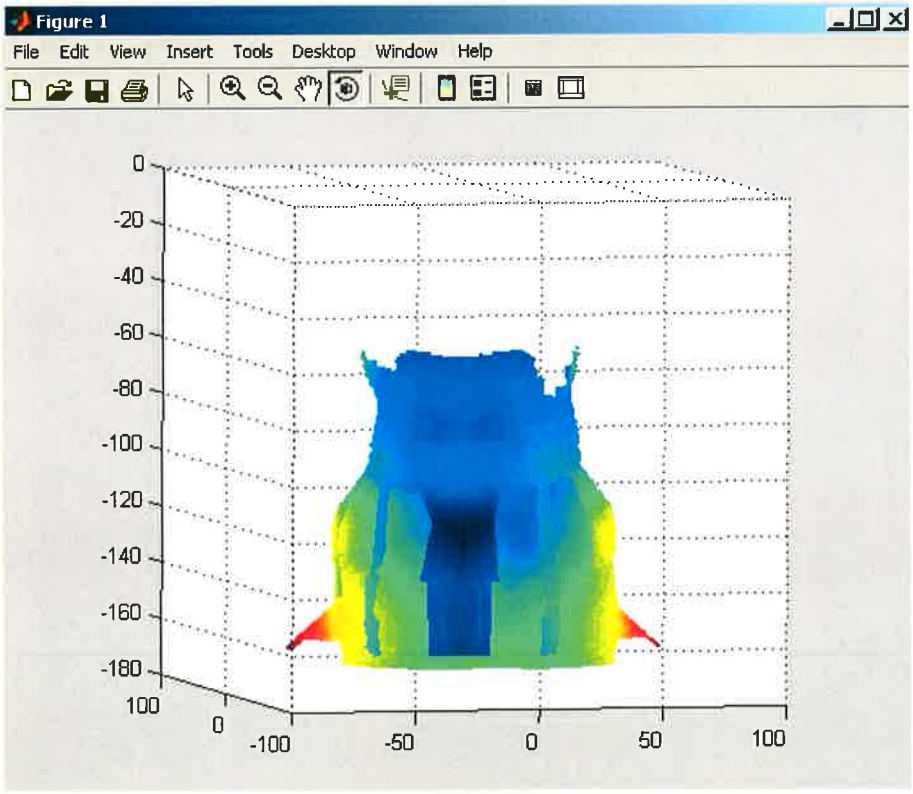


Figura 6.3.- Reconstrucción del objeto ubicando el dispositivo láser a 30 grados aprox.

