UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

T E S I S

"EMPLEO DE FPGA's PARA EL SISTEMA DE CONTROL DE MOTORES DE PASOS BIMODALES EN MICROMECÁNICA"

PRESENTAN:

CARLOS ALBERTO MUÑOZ LEINES

CÉSAR AUGUSTO SANTOS CARRASCO

PARA OBTENER EL GRADO DE

INGENIERO ELÉCTRICO Y ELECTRÓNICO
(AREA ELECTRÓNICA ANALÓGICA)

DIRECTOR: **DR. ERNST M. KUSSUL**

CO-DIRECTORES: **M.I. ALBERTO CABALLERO RUÍZ**
M.I. LEOPOLDO RUÍZ HUERTA

CIUDAD UNIVERSITARIA, MÉXICO, D. F., JUNIO 2003

AGRADECIMIENTOS

A nuestra querida U.N.A.M. que nos dio la oportunidad de un desarrollo profesional y productivo para México.

A la Facultad de Ingeniería, que por medio de sus profesores y compañeros nos transmitieron sus innumerables conocimientos y experiencias.

Al Laboratorio de Micromecánica y Mecatrónica del CCADET, que junto con su infraestructura y amigos que encontramos, logramos este trabajo.

Al Dr. Ernst Mikhailovich Kussul con quien fue un honor haber podido trabajar, aprendiendo de su gran experiencia y calidad humana.

Al M.I. Leopoldo Ruiz Huerta y al M.I. Alberto Caballero Ruiz; que con la invaluable amistad, conocimientos y consejos nos apoyaron en todo momento en el desarrollo de la misma. Gracias

A la Dra. Graciela Velasco, quien siempre nos apoyó en nuestro buen desarrollo académico, y por ser una excelente amiga.

Al Sr. Mario Rodríguez, por asesorarnos y guiarnos en las labores mecánicas propias del proyecto.

A nuestros sinodales que gracias a sus aportaciones enriquecieron aun más esta tesis.

A CONACyT, por creer en nosotros y apoyarnos durante el desarrollo y el financiamiento del proyecto 33944-U.

... a Dios por haberme dado vida y permitirme concluir uno de mis más grandes anhelos.

... a mi querida madre por darme su amor incondicional incomparable, su apoyo familiar, moral y económico, además de sus grandes consejos que nunca me han faltado; por eso y por mucha mas le dedico esta tesis, que significa el mayor triunfo que he podido realizar en la vida... TE AMO.

... a mi padre, que aunque no este hoy conmigo en vida, siempre me motivo por ser el mejor de las filas... ¡Te cumplí Pa! †

... a mis hermanas Ingrid y Nancy, que siempre me han forjado una conciencia formativa para ser un hombre de provecho... Maestras, ¡las quiero!.

... a Gabo que en esos momentos tan difíciles en la facultad, que me hizo sonreír cuando más lo necesitaba; por su comprensión al molestarme debido a su inocencia al romper mis proyectos, por eso quiero que mi ahijado vea esto como un logro suyo, para que lo tome de estímulo para que siga poniendo empeño en sus estudios.

... al profesor Arnulfo que estuvo conmigo apoyándome toda la carrera, y que gracias a su forma de ser, pude ver que la familia es primero y que se dicta en el corazón.

... a Alejandra por el apoyo brindado en los momentos más difíciles de esta tesis, que junto con su alegría y amor, me ha hecho vivir al máximo, anhelando que nunca se aparte de mí.

... a la familia Santos Carrasco, que con su gran amistad siempre me apoyaron sin condiciones, así mismo por sus consejos de cómo ser una buena persona.

... a mi querida familia, encabezada por mi abuelita Gloria, que siempre me apoyo desde niño, y que junto con mis tías, tíos y primos hacemos a los Leines.

... a Ari, Toñito, Isrrra y Erving, que me apoyaron sin condiciones en las buenas y en las malas, dentro y fuera de la facultad, para que forjáramos una amistad duradera. SUERTE CUATES!!!

... a todos mis queridos amigos que llevo en el alma, Noé, Ubaldo, Víc, Marín, Nico, Nelly, Ruth, Irene, Cholo, César, Yaz, Oli, Martha, Kari, Lucy, Ulises, Emerson, Fer, Sergio y aquellos que no nombre, pero están conmigo siempre.

... a Cesarín mi gran brother, que junto con él pude concluir con empeño y entusiasmo mi gran sueño, Ok! my Jedi. The force by with you, always.

Carlos Alberto

AGRADEZCO

A mi madre, por tu gran fortaleza, amor incondicional, apoyo permanente, dedicación y cariño. Tus palabras son mi refugio y tu mirada protectora mi horizonte. Te agradezco que me hayas enseñado a levantarme ante la eventualidad, a gozar la felicidad y a aprender de los errores. Siempre serás un ejemplo a seguir durante toda mi vida. Te quiero mucho.

A mi padre, gracias por tu amor y cariño, pero sobre todo, por apoyarme en este camino elegido así como acompañarme en mi recorrido siempre. Gracias mi viejo.

A mi hermana, quien con su cariño y apoyo, supo aconsejarme con gran inteligencia y madurez; y por toda la alegría que compartimos.

A Gaby, por toda tu paciencia, compresión, amor y compañía que me has dado a manos llenas. Fue maravilloso estar contigo durante la carrera. Gracias por tu amor y deseo de compartir un camino juntos.

A la familia Mendieta Rebollo, por darme la oportunidad de llegar a fraternizar con ellos en las buenas y en las malas.

A Carlos, por permitirme construir contigo un trabajo digno pero sobre todo por tu amistad que es de esas que trasciende el tiempo y la distancia. Gracias mi joven Padawan.

A la familia Muñoz Leines, quienes siempre apoyaron las ideas revolucionarias de estos dos locos.

A la Comisión Nacional de Ahorro de Energía, en donde encontré mas que amigos y donde se me apoyo con una beca durante mi estancia.

A mi U.N.A.M., mi Alma Mater por su gran generosidad, por que aquí aprendí a valorar las cosas en su justa dimensión y encontré a muchas personas valiosas con las que pude compartir bellos y divertidos momentos.

Al maestro Yoda, de quien siempre supe escuchar sus excelentes y sabios consejos.

Y a quienes omití, no por olvidar sino por que en realidad nunca terminaría esta gran lista, pero saben que siempre están presentes en lo mas profundo de mis sentimientos.

César Augusto

CONTENIDO

CAPÍTULO 3

ARREGLO DE COMPUERTAS PROGRAMABLES DE CAMPO **25**

CAPÍTULO 4

IMPLEMENTACIÓN DEL FPGA CON LOS MOTORES BIMODALES **41**

CAPÍTULO 5

INDICE DE FIGURAS Y TABLAS

Tabla **Página**

INTRODUCCIÓN I

Se describe brevemente el desarrollo del proyecto, interviniendo el empleo de FPGA's para el control de motores de pasos utilizados en micromecánica.

La industria que existe hoy en día, requiere de sistemas compactos de alta eficiencia, bajo consumo de espacio y un gasto energético mínimo. Dadas estas necesidades en la Universidad Nacional Autónoma de México (UNAM) se desarrolla micromecánica, que con herramientas convencionales se llegue a un prototipo en donde reúna las expectativas antes mencionadas.

Estos prototipos, pertenecen a una primera generación, construidos en el Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET), de particular interés desarrollado en el Laboratorio de Micromecánica y Mecatrónica (LMM), encabezado por el Dr. Ernst Kussul.

Este prototipo realiza procesos de maquinado por arranque de material. Trabaja como torno, fresadora y taladro; así entonces, se planea desarrollar una segunda generación con micropiezas elaboradas por su generación antecesora.

El presente trabajo esta divido como a continuación se detalla:

El *primer capítulo* menciona un marco tecnológico en el ámbito mundial sobre la micromecánica, así como el papel que desarrolla México, planteando algunos fundamentos y principios de técnicas

convencionales usadas en el desarrollo de microequipo. También, en este apartado se mencionan las necesidades, los recursos y el objetivo primordial que se busca alcanzar con la investigación.

El *segundo capítulo* establece los aspectos y conceptos fundamentales de un motor de pasos, tipos de control y sus aplicaciones.

El *capítulo tercero*, introduce el concepto de FPGA, desde sus antecedentes hasta la importancia de pertenecer a un grupo de lógica configurable, así mismo, se describen las características de un dispositivo de lógica programable de la compañía Altera, su forma de uso y programación mediante su software.

El capítulo cuarto describe la realización del proyecto, donde aplicando un FPGA en una nueva forma de controlar motores de pasos, se obtiene una mejora en el desarrollo de microequipo, mostrando el tipo de control y el diseño electrónico utilizado.

El *capítulo quinto* describe los resultados obtenidos durante el desarrollo del presente proyecto, a través de pruebas que se hicieron a los motores pasos, que se denominaron bimodales.

Por último en el apartado dedicado a *conclusiones y trabajo a futuro* se analizan los resultados obtenidos del proyecto, y se proponen los siguientes pasos para la continuación de esta línea de investigación.

MOTORES DE PASOS EN MICROMECÁNICA

Capítulo 2

En este capítulo se establecen conceptos fundamentales de los motores de pasos así como sus tipos de control, y el funcionamiento del motor bimodal, empleado en micromecánica.

MOTORES DE PASOS

Un motor de pasos o también llamado motor a pasos es una máquina eléctrica cuyo eje (flecha) gira en un ángulo preciso al que se le denomina paso, debido a la aplicación de un conjunto determinado de pulsos eléctricos. Este posicionamiento está determinado por el número de bobinas que tenga el motor; el motor de pasos es un dispositivo que convierte pulsos digitales en movimiento mecánico de rotación.

PARTES DE UN MOTOR DE PASOS

Un motor de pasos es generalmente dividido en dos partes básicas que son: rotor y el estator. (Figura 2.1) El rotor esta constituido por un imán permanente o una pieza dentada de material magnético, dependiendo del tipo de motor; en tanto, el estator está constituido por un número de embobinados que rodean al rotor, basados

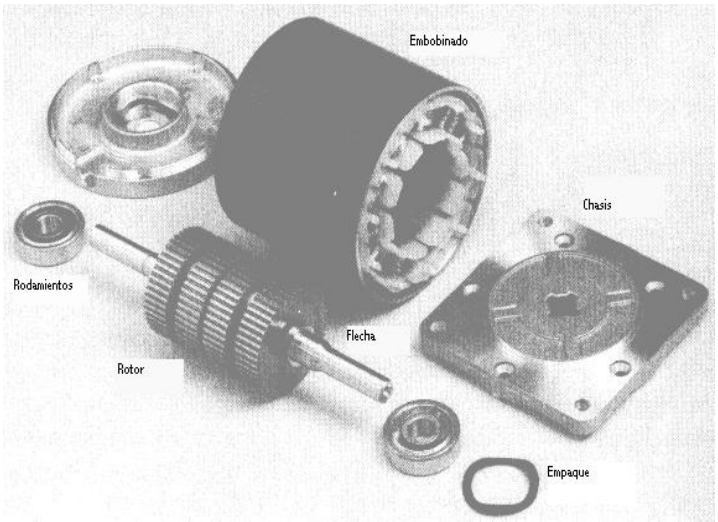


Figura 2.1 Partes principales de un motor a pasos

en el principio de atracción y repulsión de los polos magnéticos. Al aplicar una corriente en el embobinado del motor, obliga al estator a girar hasta una posición de reposo, la cual cambia al modificar la energización de los embobinados.

VENTAJAS DE LOS MOTORES DE PASOS

Podemos decir que los motores de pasos tienen varias ventajas como su alta rentabilidad, estructura simple y fuerte, pequeño tamaño y mantenimiento mínimo, como se muestran en la figura 2.2.

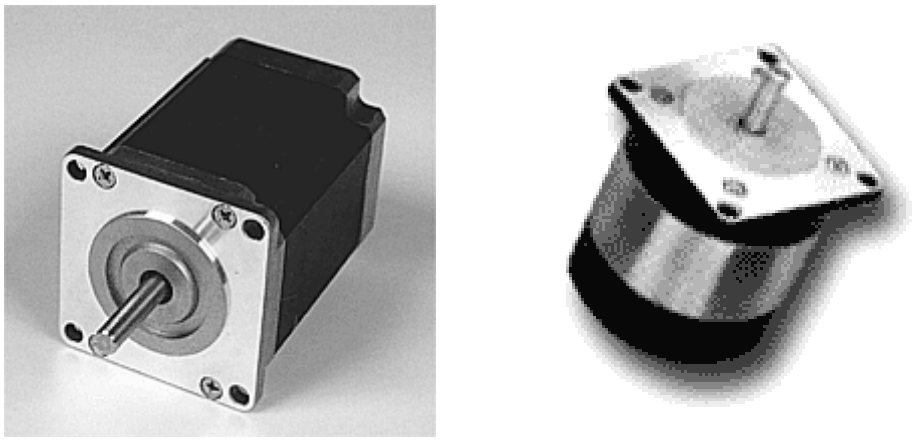


Figura 2.2 Motores de pasos

Los motores de pasos son muy precisos y confiables por lo que se emplean comúnmente donde se requiere una gran precisión en el posicionamiento, gracias a su par estático, el eje conserva la posición deseada hasta que se envía una nueva orden para hacerlo girar nuevamente, además, poseen un elevado par en bajas revoluciones, permitiendo un bajo consumo de energía en su rango de operación.

LIMITANTES DE LOS MOTORES DE PASOS

No son muy rápidos en términos de revoluciones por minuto, decrementan su torque al aumentar su velocidad, presentan resonancia en velocidades bajas y la excitación de los embobinados debe darse de una forma precisa.

Los motores de pasos son controlados por medio de la cantidad de pulsos enviados, de manera que su velocidad y su sentido sean operados mediante su circuito electrónico; el sentido se modifica invirtiendo la secuencia de pulsos, y la velocidad varia de forma directa con la frecuencia de defasamiento de los pulsos, ocasionando la necesidad de un circuito de control por cada motor de pasos.

MODO DE FUNCIONAMIENTO DE LOS MOTORES DE PASOS

Como se explicó con anterioridad, el motor de pasos es un dispositivo que convierte una entrada digital en movimiento mecánico, dependiendo del número de bobinas que tenga será el número de señales (trenes de pulsos); los cuales estarán defasados 90° para que exista un movimiento cíclico del rotor entre ellos, ya sea en sentido horario o antihorario; enviando esta señal, a la etapa de potencia que regula la corriente para cada una de las bobinas.

Cabe destacar que la velocidad de un motor de pasos es controlada por la frecuencia que se le aplique mediante un dispositivo electrónico, llamado controlador digital, en donde tendrá sus límites de velocidad a causa de las características del motor.

Es importante la parte física del motor, en especial la del embobinado con el rotor que fue fabricado de material permanentemente imantado y es magnetizado en forma radial, dado que su finalidad principal es la precisión que este desarrolle, es decir, que se mueva tan rápido como le sea posible en respuesta a un impulso de entrada; y no solo para el arranque sino también para su frenado en el momento que se le ordene.

CLASIFICACIONES DE LOS MOTORES DE PASOS

Básicamente se pueden catalogar a los motores de pasos en los siguientes tipos:

- Reluctancia variable
- Imán permanente (unipolares y bipolares)
- Híbridos

Motor de reluctancia variable

Un motor de reluctancia variable (figura 2.3) consta de varias bobinas donde estas poseen un cable de alimentación común de un extremo a cada bobina y es empleado en aplicaciones que no requieren un gran par.

En la figura 2.3 se muestra este tipo de motor, el cuál esta compuesto de tres fases con seis dientes en el estator, el acoplamiento se encuentra a 180°, perteneciendo a la misma fase, cada uno con una polaridad opuesta. Al estar energizados estos pares de dientes son los que originan el juego de polos norte y sur, en un estado en el que existan cambios de posición de campo magnético, y así causa el giro del rotor.

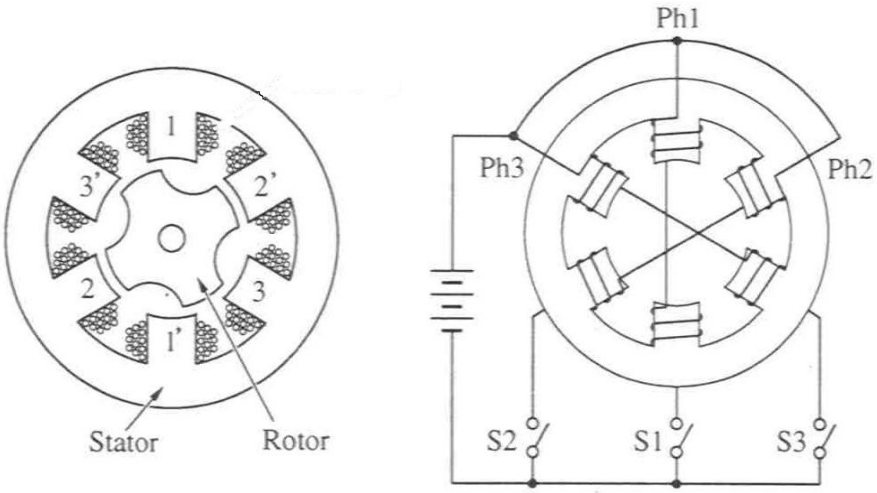


Figura 2.3 Corte transversal de un motor de reluctancia variable

Motor de imán permanente

Este tipo de motores consiste como su nombre lo indica, en que el rotor esté conformado por un imán permanente. Esto puede ocasionar que opere a velocidades bajas, pero la ventaja es que el motor a imán permanente alcanza un par relativamente alto, además de un costo muy bajo, siendo ideal para cuando se desea tener varios motores en alguna producción en serie. Es el más sencillo de los motores de pasos, su estator tiene polos salientes, mientras que el rotor sea de forma cilíndrica; como el mostrado en la figura 2.4.

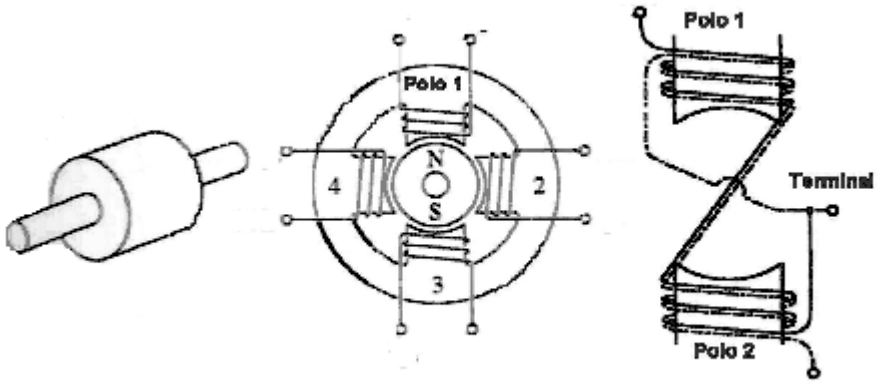


Figura 2.4 Conexiones de un motor de imán permanente

Motores unipolares y bipolares

Una característica principal de los motores unipolares es tener un "tap central", el cual generalmente se conecta a la alimentación mas positiva. Esta alimentación se realiza mediante algunos cables, donde forzosamente se necesitan dos para cada embobinado, se podría decir que algunos cables pueden estar conectados internamente para varios embobinados y otro para la su alimentación. Gracias a la estructura de este tipo de motores, se permite operaciones a altas velocidades teniendo una desventaja, que cuando se llega a revoluciones altas decrementa su par.

Los motores bipolares son los que no tienen "tap central". Están diseñados en tener dos bobinas independientes cuyas polaridades necesitan ser invertidas en cada fase de manera que se pueda hacer el campo magnético para que se logre girar el rotor. Sólo tienen cuatro conexiones, dos para cada bobina.

Cabe recalcar que este tipo de motores de pasos bipolares se fabrican en el LMM, obteniendo muy buenos resultados en tamaño, par, velocidad y costo, como se muestra en la figura 2.5.

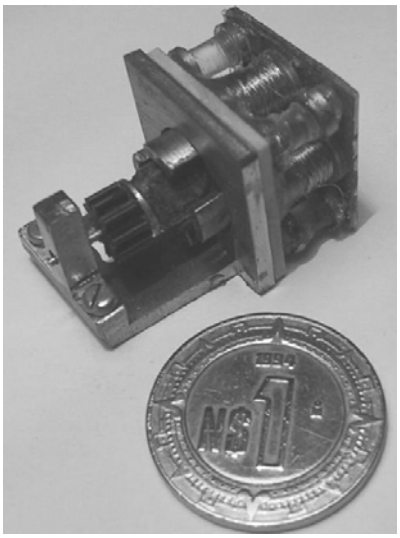


Figura 2.5 Motor de pasos diseñado en el LMM

Motores híbridos

Estos motores contienen la tecnología de los dos tipos de motores mencionados con anterioridad, dado que combinan algunas características de los motores de reluctancia variable y de los motores a imán permanente, como se muestra en la siguiente figura 2.6.

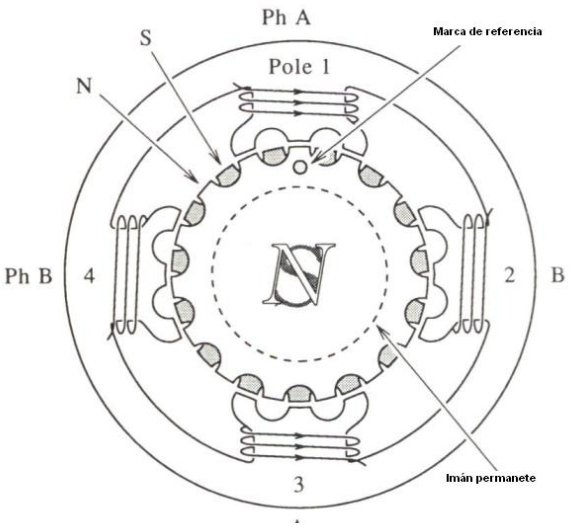


Figura 2.6 Corte transversal de un motor híbrido

En este tipo el rotor, se utiliza un imán permanente polarizado, cubierto de otro cilindro cerrado, cuenta con un alto par estático y dinámico, alcanzando altas velocidades; por lo que los motores híbridos son empleados en una gran variedad de aplicaciones industriales.

CONTROL DE MOTORES DE PASOS

Es necesario contar con un circuito electrónico capaz de convertir pulsos de reloj en señales de control para energizar las bobinas, además de una etapa de potencia, a la salida de dicho circuito de control que entrega la suficiente corriente a las bobinas del motor.

Cuando se quiere conectar un motor con una computadora, basta con la etapa de potencia dado que la computadora, a través de uno de sus puertos (bien sea paralelo, serie o USB), se encarga mediante un programa, enviar la señal en forma secuencial.

Para controlar un motor de pasos por un puerto de salida de la computadora, se asume que la resistencia de la bobina es suficiente para limitar la corriente que una etapa de potencia le proporcione.

En la actualidad es muy común para controlar motores de pasos, utilizar circuitos integrados en proporción al número de

bobinas, aunque no son indispensables, son muy importantes cuando se quiere obtener un espacio reducido en la etapa de potencia.

Un circuito capaz de controlar un motor de pasos puede ser el mismo para varios motores, únicamente diferenciándose en la lógica configurable programable. La elección del circuito controlador dependerá de las condiciones en las que se desea someter al motor, para su velocidad y dirección.

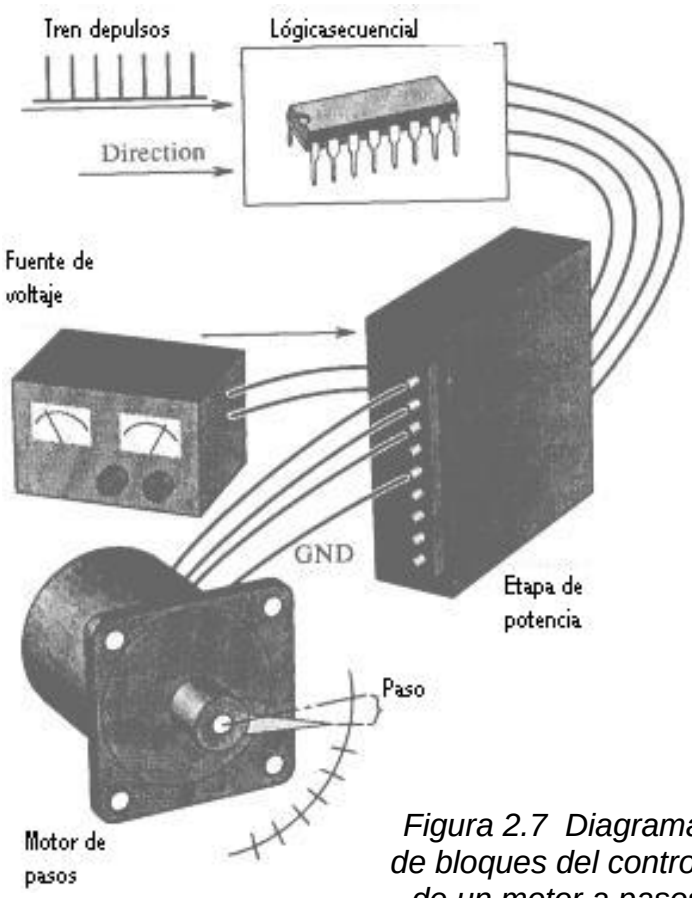


Figura 2.7 Diagrama de bloques del control de un motor a pasos

Esto quiere decir, que un sistema completo de control de un motor de pasos, como se ve en la figura 2.7. Incluye tres elementos básicos:

- El sistema de control par emitir los pulsos
- La etapa de potencia y
- El motor de pasos,

Un dispositivo lógico programable es capaz de generar los pulsos y el defasamiento de estos, dado que la etapa de potencia energiza las bobinas del motor donde usualmente se usan transistores tipo Darlington para soportar la corriente que se requiere.

Para generar las trayectorias de control para los motores de paso, por lo regular se establece una síntesis directa de estas, con la ayuda de la lógica discreta, que se realiza generalmente con ayuda de

circuitos flip-flop's tipo D o J-K. Aunque exista una gran cantidad de empresas fabricantes de circuitos de control para motores de pasos que incluyen una lógica más complicada y permiten el control en paso completo y medio paso, o que incluso algunos integran la etapa de potencia, algunas de estas soluciones son válidas para diversas aplicaciones. Actualmente, ya existen circuitos integrados controladores de interfaces programables, que permiten remplazar a la computadora y así tratar a la señal del modo requerido.