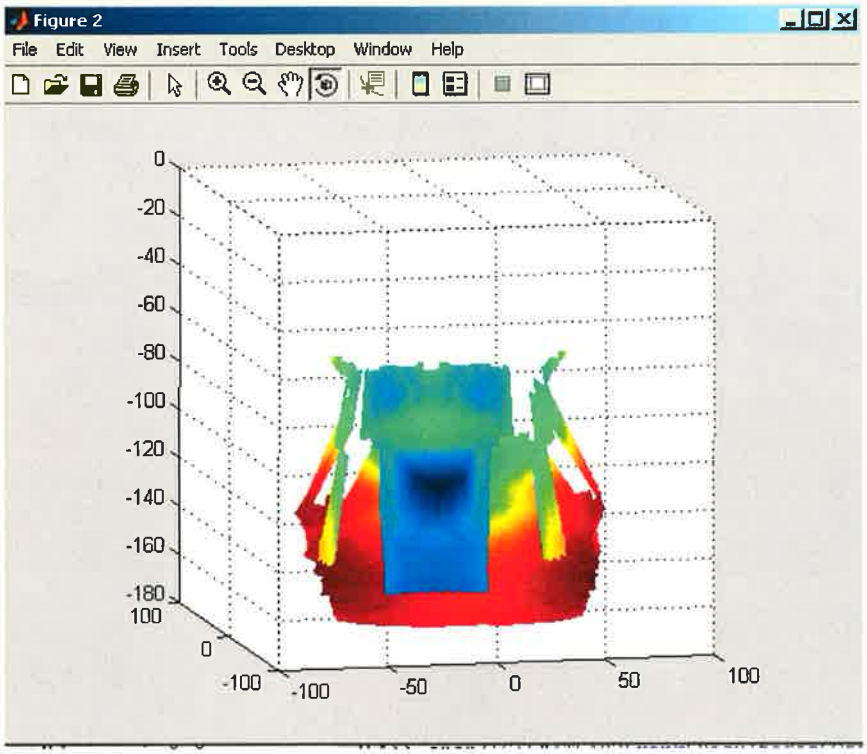


Figura 6.4.- Reconstrucción del objeto, ubicando el dispositivo láser a 45 grados aprox.

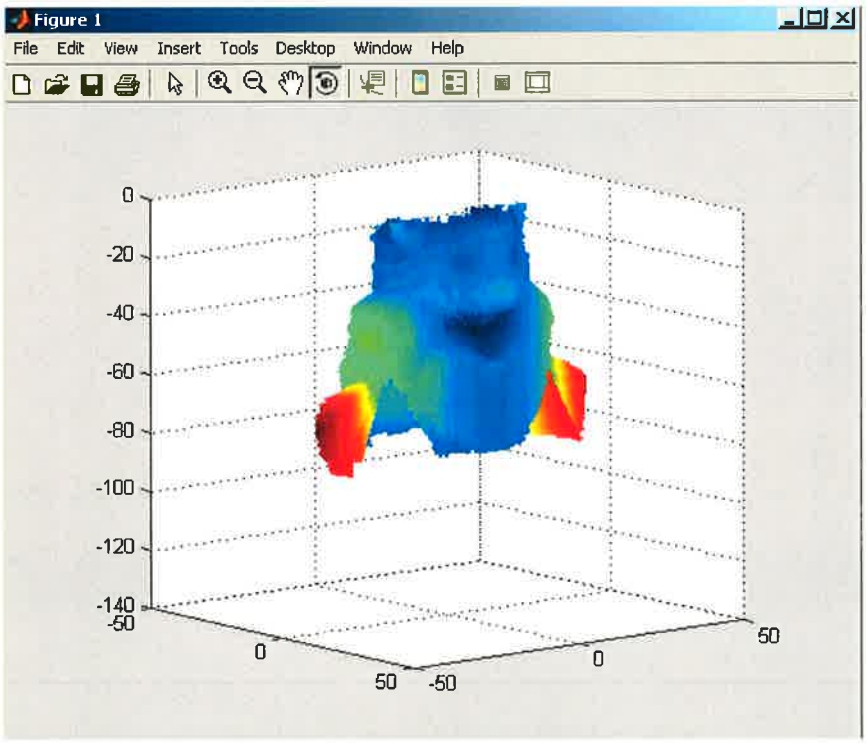


Figura 6.5.- Reconstrucción del objeto ubicando el dispositivo láser a 40 grados aprox.