Algunas ventajas de un sistema de control es que comparadas con otros mecanismos pueden llevar a cabo funciones iguales o similares, no es requerida la retroalimentación para el control de posicionamiento, dirección y velocidad, los errores de posición no son acumulativos y la compatibilidad con equipos digitales modernos pueda ser excelente.

TIPOS DE CONTROL

Dicho anteriormente el control de un motor de pasos, es un sistema que provee la información necesaria para emitir el número de pasos que indica su frecuencia y su dirección. Muchas aplicaciones requieren que este sistema maneje otras funciones, tales como la aceleración, la desaceleración y el número de pasos por segundo.

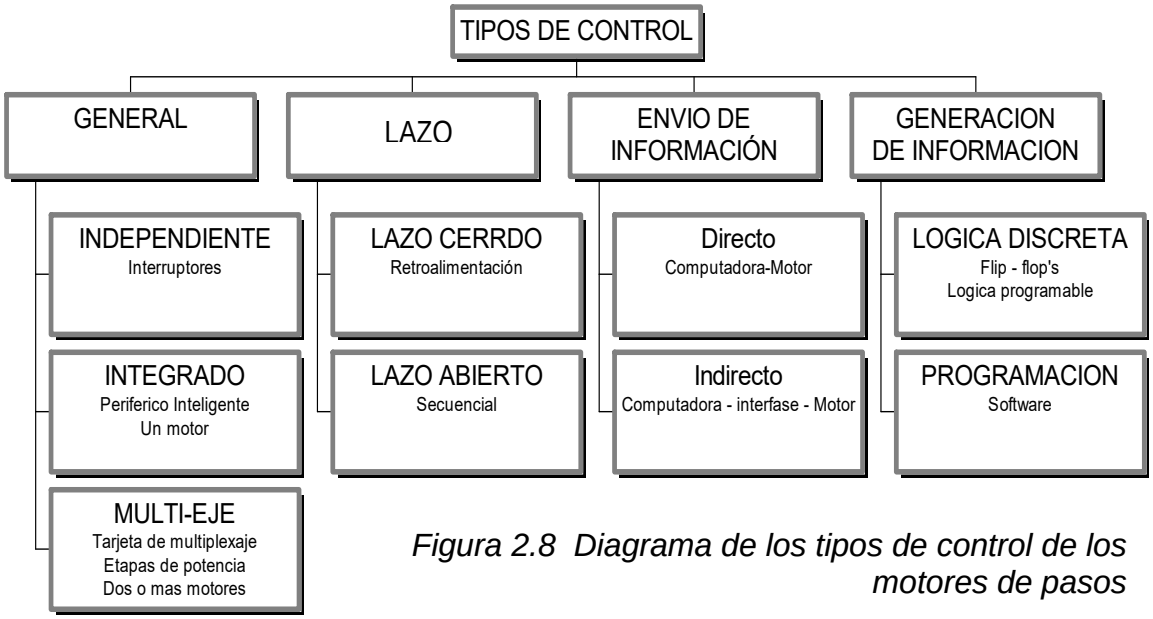


Figura 2.8 Diagrama de los tipos de control de los motores de pasos

La computadora es una ayuda excelente dado que su microprocesador ofrece una gran flexibilidad y rapidez, logrando recibir comandos de alto nivel de una señal emitida mecánicamente, además de generar los pulsos necesarios para determinar el número de pasos y la dirección de giro.

Control por implantación física

Existen tres principales formas de control si se toma el sistema completo de control:

- Independiente
- Integrado
- Multi-eje

El *control independiente* opera de manera autónoma, independiente de cualquier computadora. Los programas con la secuencia de movimiento pueden ser proporcionados por distintos tipos de interfase, tales como teclados, interruptores, etc., una vez que éstos hayan sido guardados en algún tipo de memoria. El control independiente normalmente va acompañado de la etapa de potencia, y de una fuente de alimentación, o incluso de un sistema de retroalimentación.

El *control integrado* señala que el motor recibe las instrucciones de una computadora, que procesa la información que controla los pasos que realiza el motor.

Cuando se desea controlar más de un motor, resulta viable emplear un sistema de *control multi-eje*, dado que este sistema de control tiene una tarjeta controladora y la etapa de potencia para cada motor. Además, si se demanda un alto grado de sincronización en varios motores el procesador central coordinando cada movimiento de ellos y en caso de no existir movimiento simultáneo, cada uno de los motores se movería a la vez con su debida rutina, combinando los pulsos y la dirección.

Control directo e indirecto

Los sistemas de control para un motor de pasos pueden ser directos e indirectos, el directo produce la secuencia de control y la transmite directamente por el puerto de la computadora hacia los transistores que controlan los embobinados del motor. Lee la posición real del motor guardada en un registro y se compara con la posición deseada, guardada en otro registro. Si el eje se encuentra en posición, no se realiza ningún cambio, pero en el caso contrario, se activa una interrupción la cual procesa la secuencia de control para cada uno de los embobinados en la dirección requerida y para un solo paso; mientras tanto el control indirecto, envía la secuencia de pulsos a través del puerto de la computadora a una tarjeta que sirve de interfaz, quien acciona las bobinas del motor.

Control a lazo abierto y a lazo cerrado

Hay otra clasificación en el control de los motores de pasos, basado en la retroalimentación y son a lazo cerrado o a lazo abierto.

Control a lazo abierto

El sistema de control de lazo abierto es él más común dado que es ideal para sistemas que operan a bajas aceleraciones y cargas estáticas. En un sistema de control de lazo abierto, no existe la retroalimentación y la posición del rotor es desconocida, sin embargo, el modelo de simulación puede predecirla, basándose en su posición relativa y de velocidad como se muestra en la figura 2.9.

El sistema de control explicado proporciona la información necesaria en pulsos y control de pasos dados por una fuente externa y se espera que el motor de pasos sea capaz de seguir cada pulso.

El lazo abierto es usado en aplicaciones de control de velocidad y posición. Sin embargo, el rendimiento de un motor de pasos es limitado mediante el control de lazo abierto, cuando se desea trabajar a velocidades muy altas.

Aunque el lazo abierto es un método de control económicamente ventajoso, no está libre de limitantes, una de estas desventajas es que

un motor de pasos en modo de lazo abierto puede fallar para seguir un envío de información de tren de pulsos cuando la frecuencia es alta.

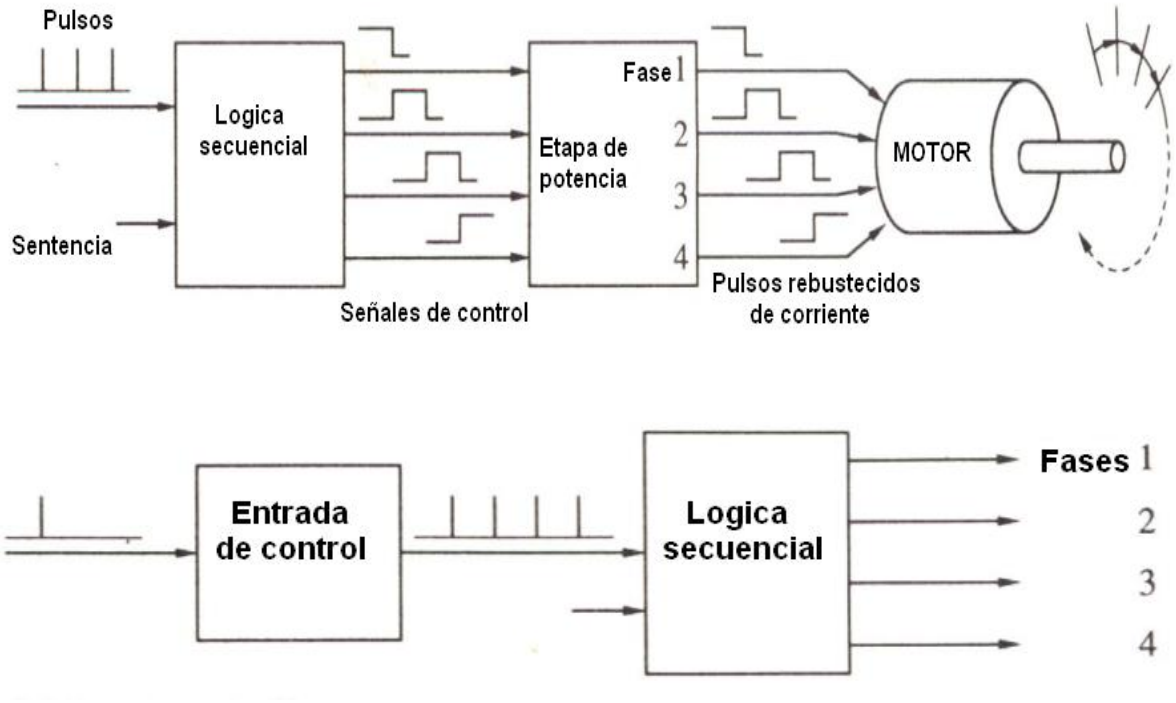


Figura 2.9 Diagrama de bloques del control a lazo abierto

Control a lazo cerrado

El control a lazo cerrado es un método muy efectivo, libre de inestabilidades y capaz de acelerar rápidamente. El sistema de lazo cerrado es esencial para altas velocidades y cargas variables. En lazo cerrado existe un sensor que monitorea directamente el rotor del motor, detectando su posición y la envía al sistema de control.

Cabe recalcar que la mayoría de aplicaciones de los motores de pasos en sistemas de control en lazo cerrado se basan en este modelo. Normalmente se emplea un microprocesador de alguna computadora cuando hay una retroalimentación, dado que es más sencillo usar las entradas y salidas de dicho microprocesador.

De tal forma la operación de los motores de pasos, se puede mejorar empleando retroalimentación de su posición o de su velocidad y así determinar la fase apropiada, para ser activado el campo magnético en el embobinado. En un sistema más avanzado, en lugar

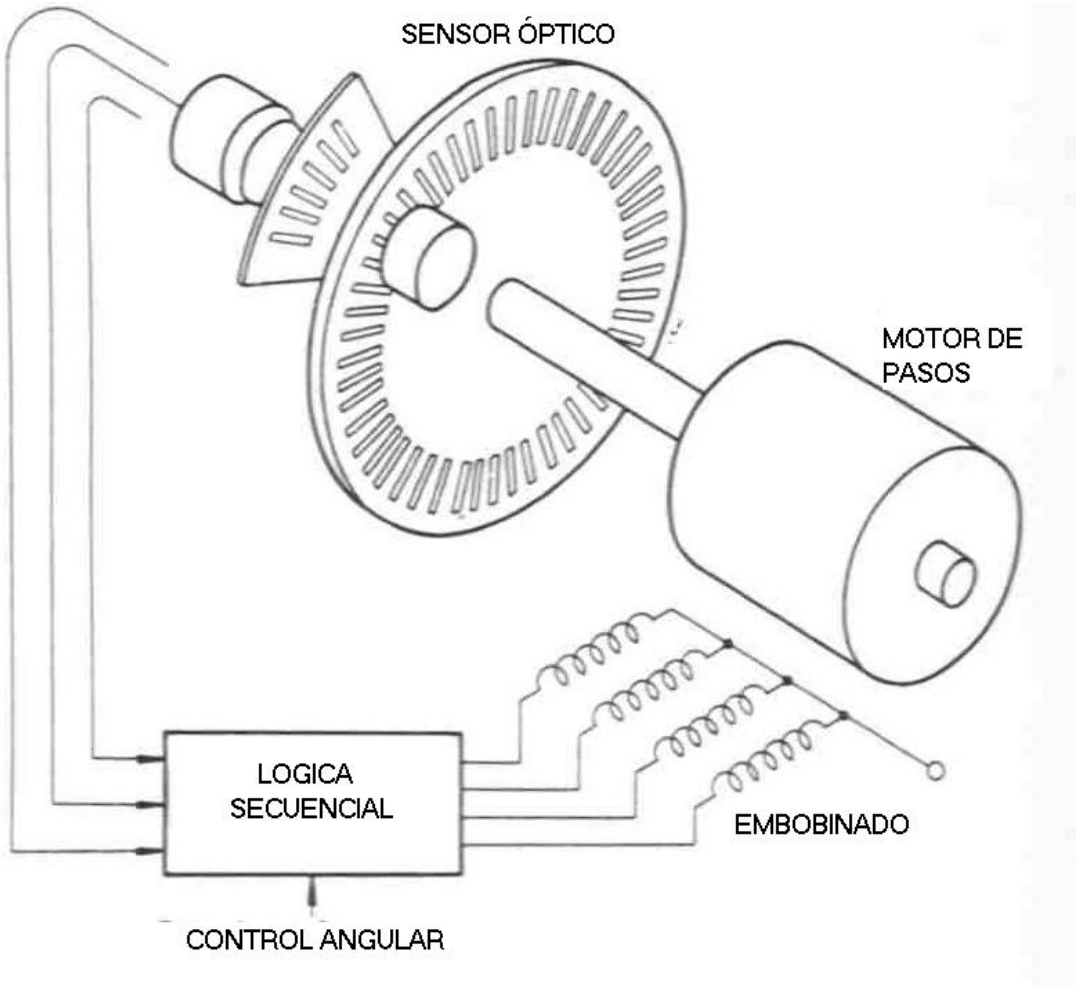


Figura 2.10 Diagrama de bloques del control a lazo cerrado de un motor a pasos

de un sensor mecánico adicional, la posición del rotor es sensada por dispositivos ópticos que pueda determinar la posición del rotor, como se muestra en la figura 2.10.

A diferencia del control de lazo abierto el control a lazo cerrado es más útil, para velocidades mas grandes, dado que la pérdida de pasos nunca ocurre, debido aun doble sensor óptico en el rotor.

En cuanto a los pasos que realiza el motor son debidos al dispositivo óptico acoplado al rotor que detecta la posición del mismo,

emitiendo su información al sistema de control lógico, donde el controlador de pulsos determina las fases apropiadas a ser excitadas.

Normalmente un motor a pasos que se controle a lazo cerrado, su construcción se basa en una fuente de luz y un sensor además de un encoder originando un bloque periódico de luz en la cual se genera una información digital en el sensor de luz, como se muestra en la figura 2.11.

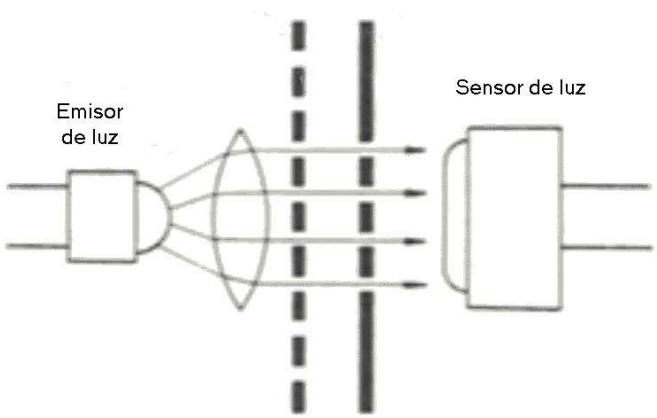


Figura 2.11 Principio óptico del encoder

Al conocer la última posición del motor, es posible determinar la diferencia entre la posición actual y la posición final o deseada, enviando esta información a un microprocesador. De esta forma, la posición final es calculada por el microprocesador dando el pulso exacto para que el rotor se ubique en el lugar deseado.

Estos microprocesadores aparte de ser económicos, su empleo ayuda a que la señal sea retroalimentada al hardware, que monitorea todo el sistema, dado que el software debe ser programado de manera que el microprocesador determine el mejor desempeño para defasar las señales que controlará al motor, basándose en las secuencias anteriores.

Hoy en día existen algunos circuitos integrados que permiten la operación de un motor en lazo cerrado, los cuales reducen mucho el espacio y costo del sistema de control, aunque son muy limitados para el cambio de instrucciones dado que estos ya vienen programados de fábrica.

Una forma de controlar un motor de pasos es mediante el uso de un controlador lógico, que es un circuito que controla la excitación de los embobinados respondiendo al comando de pasos por los pulsos. Está usualmente compuesto de registros “shift” y compuertas lógicas. Existen circuitos lógicos construidos para este propósito, la ventaja de

ARREGLO DE COMPUERTAS PROGRAMABLES DE CAMPO

Capítulo 3

Este capítulo introduce el concepto de FPGA, desde su evolución hasta las ventajas de su lógica configurable, así como también las características principales del FPGA de la compañía Altera, incluyendo su software de desarrollo MAX+PLUS II.

¿QUE ES UN FPGA?

Un FPGA (Field Programmable Gate Array) es un dispositivo que combina las ventajas de la tecnología de los Gate Array's con la flexibilidad y facilidad de diseño de la lógica programable.

Al igual que un Gate Array, un FPGA consta de una base de células independientes programables, que se pueden interconectar a través de canales también programables, originando funciones de entradas y salidas.

EVOLUCIÓN DE DISPOSITIVOS DE LÓGICA PROGRAMABLE

El primer tipo de circuito integrado programable por el usuario que podía llevar a cabo operaciones lógicas, era la memoria programable de solo lectura, llamada *PROM*; en la que pueden usarse líneas de dirección como entradas del circuito lógico y líneas de datos como salidas. Una PROM contiene un decodificador ocupado en su totalidad para sus entradas de dirección; teniendo así las PROM's una arquitectura ineficaz para implantar circuitos lógicos.

El próximo dispositivo desarrollado conocido como PLA tiene una estructura basada en compuertas lógicas AND y OR. Cada salida del nivel OR puede configurarse para producir la suma lógica de cualquiera de las salidas de un nivel AND. Con esta estructura, los PLA's están preparados para llevar a cabo funciones lógicas en forma

de suma de productos. Ellos también son bastante versátiles, dado que ambos, el término AND y término OR pueden tener muchas entradas (esta característica es a menudo llamado ancho de compuertas AND y OR).

Cuando los PLA's se introdujeron a principios de los años setenta, por la compañía Phillips, sus inconvenientes principales eran los altos costos de fabricación y ofrecían una velocidad de acción bastante lenta.

Se desarrollaron los PAL's. Las PAL's ofreciendo sólo un nivel de programación, que consiste en programar AND's que alimentan a compuertas OR fijas. Para compensar la falta de generalidad debida al nivel OR fijo, se produjeron algunas variantes de PAL's, con números diferentes de entradas y salidas, y varios tamaños de compuertas OR.

Las últimas versiones de PAL's contienen flip-flop's conectados a las salidas de las compuertas OR para poder realizar circuitos secuenciales. Las PAL's son importantes porque cuando se introdujeron tuvieron un efecto profundo en el diseño del hardware digital, y también son la base para algunas de las más nuevas y sofisticadas arquitecturas. Todos los PLD's pequeños, PLA's, PAL's, y los dispositivos tipo PAL se agrupan en una sola categoría llamada PLD's simple, es decir SPLD's, cuyas características más importantes son bajo costo y muy alta velocidad de actuación entre pines, así mismo se puede observar en la figura 3.1.

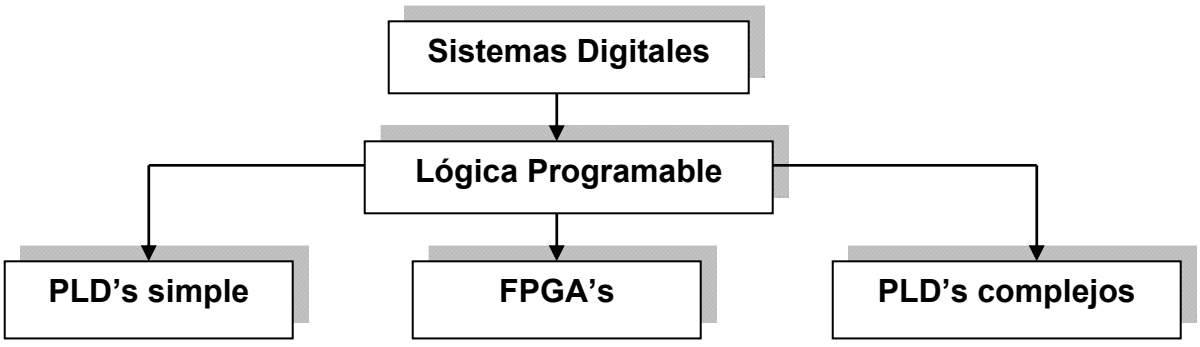


Figura 3.1 Organigrama de la lógica configurable programable

Con el avance de la tecnología, se han producido los SPLD's, de mayor capacidad. La dificultad con el aumento de capacidad de una arquitectura de SPLD, es que la estructura de los niveles lógicos programables crecen demasiado en tamaño a medida que el número de entradas aumenta. La única manera factible de proporcionar dispositivos de capacidades grandes, basadas en arquitecturas de SPLD's es el integrarlas en un solo chip y proporcionar interconexiones para conectar los bloques SPLD en forma programable. Muchos productos de dispositivos de cambios programables (FPD) comerciales existen en el mercado hoy con esta estructura básica, y son llamados PLD's Complejo (CPLD's).

Pionera en los CPLD's, Altera fue la primera que lanzó al mercado una familia de chips llamada EPLD's Clásico, y tres series adicionales llamadas, MAX 5000, MAX 7000 y MAX 9000. Debido a un crecimiento rápido del mercado de los FPD's, otros fabricantes desarrollaron dispositivos en la categoría de CPLD y ahora hay muchas opciones disponibles. Los CPLD's proporcionan una capacidad lógica equivalente a aproximadamente 50 dispositivos de SPLD's típicos, pero es algo difícil de extender estas arquitecturas a densidades más altas.

La tecnología con mayor capacidad de lógica disponible hoy son los chips de propósito general, tradicionalmente llamados arreglos de matrices de compuertas programables (MPGA's). Los MPGA's consisten en una serie de transistores que pueden personalizarse en el circuito lógico del usuario conectando los transistores entre sí de acuerdo a la necesidad. La personalización se realiza durante la fabricación del chip especificando las interconexiones de metal, y esto significa que para que un usuario emplee un MPGA existe un costo industrial grande y el tiempo de manufactura es largo.

Aunque los MPGA's no son claramente FPD's, tiene su importancia porque motivaron un diseño programable por el usuario, llamados FPGA's (Arreglos de compuertas programables de campo).

Así como los MPGA’s, los FPGA’s comprenden una serie de elementos de circuito no especificados, llamados bloques lógicos, e interconexión de recursos, pero la configuración del FPGA se realiza a través de una terminal de programación del usuario. Este ha sido el único tipo de FPD que soporta una capacidad alrededor de dos mil compuertas lógicas, los FPGA’s han sido los responsables del cambio en la manera que se diseñan los circuitos digitales.

Existen otro tipo de FPGA’s para aplicaciones específicas según su capacidad para almacenar compuertas lógicas (máquinas de estado, arreglo de compuertas analógicas y grandes interconexiones). Aunque el uso de estos dispositivos es muy limitado.

LÓGICA CONFIGURABLE Y PROGRAMABLE

La lógica programable se define como un mecanismo de lógica configurable, acompañados de flip-flop's unidos por medio de interconexiones programables. Las células de memoria controlan y definen la función lógica a realizar y cómo estas se interconectan. Aunque los dispositivos usan arquitecturas diferentes, todos están basados en este principio.

Ventajas de la lógica configurable programable

Es indispensable el conocimiento de los FPGA’s dado que estos dispositivos y su aplicación son excelentes a nivel estudiantil, pudiendo crear un diseño electrónicos y ser programados en estos dispositivos sin los problemas de costos.

No hay costos de ingeniería recurrente, tampoco tardanza mientras se esperan los prototipos para ser fabricados.

Dado que los dispositivos son software configurado y usuario - programador, las modificaciones son de menor complejidad y pueden ser hechas a cualquier hora, lo cual ahorra costo en el diseño
Utiliza una simplicidad extrema, únicamente tiene el uso de lógica programable discreta.

Es la herramienta mas rápida de la lógica configurable programable, de alta confiabilidad y de bajo costo, tomando en cuenta la cantidad de compuertas que se utilizan dentro del diseño.

Es fácil la modificación en los diseños, reutilizando circuitos, lo que permite asegurar el mantenimiento del producto, aumentando su vida comercial en la industria. Mejora el aprovechamiento de los recursos de ingeniería, teniendo como resultado menores costos de desarrollo.

Existen dispositivos de lógica programable, clasificándolos como a continuación se muestra según su arquitectura:

- Dispositivos de Lógica Programable (PLD's)
- Dispositivos de Lógica Programable simples (SPLD's),
- Dispositivos de Lógica Programables complejos (CPLD's)
- Arreglos de Matrices de Compuestas Programables (MPGA's)
- Arreglos de Compuertas Programables de Campo (FPGA's)

Dispositivos Lógicos Programables (PLD's)

Los PLD's, entre los que se destacan las PAL por su popularidad y comercialización, son una buena solución para densidades de integración de unos pocos cientos de puertas. Una de las principales limitaciones de estos circuitos es el número de flip-flop's, el numero de entradas y salidas así como la lógica de las compuertas AND y OR, además de sus interconexiones como se muestra en al figura 3.2.

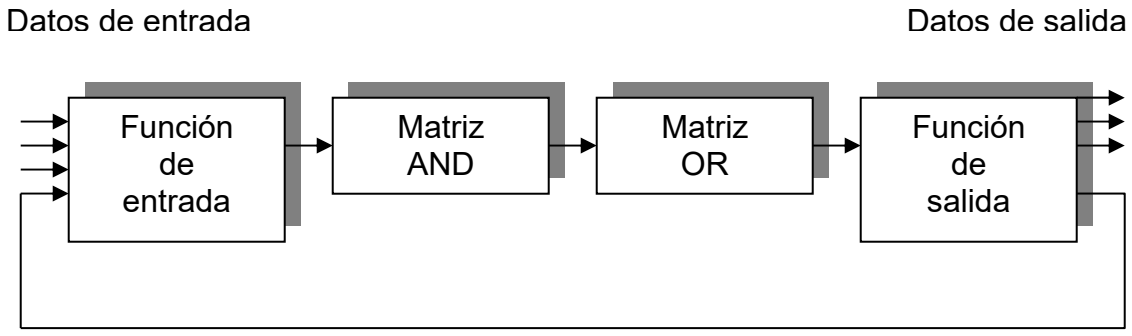


Fig. 3.2 Diagrama de bloques de un PLD simple

Dispositivos de Lógica Programable simples (SPLD's)

Los SPLD's constan de una mayor capacidad en comparación con los PLD's, su estructura de los niveles lógicos programables van aumentando según el número de entradas; todas ellas contenidas en un solo circuito integrado proporcionando interconexiones programables.