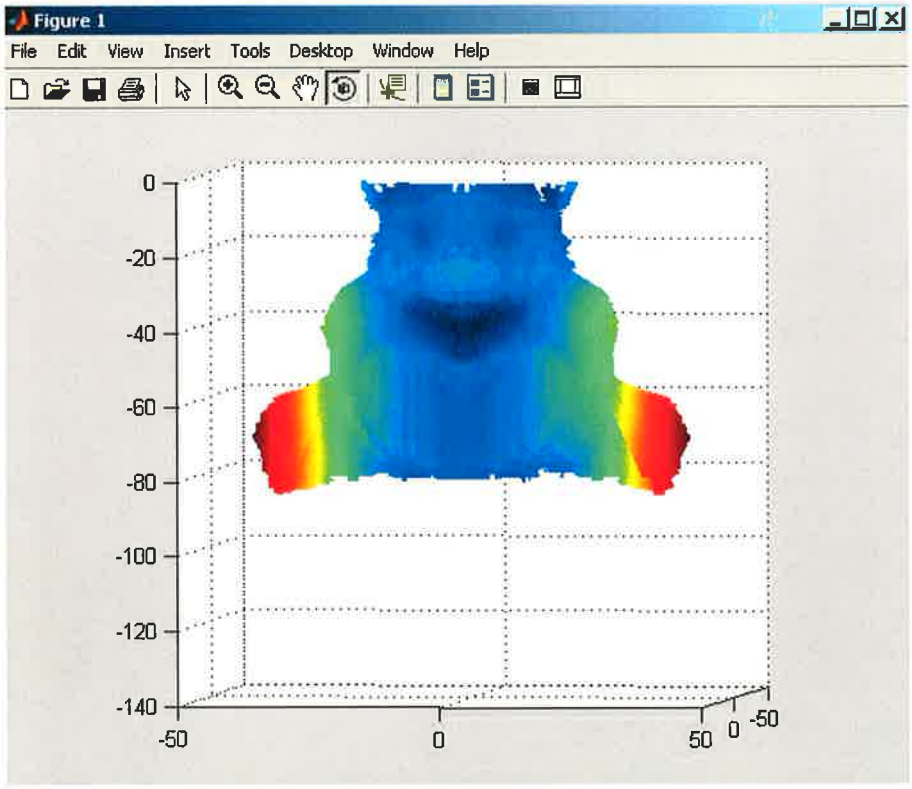


Figura 6.6.- Reconstrucción del objeto en 3D

Figura 6.7.- Objeto Real

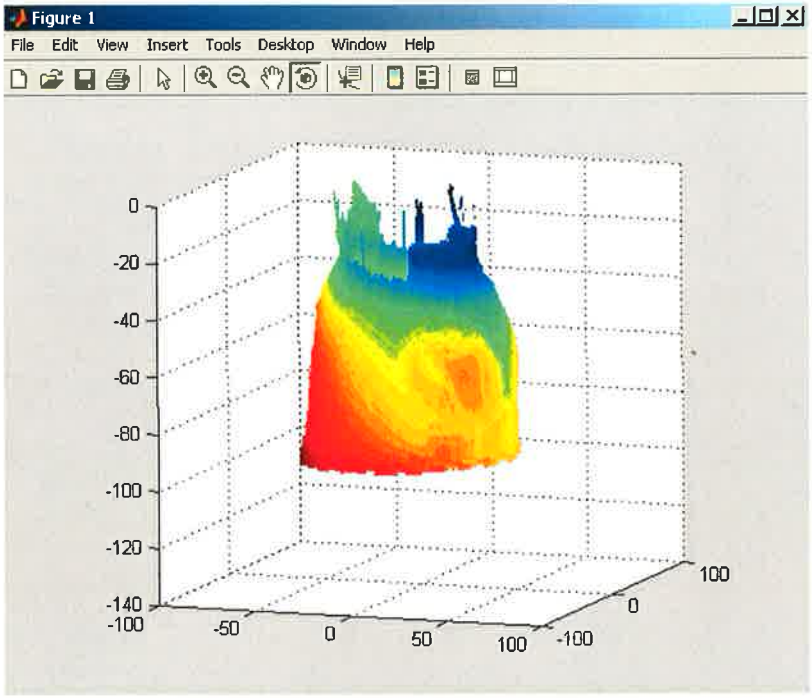


Figura 6.8.- Reconstrucción del Objeto

Figura 6.9.- Objeto Real

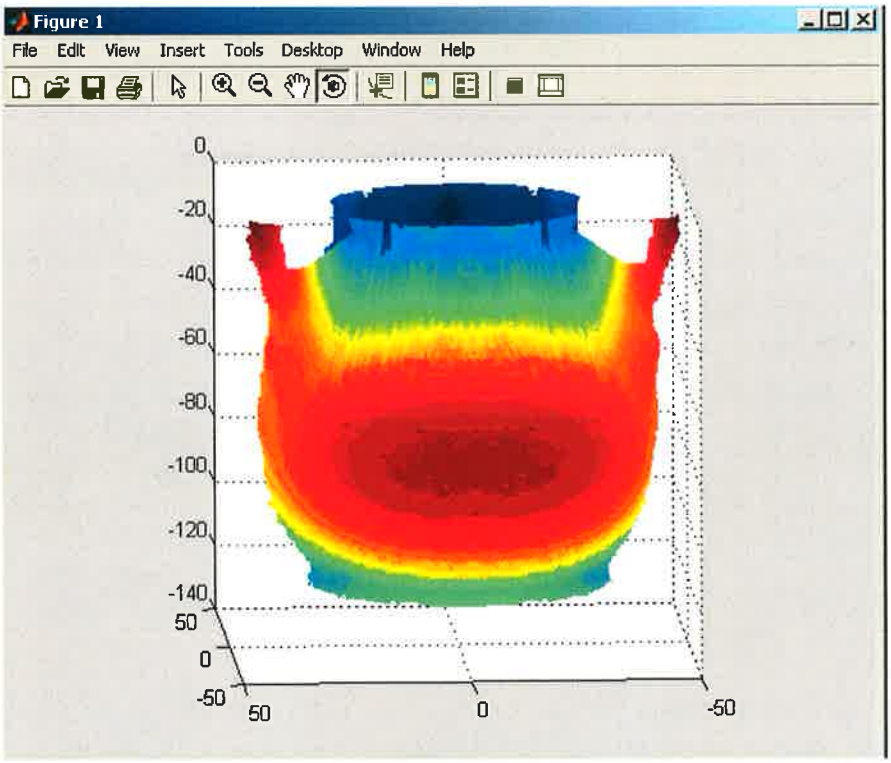


Figura 6.10.- Reconstrucción del Objeto

Figura 6.11.- Objeto Real

Otra interrogante que quisimos confirmar fue los resultados que obtendríamos al utilizar cámaras USB con distintas características; Los resultados mostrados hasta ahora fueron los que se generaron al utilizar la cámara 2 (ver características Anexo2); la imagen siguiente se reconstruyo utilizando la cámara 1(ver características Anexo 2).