Fig. 3.3 SPLD de Lattice

Dispositivos de Lógica Programable Complejos (CPLD's)

Los CPLD's son similares a SPLD's sólo que ellos son de capacidad significativamente más alta. Un CPLD típico es el equivalente a un dispositivo de lógica programable, que va desde el doble hasta sesenta y cuatro veces un SPLD. Un CPLD contiene de decenas a centenas de macroceldas.

Generalmente los CPLD's son de tecnología CMOS y usan celdas de memoria no-volátiles como EPROM, EEPROM, o FLASH EPROM. Muchas de las familias de CPLD introducidas recientemente usan un EEPROM o FLASH.

Arreglos de Matrices de Compuertas Programables (MPGA's)

Son circuitos integrados de propósito general, y formados por transistores en un circuito lógico; existiendo conexiones entre si., dando como resultado que al usuario le sea difícil absorber el costo industrial que produce y el tiempo de manufacturado es prolongado.

Aun cuando no cuentan con la clasificación de FPD's, los MPOGA's originaron diseños programados directamente por el usuario para mas tarde evolucionar a FPGA's.

Arreglos de Compuertas Programables de Campo (FPGA's)

Una FPGA es un circuito integrado el cual se puede programar las veces que se desee. Se puede programar para que se comporte como un simple inversor o como algo tan complejo como un microprocesador. Utilizando una herramienta de diseño asistido por computadora realiza un diagrama esquemático o con lenguaje de programación, después un software genera automáticamente un archivo de configuración para que posteriormente se descargue al chip.

Hay distintas metodologías para la introducción del diseño de circuitos como la captura de diagramas esquemáticos, los lenguajes de descripción hardware, y la descripción de máquinas de estados finitos.

Los FPGA's han llegado a ser populares por los prototipos y diseños de complejos sistemas de hardware. La estructura de un FPGA (figura 3.5) puede ser descrita como un "arreglo de bloques" conectados unos con otros, por medio de interconexiones programables. Un ingeniero puede refinar el diseño de un dispositivo explotando la reprogramación de este. El fabricante Xilinx introdujo por vez primera la palabra FPGA, en su chip XC2064, en 1985. Este dispositivo contenía aproximadamente 1000 compuertas lógicas. Desde entonces, la densidad de compuertas de los FPGA's se han incrementado 25 veces.

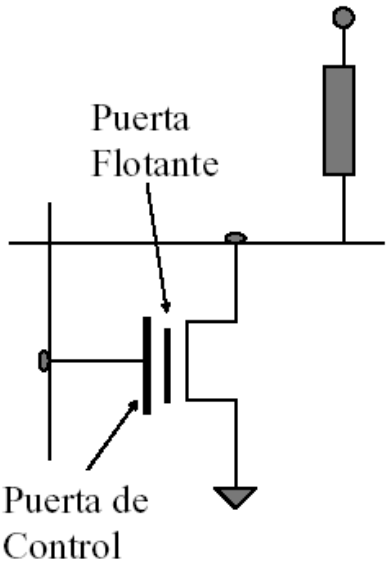


Fig. 3.4 Principio de programación del FPGA

Los primeros FPGA aparecen en 1985 producidos por la compañía norteamericana Xilinx, fundada un año antes, basándose en una idea revolucionaria, que consiste en combinar la alta densidad y versatilidad de las matrices de compuertas con la popularidad y ventajas de los circuitos programables por el usuario (PLD's), como se muestra en la figura 3.4.

FPGA

- **Ventajas**
  - -Diseños reutilizables
  - -Cambios en especificiones
  - -Capacidad de almacenamiento
- **Aplicaciones**
  - -Emulación de sistemas de hardware enteros
  - -Lógica de azar
  - -Integración de múltiples SPLD's
  - -Controladores de dispositivos
  - -Filtros
  - -Pequeños y medianos sistemas de SRAM
  - -Prototipos de diseños
  - -Entre otras
- **Arquitectura**
  - -*Bloques de lógica configurable* (LCA): cada uno de ellos utilizan lógica combinacional programable y registros de almacenamiento
  - -Bloques de entrada y/o salida: cada bloque puede ser configurados por el usuario como pin de entrada y/o salida, con control de triestado o bidireccional
  - -Bloque de interconexión: son los recursos programables que controlan la interconexión entre dos puntos cualesquiera en el chip
- **Tarjetas de desarrollo**
  - -Es posible trabajar físicamente lo que se ha programado en VHDK o n un diagrama esquemático
  - -Cuenta con un puerto de datos para la PC (LPT), conectores para los acceso para los puertos, socket para alojamiento de CPLD's buffer de tres estados, regulador de voltaje de 5volt y alimentación de 6 a 27 volts
- **Software de programación**
  - **Gráfico**
    - -Permite simulación
    - -Permite verificar el diseño real de donde va a ser utilizado
    - -Fase de comprobación
    - -Compilación de circuito utilizando herramientas CAD
  - **Estructurado**
    - -Crea y edita diseños de texto, basados en los lenguajes AHDL, VHDL y Verilog HDL
    - -Crea, visualiza o edita archivos ASCII

Tabla 3.1. Cuadro sinóptico de las características del FPGA

La aplicación de estos dispositivos ha sido abundante en muchos campos de la ciencia. Algunas de las aplicaciones de estos dispositivos son por ejemplo en la implantación de redes neuronales y sistemas complejos de control.

Un aspecto importante de los FPGA's es que están involucrados en tecnología evolutiva, dado que estos dispositivos se pueden reconfigurar de forma dinámica.

Desde entonces, los FPGA's han experimentado una importante evolución, tanto en ventas, como en tecnología dado que en menos de tres años se aumentó la densidad en 7 veces, la velocidad se multiplicó por 5 y el costo se redujo en un 25% del inicial.

DISEÑO DE LA LÓGICA PROGRAMABLE

En el diseño de un arreglo lógico programable se usa una metodología típica que involucra los siguientes casos:

Se debe *diseñar una entrada*, con el gran número de herramientas disponibles, existen varios tipos como el paquete esquemático, aunque también se usa un idioma de descripción de hardware como Verilog, VHDL, o ABEL.

Se diseña la aplicación, dentro de un esquemático, según el diseño que se eligió, involucrando convertir el plan en el formato apoyado internamente por las herramientas.

La comprobación o *simulación* del programa, ocurre a varios niveles y pasos a lo largo del planeamiento de programación. Hay algunos tipos de comprobación, dependiendo de la lógica programable aplicada. Esta simulación funcional se realiza junto con la compilación del programa.

Esquemáticamente una forma de representar como se graba un programa en un FPGA es el de la figura 3.5.

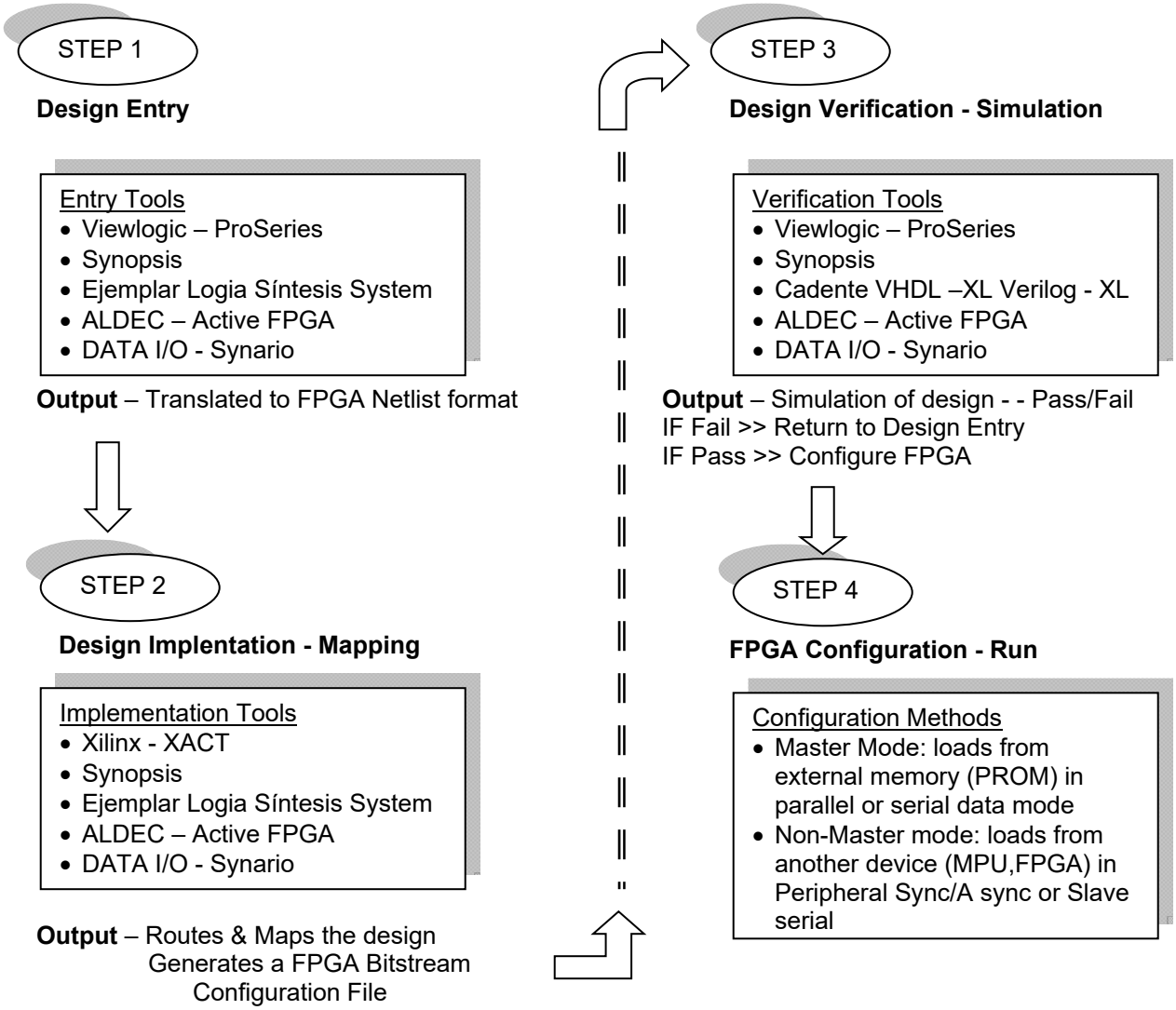


Fig. 3.5 Pasos para la programación de un FPGA

Y finalmente la *programación* del dispositivo, dado que después de crear un archivo de la programación, el dispositivo programable se configura y se prepara para dicha acción. Las tecnologías de lógica programables, incluso PROM's para FPGA's o SRAM, requieren alguna clase de programador del dispositivo.

FPGA's DISPONIBLES COMERCIALMENTE

Como uno de los segmentos más crecientes de la industria del semiconductor, ha surgido el FPGA. Como tal, la serie de compañías involucradas cambia rápidamente y es algo difícil decir qué productos

COMPAÑIA REQUERIMIENTOS	ALTERA	XILINX	AT&T Y ORCA	ACTEL	QUICKLOGIC
ESTRUCTURA	EPROM y EEPROM	SRAM	SRAM	Anti-Fuse	Anti-Fuse
CAPACIDAD (Compuertas lógicas)	30,000	15,000	40,000	-	-
FACILIDAD DE USO	Sencillo	Aceptable	Complejo	Complejo	Complejo
INTERCONEXIONES	Aplicable	Aplicable	No aplica	No aplica	No aplica
NIVEL DELTIEMPO DE DESARROLLO	Alto	Alto	-	-	Medio
SOPORTE TÉCNICO	Aplicable	Aplicable	Aplicable	Aplicable	Aplicable
TIPOS DE PROGRAMACIÓN VHDL	Gráfico (Max+Plus II) y Estructurado (VHDL)	VHDL y Verilog	Estructurado	-	Estructurado
SOFTWARE	Librerías, editores esquemáticos y simulación	Librerías y simulación	-	-	-

Tabla 3.2 Tabla comparativa de FPGA´s comerciales

serán los más importantes y significativos cuando la industria alcance un estado estable. Por esta razón no se mencionan a los fabricantes mas representativos de FPGA's que actualmente existen como se muestra en la tabla 3.2.

Xilinx y Altera son los fabricantes principales en términos de número de usuarios, seguidos por AT&T. Para los productos basados en antifuse, están Actel y Quicklogic . Algunas otras de las compañías que desarrollan esta tecnología son: Lucent, Motorola, Phillips, TI (Texas Instruments), Vantis y XESS Corporation.

FPGA DE ALTERA

INFRAESTRUCTURA, ESPECIFICACIONES Y APLICACIONES

Los entornos EDA (Electronic Design Automation) son herramientas de trabajo comúnmente empleadas en el diseño e implantación de circuitos lógicos, permiten una gran versatilidad al momento de simular el funcionamiento de los diseños.

El entorno EDA es sencillo de manejar debido a las interfases gráficas que proporcionan las plataformas de diseño como Altera, OrCAD y Xilinx, permitiendo su instrumentación y su implantación en los CPLD's y FPGA's.

Para la implantación de los diseños se cuenta con una tarjeta de desarrollo llamada UP1, que contiene dos CPLD's. Esta tarjeta va a permitir una fácil implantación y verificación de diversos diseños digitales, mediante una serie de visualizadores, pulsadores, microinterruptores y otros elementos útiles para realizar simulaciones físicas en la tarjeta de desarrollo.

En el siguiente diagrama (Tabla 3.3) se contemplan las características mas sobresalientes del hardware y software del FPGA de la compañía Altera.

SOFTWARE Max+Pus II

Figura 3.7

APLICACIONES

- Hierarchy Display

Muestra la distribución actual de la jerarquía del proyecto, representando diseños

- Graphic Editor

Crea un diseño visual en un ambiente (WYSIWYG) "Lo que ves es lo que tienes"

- Symbol Editor

Convierte diseños previos en símbolos nuevos para ser utilizados en la jerarquía

- Text Editor

Crea y edita diseños de texto basados en los lenguajes AHDL, HDL y Verilog HDL. Permite crear, ver o editar archivos ASCII

- Waveform Editor

Herramienta de diseño. Herramienta para crear vectores de prueba y observar los vectores resultantes de la simulación

- Floorplan Editor

Realiza asignación manual de pines en el dispositivo en relación al diseño lógico

- Compiler

Procesa los proyectos creados, encontrando posibles errores y generando los archivos de programación necesarios

- Simulator

Permite revisar la operación lógica con diagramas de tiempo analizando las entradas y encontrando las salidas

- Timing Analayzer

Analiza el rendimiento del circuito después de ser compilado

- Programmer

Verifica, examina, prueba y programa los dispositivos

- Message Processor

Muestra mensajes de error, advertencias y el estado actual del proyecto

FAMILIAS

SUMA DE PRODUCTOS
- Max 5000
- Max 7000
- Max7000S
- Max 9000

LOOK UP TABLE
- Flash Logic
- Flex 8000
- Flex 10k

ARQUITECTURA

Contiene:

- Elementos lógicos bloques básicos basados en LUT
- Celdas lógicas Bloques básicos de AND y OR
- Macrodeldas Boque básico de suma de productos lógicos

TARJETA DE DESARROLLO

Figura 3.6

UP1

1. DC_IN Regulador de voltaje 1A
2. Jumpers de configuración
3. JTAG_IN Header
4. Generador de reloj a 8MHz
5. Display's de 8 segmentos
6. Conectores de expansión
7. Dispositivo EMP7128SLC
8. Dip Switch
9. Push Buttons
10. LED's
11. Dispositivo EPF10k20

Tabla 3.3 FPGA de la compañía ALTERA

38

38

LENGUAJE DE ALTO NIVEL VHDL

El VHDL, contiene siglas las cuáles podemos dividir esta palabra en dos: VHDL (Very high speed integrated circuit) y HDL (Hardware Description Language).

El VHDL ofrece tres características, entre las que intervienen el modelado, donde es la descripción abstracta de circuitos integrados, la simulación en el cuál se realizan los vectores de prueba y la captura de resultados y por último la síntesis, en la que se da la información de instrucciones VHDL a circuitos digitales físicos.

Este tipo de creación de proyectos usando VHDL es tipo Top-Down, que es, el proceso de capturar una idea en un alto nivel de abstracción, después implementar esa idea e ir hacia abajo incrementando el nivel de detalle, según sea necesario. Así, mediante el software disponible por la computadora, se pueden diseñar circuitos más complejos en periodos cortos de tiempo. Las ventajas del diseño Top-Down son que incrementa la productividad del diseño, además de la reutilización del diseño y la detección de errores.

Así mismo el VHDL tiene varias plataformas, donde las más conocidas son, Active- HDL, Foundation, Model-sim, Leonardo Spectrum, Warp y Renoir.

MAX+PLUS II

El software Max+Plus II (Múltiple Array Matrix Programmable Logic User System II) suministra una plataforma múltiple e independiente de una arquitectura en particular, que se adapta fácilmente a las necesidades de cualquier diseño. Y por ello ofrece una forma fácil de diseñar proyectos y hacer su programación directa en los dispositivos.

En esta tesis se toma como referencia el uso de la versión 10.1 de libre distribución para estudiantes. Una vez instalado el software resulta necesario solicitar una licencia vía correo electrónico y conocer

el número de serie del disco duro de la PC en que se ha instalado. El software utilizado no es exigente respecto a los requisitos hardware en que se instale, de modo que tendrá una ejecución muy satisfactoria si se cuenta con un procesador Pentium y al menos 64 Mb de memoria. Como último paso, se carga la nueva librería (licencia) y se puede empezar a utilizar el software. Figura 3.7

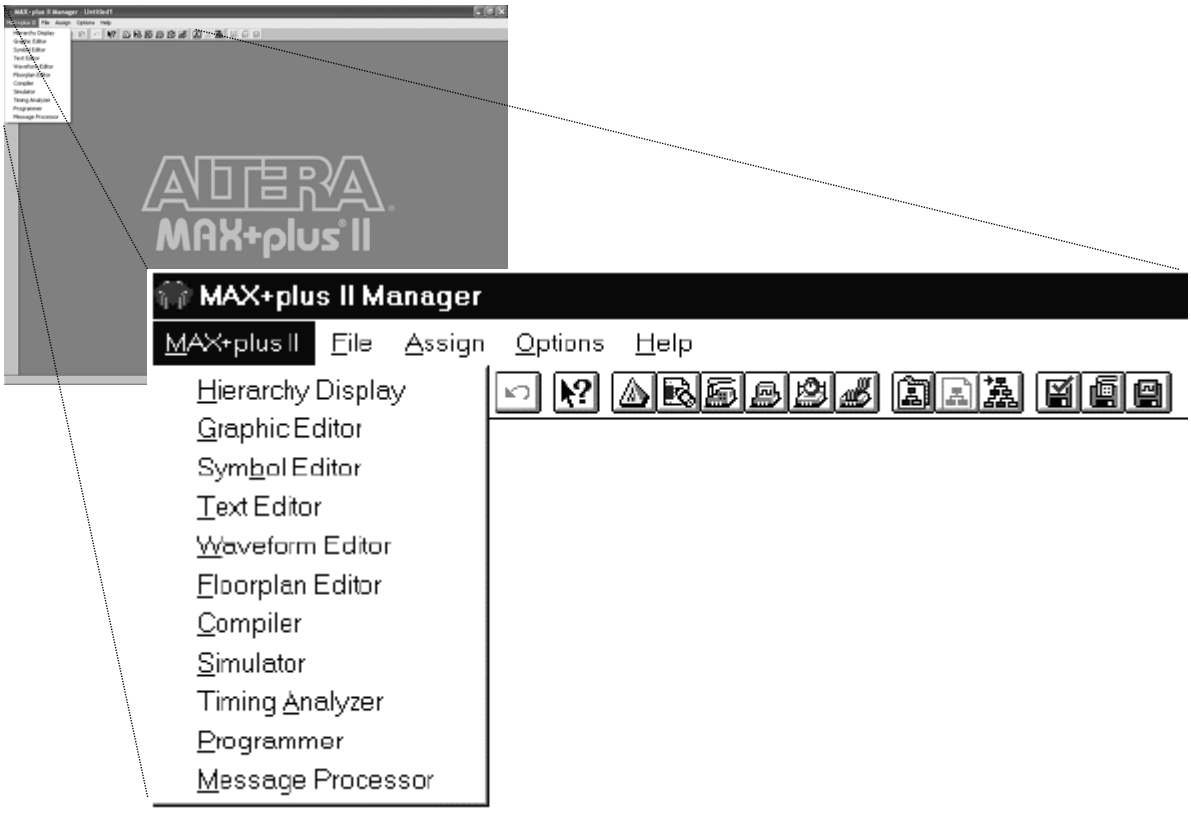


Figura 3.7 Desplegado de las once aplicaciones de Max+Plus II

IMPLEMENTACIÓN DEL FPGA CON LOS MOTORES BIMODALES

Capítulo 4

En este capítulo se explica como se conjunta la tarjeta de desarrollo (FPGA) con el motor bimodal.
.

ASPECTOS GENERALES

Para la integración o incorporación de las interfaces fue necesario verificar de manera independiente cada una de las partes que componen la totalidad del proyecto, entre las cuales destacan las enunciadas a continuación:

- FPGA (Hardware)
- MAX+PLUS (Software de Altera)
- Motor bimodal (a pasos – corriente directa)
- Etapa de potencia (de cada motor)
- Optoelectrónica (Sensado)
- Tarjeta de protección para LPT (Mother Board)
- Programa de control (Borland C++ Builder)
- Circuitos Impresos (P-CAD)
- Simulaciones (PSpice, Electrónica Workbench y Protel)

RECURSOS

Este proyecto fue desarrollado en el Centro de Ciencias Aplicadas y Desarrollo Tecnológico UNAM, a través de su laboratorio de Micromecánica y Mecatrónica, en donde se contó con la siguiente infraestructura y recursos:

- Un taller mecánico (Fresa, Torno, Esmeril y herramientas convencionales) Figura 4.1.
- Una sección de electrónica (con Osciloscopio, Fuentes de voltaje y corriente, etc.).
- PC's de escritorio con procesador Pentium II.

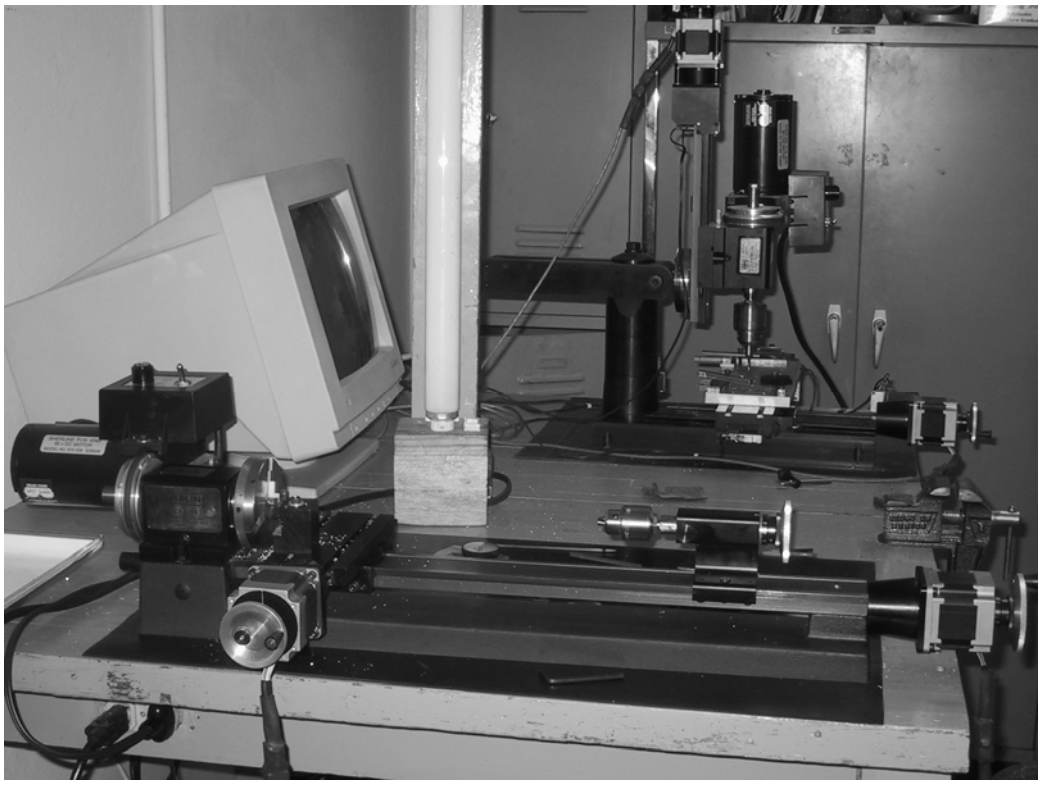


Figura 4.1 Máquinas herramientas del LMM torno (izquierda) y fresa (derecha)

Y por otra parte se contó con recursos humanos, y entre ellos destacan los siguientes:

- Dirección y supervisión del Dr. Ernst Kussul, Coordinador del Laboratorio de Micromecánica y Mecatrónica de CCADET.

- Codirección por parte del M.I. Leopoldo Ruiz y del M.I. Alberto Caballero.
- Información mecánica por parte del técnico Mario Rodríguez.
- De manera adicional, para el diseño y construcción de dicho proyecto fueron proporcionados recursos financieros por parte del Consejo Nacional de Ciencia y Tecnología (CONACyT) a través de su proyecto 33944-U.

PROCEDIMIENTO

Para desarrollar satisfactoriamente este tema de tesis, fue necesario establecer una secuencia de trabajo la cual se enuncia a continuación:

El inicio del proyecto se relaciona a las micromáquinas en México y en el Mundo, con esto como marco conceptual, con la finalidad de mejorar del microequipo existente y también los proyectos en desarrollo, enfocándonos en sus motores, que deban ser controlados a muy altas o bajas revoluciones, según sus requerimientos de posicionamiento.

DESARROLLO DEL PROYECTO

En el LMM se trabaja con microequipo, en la sección de micromanipulación y de microfabricación, forzosamente se requiere el uso de motores que proporcionen el movimiento deseado de piezas, así mismo se llega a la necesidad de trabajar con motores de pasos, que por sus características otorgan el par mecánico deseado.