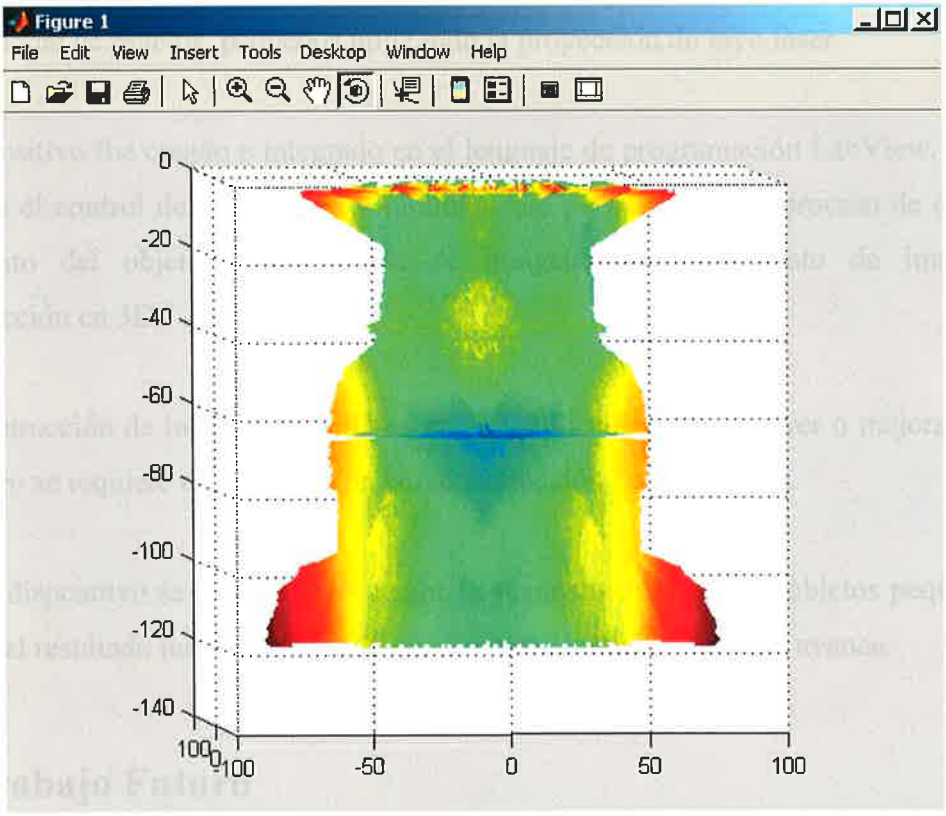


Figura 6.12.- Reconstrucción de objeto utilizando cámara 1

Como se esperaba, la reconstrucción de los objetos fue con mayor aproximación cuando se utilizo una cámara con mayor resolución que cuando se utilizó una con menos resolución (Ver Figura 6.12).

Es muy importante mencionar que antes de ejecutar el proceso para llevar a cabo la reconstrucción en 3D, se acomodaba la cámara de tal manera que la imagen que se estaba adquiriendo apareciera centrada.

Bibliografía

1. William Bolton, Mecatrónica Sistemas de control electrónico en ingeniería mecánica y eléctrica, Editorial Alfaomega, 2ª. Edición.
2. http://robots-argentina.com.ar/MotorPP_basico.htm
3. http://www.todorobot.com.ar/informacion/tutorial%20stepper/stepper-tutorial.htm
4. http://www.monografias.com/trabajos17/motor-paso-a-paso/motor-paso-a-paso.shtml
5. http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/hernandez_b_ii/capitulo3.pdf
6. Sebastián Salinas V. (Y MAS), Manual de Sensores, Motores y Controladores, Laboratorio de Mecatrónica - Departamento de Ingeniería Eléctrica - Universidad de Chile
7. www.monografias.com/trabajos5/**matlab/matlab**.shtml
8. http://www.tracnova.com/tracnova-pub/Qu%E9%20es%20LabVIEW.pdf
9. http://omega.ilce.edu.mx:3000/sites/ciencia/volumen2/ciencia3/084/htm/sec_9.htm
10. National Instruments, User Guide NI IMAQ for USB Cameras, Enero 2005, Part Number: 371492A-01.
11. National Instruments, IMAQ Vision for G Referente Manual, Enero 1999, Part Number: 322366A-01.
12. National Instruments, LabView User Manual, Enero 1998, Part Number: 320999B-01.
13. National Instruments, IMAQ Vision for LabView User Manual, Agosto 2004, Part Number: 371007A-01.
14. Visión por computador: imágenes digitales y aplicaciones, Gonzalo Pajares Martín Sanz y Jesús M. De la Cruz García, Alfaomega, 2002.
15. Hurtado Castañeda Delia Maria, Adquisición y Procesamiento de Imágenes en Labview, Estancia Técnica de investigación, Enero 2006.
16. Martín Díaz Guillermo, Motores, Curso 2005-2006
17. Pedraza-Ortega, J. C. Image Processing for Real World Representation Using Depth From Focus Criteria, Tesis de Doctorado, Universidad de Tsukuba, Japón, Marzo 2002.

Cámara 2

1. Cámara Web USB
2. Marca Logitech QuickCam
3.

Anexo 3.- Lista de materiales que se necesitan para armar un puente H

- 2 transistores NPN, TIP120 o en su defecto TIP31C (en este caso necesitaremos los diodos de protección)
- 2 transistores PNP, TIP125 o en su defecto TIP32C (en este caso necesitaremos los diodos de protección)
- 4 transistores pn2222
- 1 condensador 470 [μF]
- 2 resistencias de 47 [Ω]
- 2 resistencias de 470 [Ω]
- 2 resistencias de 3 [kΩ]
- 2 resistencias de 10 [kΩ]
- 4 diodos zener (van a depender del voltaje que ocupen los motores)