Con el desarrollo de prototipos de microequipo, se tenia la necesidad de diseñar otros motores de pasos, con mayor rendimiento en espacio, además de una nueva forma de controlarlos, por la razón de que el tiempo de producción de los prototipos para generar micropiezas era demasiado.

El proyecto de la tesis entonces se enfocó al desarrollo de los motores, que se venían elaborando en días anteriores, se tenia que trabajar en el proyecto de control de microequipo, mediante un proceso que en un futuro contara con mayor control sobre la posición

de sus movimientos, además de ser cada vez mas independiente de un usuario y hasta de la misma computadora.

Como no se contaba con motores que realizaran movimientos rápidos, especialmente cuando se requería desplazar a un posición inicial, se planteó el diseño de un control en el cuál, uno o varios motores de pasos trabajaran por arriba de las 10,000 r.p.m. de forma independiente entre ellos, además tuvieran aún la oportunidad de trabajar a pasos, aprovechando su precisión que los caracteriza, y así obtener un *motor bimodal*.

Para satisfacer estas necesidades se plantea el uso de la lógica programable, donde un FPGA, reúne las características para llevar a cabo este primer paso, que nos deslinde de la computadora.

Como la última parte del proyecto, era tener un control total de los pulsos generados, cuantificarlos y obtenerlos de una forma muy rápida para hacer mediciones de posiciones en el microequipo, para ello se plantea una solución con el monitoreo de los pulso de los motores de pasos con un programa en la computadora, en donde se sugirió en uso de C++, por su versatilidad de programación por ser orientado a objetos.

A continuación se plantea minuciosamente el desarrollo de las etapas de cómo se trabajo y la solución a estos problemas.

PLANEACIÓN DEL CONTROL DEL MOTOR

Para la solución de movimientos del microcentro de maquinado, se consideraron varias alternativas de actuadores, en la cual se optó por los diseños que presentan sencillez de manufactura, ensamble y bajo costo.

Los motores que se han desarrollado en el LMM de tipo bipolar, se componen de 4 bobinas soportadas por una placa de latón y otra de acero que soportan al eje del rotor, el cual esta fabricado en material ferromagnético y es magnetizado de forma radial. Las bobinas, con una impedancia de 8 ohms aproximadamente se conectan de forma cruzada para obtener cuatro puntos magnéticos de contacto, como se muestra en la figura 4.4.

HACIA UN NUEVO CONTROL DEL MOTOR

El funcionamiento común de un motor a pasos está determinado por la forma de alimentación del embobinado, el cual es generada a través de pulsos en cada una de sus fases para cada bobina (dependiendo del diseño del motor); antecedidos por una etapa de potencia de la señal como se muestra en la figura 4.2, se puede

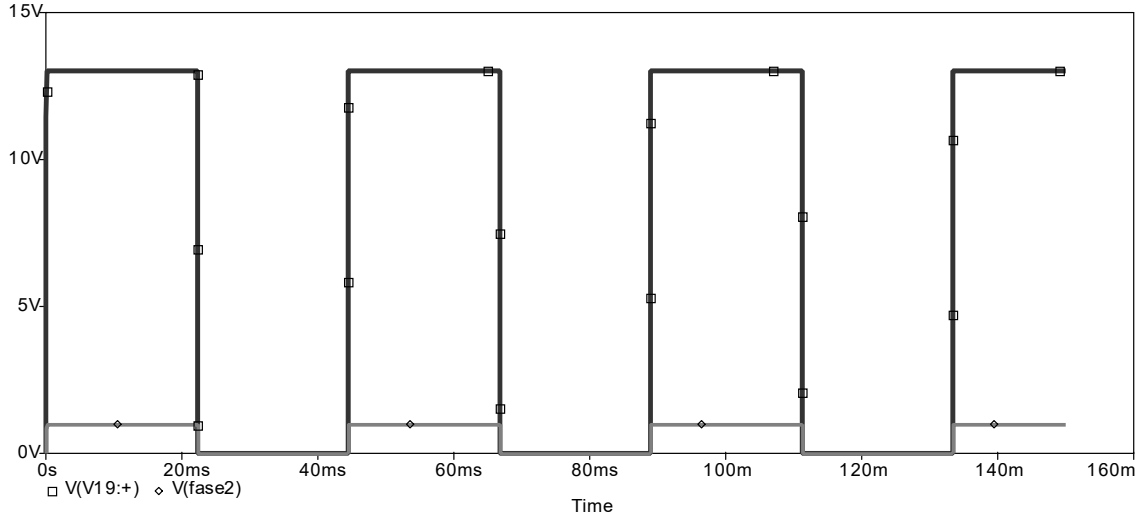


Figura 4.2 Robustecimiento de la señal para la alimentación del embobinado

alimentar al sistema de bobinas del motor de forma que un defasamiento proporcione la manipulación magnética del rotor donde el tren de pulsos enviados son generados por un sistema externo, como lógica programable o alguna computadora.

Una de las restricciones que tiene la ultima versión de los motores ocupados en el microequipo desarrollado en el LMM, son las frecuencias que utilizan los motores, donde al aumentar sus velocidades los pulsos se traslapan, llegando el momento en que el embobinado no puede determinar si la señal enviada es de un “1” lógico o bien “0” lógico, ocasionando pérdidas de información e interrumpiendo las revoluciones del motor.

Dicho problema, se decidió atacar haciendo el diseño de un prototipo del control del motor, en el cual pueda aumentarse el número de los pulsos que controlan los embobinados de los motores, obteniendo velocidades superiores en un 100% con respecto a del control anterior. Cabe destacar que su alimentación no seria la

convencional a pulsos, porque debe de llevar una etapa alimentada por corriente directa se pueda controlar la frecuencia que determinará la velocidad del motor.

Desarrollo del motor controlado a lazo abierto con CC. Se desarrollo un motor alimentado por corriente directa, este motor de pasos utilizado esta conformado mediante técnicas convencionales de la mecánica, fabricado en el LMM donde sus resultados obtenidos en las pruebas se llegó a un mejor tamaño y mayor eficiencia en su funcionamiento, como el mostrado en la figura 4.3.

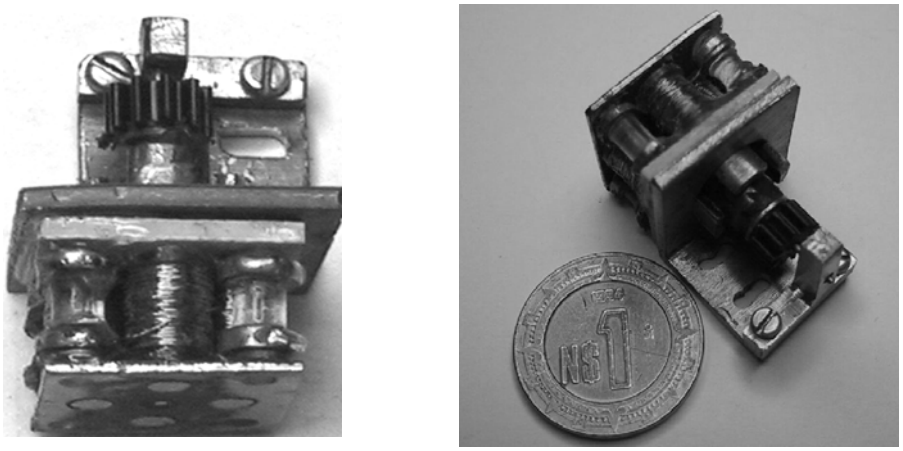


Figura 4.3 Motores de pasos diseñados en el LMM

En la investigación de la transformación de motores de pasos a motores controlados con corriente continua (CC) con dos pares de bobinas, se trabajó con un circuito que defasa 90° una señal generada con voltaje de una de las bobinas, al proporcionar un par mecánico, formando un pulso el cual sirve de alimentación y con una etapa de potencia, la otra bobina puede ser alimentada y así obtener pulsos desfasados que permitieran el movimiento del rotor convirtiendo su control de velocidad ya no por la frecuencia, sino por CC y así entonces controlar el motor.

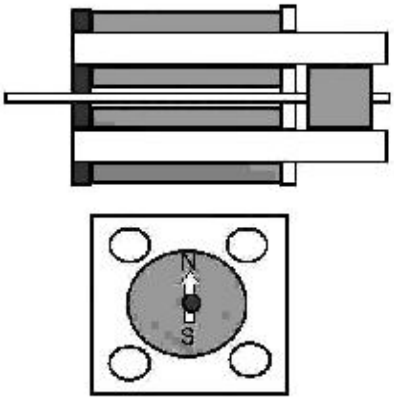


Figura. 4.4 Motor de pasos con 2 pares de bobinas

El diseño consta de dos etapas de potencia que se relacionan con una primera bobina. Cada una de ellas es

controlada por inversores analógicos, en los cuales se utilizaron transistores Darlington, y arreglos de resistencias en configuración de inversor analógico, el corrimiento de la señal se originó debido a otro arreglo tipo RC, con el cual se calculó el valor del capacitor y de la resistencia para desfasarlo 90°. Este sistema es alimentado por el voltaje que ocasiona que la segunda bobina origine el efecto de generación de voltaje, al cortar las líneas de campo magnético con un impulso mecánico en su rotor, el circuito electrónico se muestra en la figura 4.5.

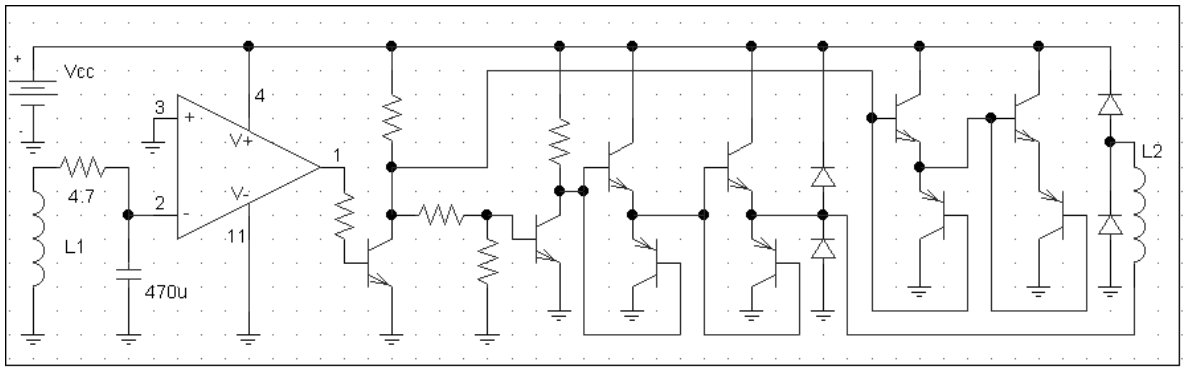


Figura 4.5 Circuito de control del motor de CC

El diseño anterior se realizó con el software PSpice para la simulación del circuito. Con este avance, mas tarde se logró obtener en forma física el control del motor de pasos que trabajara con corriente directa, pero con la principal dificultad que al iniciar su secuencia requiere un impulso mecánico en su rotor.

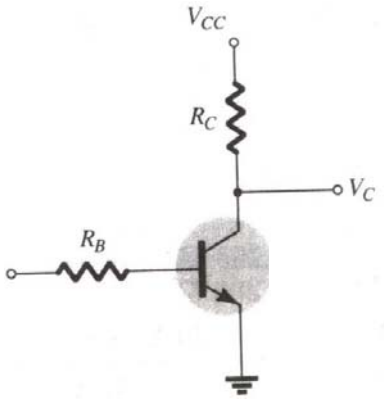


Figura.4.6 Transistor Inversor

La aplicación de los transistores no se limita solamente a la amplificación de las señales. Por medio de un diseño adecuado pueden utilizarse como interruptor para aplicaciones de control. Como se observa en la figura 4.6 la única fuente de CC esta conectada al extremo del colector, siempre y cuando no tenga una fuente de CC a la base del transistor. Estos son utilizados como inversores lógicos que trabajan de modo en que el punto de operación de operación cambie

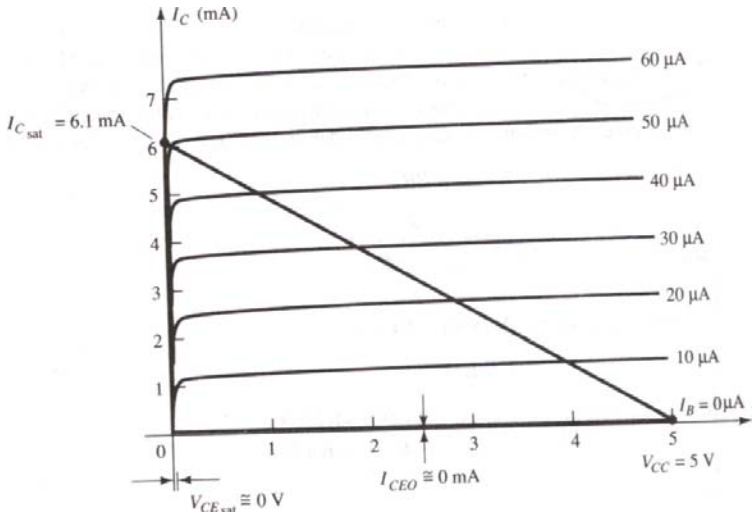


Figura 4.7 Recta de carga del TBJ

desde el estado de corte hasta el de saturación, a lo largo de la recta de carga como se observa en la siguiente figura 4.7, para que el transistor este en estado encendido se debe asegurar que este saturado con un nivel de corriente de la base (I_b) cerca del nivel de saturación; el nivel de saturación para la corriente de colector (I_C) del circuito se define como:

$$I_{C_{sat}} = \frac{V_{CE_{SAT}}}{R_{SAT}}$$

Para que el transistor tenga un estado de saturación o de corte, influye la corriente de base (I_b), dado que si la corriente (I_C) es alta, el voltaje de colector-emisor (V_{CE}) es bajo, ocasionando el estado de saturación, donde el voltaje del colector (V_C) será aproximadamente cero y la impedancia entre el colector y emisor también será próxima a cero. Por otra parte para que existan condiciones de corte la impedancia entre colector y emisor tiene que ser grande debido a la escasa corriente de la base ocasionando que el voltaje en (V_C) sea cercano al de la fuente; mientras que, la corriente de base es baja, el voltaje (V_{CE}) es alto.

Cabe destacar que para un transistor en estado de encendido el voltaje de V_{BE} debe situarse en aproximadamente 0.7 V.

CONTROL DEL MOTOR A LAZO CERRADO

Dado que el microequipo contaba con motores de pasos controlados a través de una PC, se catalogarían como un control de

lazo abierto, control directo y control por programación como se explico en el capítulo dos.

Para tener un control a lazo cerrado necesariamente surge la retroalimentación, para ello se diseñó una etapa de sensado compuesta por LED's emisores de luz infrarroja y fototransistores; además de un disco que tendría la posibilidad de interrumpir la luz emitida por estos sensores (encoder) retroalimentando un par de señales que indicarían la posición del rotor.

La etapa de potencia del motor de pasos, suministraría voltaje y corriente a las bobinas, y con ello no se tuviese caídas de tensión. Para ello se utilizó el software PSpice, en la simulación de la etapa de potencia.

La información del encoder es transmitida a una amplificación donde se corrige para después en una etapa de potencia pueda manejarse un frecuencia sin traslapes de fases, obteniendo altas velocidades.

Cabe destacar que para velocidades bajas el motor no contiene par considerable, debido a cantidad de voltaje en esas velocidades es escaso, dado que es cercano a los 5 volts.

CONTROL ELECTRÓNICO

Dentro de esta etapa del proyecto se diseñó y construyó un motor, el cual más tarde seria llamado bimodal dado que es referenciado a que dicho motor puede llegar a ser controlado de dos formas: a modo de lazo cerrado, y a modo de lazo abierto.

Con el avance de la investigación se decidió nombrar a este tipo de control junto con el motor a pasos, trabajando a CC, como *motor bimodal* (figura 4.8), que como su nombre lo indica tendrá dos posibilidades de funcionamiento, ya sea en modo a pasos o alimentado por CC; para prescindir del impulso mecánico, se realiza el *control bimodal*, el cuál comprende el control de lazo abierto (pasos) o el control de lazo cerrado (CC).

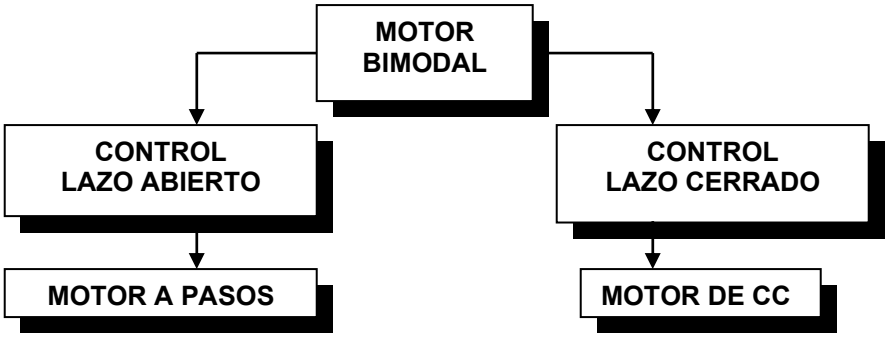


Figura 4.8 Diagrama de bloques del control del motor bimodal

El motor cuando se controla a lazo abierto, presenta una funcionalidad normal de modo de pasos, en la que se emiten pulsos generados de algún sistema lógico y por medio de una etapa de potencia controla las fases del este. El sistema de control de lazo abierto es ideal para situaciones en que operaran los motores a bajas velocidades y cargas estáticas. No existe una retroalimentación, y la posición solo es conocida mediante simulaciones.

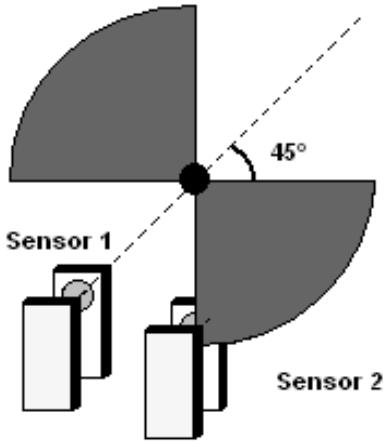


Figura 4.9 Esquema del encoder

El control a lazo cerrado es un método muy efectivo, libre de inestabilidades y capaz de acelerar rápidamente y con ello buscar altas velocidades.

El motor en modo de operación en corriente directa requirió del desarrollo un circuito con características similares al anterior para su otra fase; y la sustitución del impulso para cada fase se facilitó con un encoder como el mostrado en la figura 4.10, donde este tiene sectores cortados a 90° encontrados 180°; mientras el motor gira con el disco, la máscara pasa y periódicamente bloquea la luz monitoreada por un par de sensores, compuestos por LED´s infrarrojos y fototransistores en donde se obtiene una información digital como el mostrado en la figura 4.9, que nos indicará la posición del rotor, estas señales

digitales son transmitidas por un amplificador operacional en configuración de comparador mejorando los pulsos que emitieron los fototransistores.

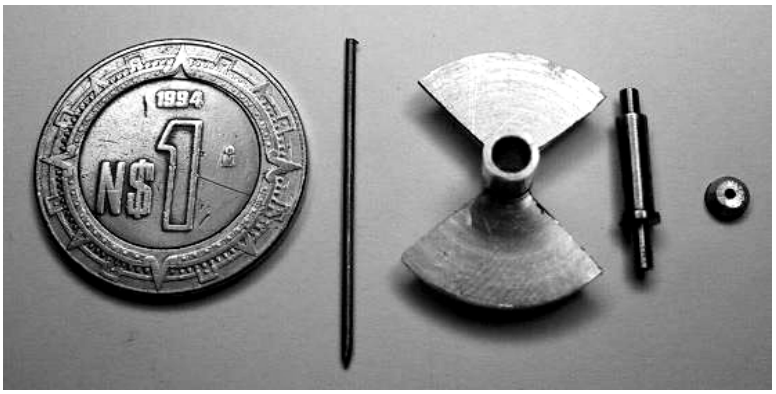


Figura. 4.10 Partes del encoder

Al girar el encoder se encienden y apagan los sensores 2 veces por cada revolución, por ello, para que la secuencia de pulsos tenga una defase de 90° se tiene que reducir al doble el giro del rotor con respecto al del disco, se utilizó un juego de engranes del doble como se muestra en la figura 4.11.

Por ello, estos motores alimentados con CC se catalogan por su control como a lazo cerrado; porque emplean una retroalimentación tomada del rotor, proporcionando su posición.

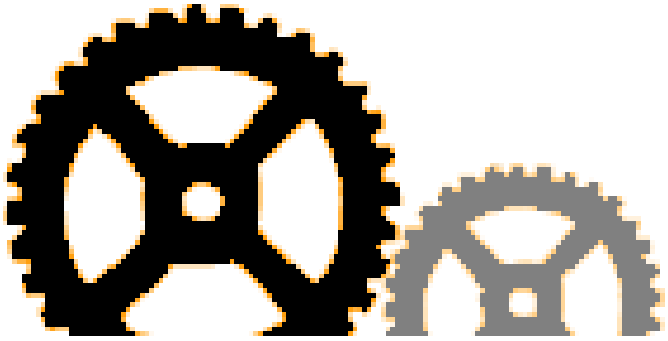


Figura. 4.11 Aumento en engranes

Es importante la forma en que debe de ser el acoplamiento del encoder con el rotor, dado que se deben acoplar las señales que alimentan a las bobinas con el apoyo de la posición del rotor, de tal manera que tengan 45° de separación el par de sensores y que el eje del rotor corresponda a un defase de 90° (figura 4.12) para que entre las dos señales A y B al ser emitidas con una etapa de potencia hagan girar al rotor del motor.

OPTOELECTRÓNICA Y SENSORES

La forma de caracterizar la posición del rotor, se llevó a cabo con sensores como los mostrados en la figura 4.13, ellos constan de un

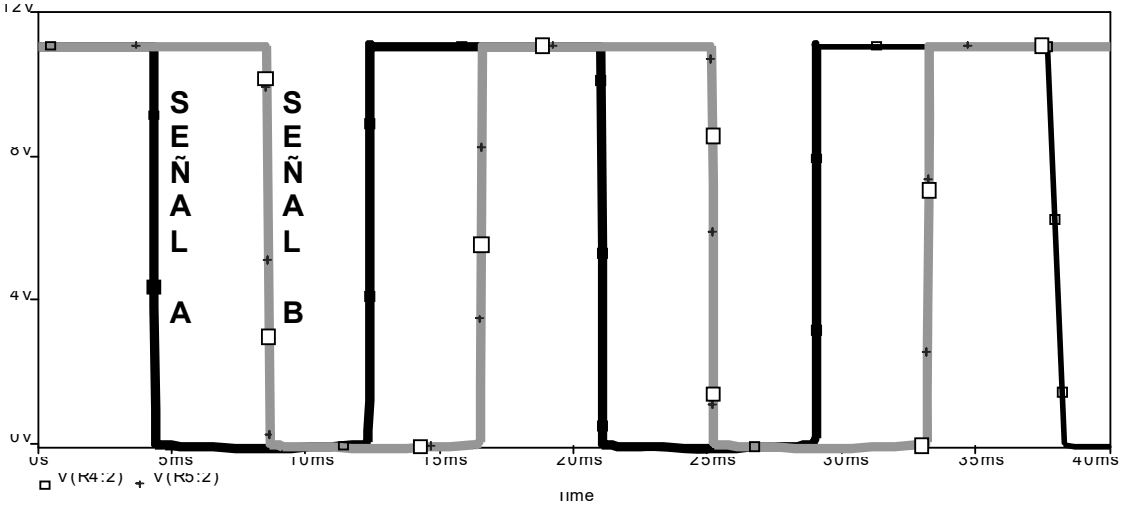


Figura 4.12 Señales A y B de los fototransistores desfasadas 90°

sistema compuesto por un LED que emitir luz infrarroja en una sola dirección, y un optosensor que pueda emitir una señal que indique la posición del encoder acoplado al rotor.

La forma de alimentación de los LED's infrarrojos se considero que fuese con un divisor de tensión a 2.5 [V] y como protección una resistencia en serie.

De fototransistores se obtuvo una salida con la cual se aplicó a un amplificador operacional LM324 (figura 4.14) el cual tiene la

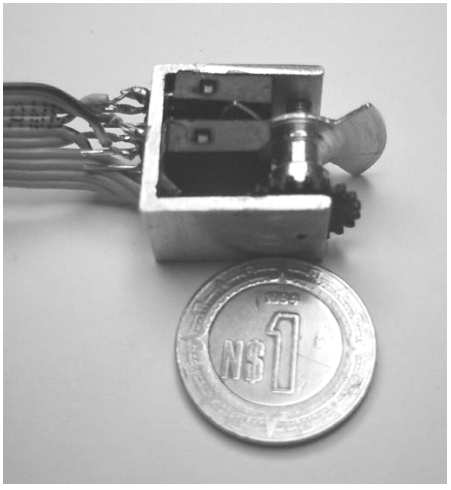


Figura. 4.13 Encoder con sensores

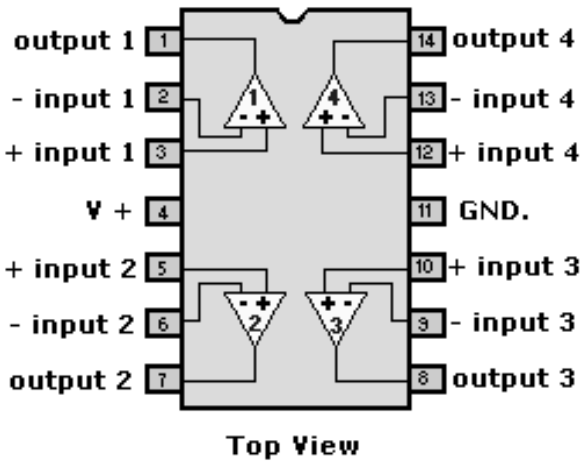


Figura 4.14 Amplificador Operacional LM324

posibilidad de amplificar señales no digitales que el optosensor emite, dado que su respuesta es de forma analógica, por tiempos de subida y bajada, en cambio, el amplificador operacional se consideró con una configuración de comparador, en la que con un voltaje de referencia, pudiese emitir una señal completamente digital y un voltaje en el que satisfaga las necesidades de retroalimentación. (Figura. 4.15).

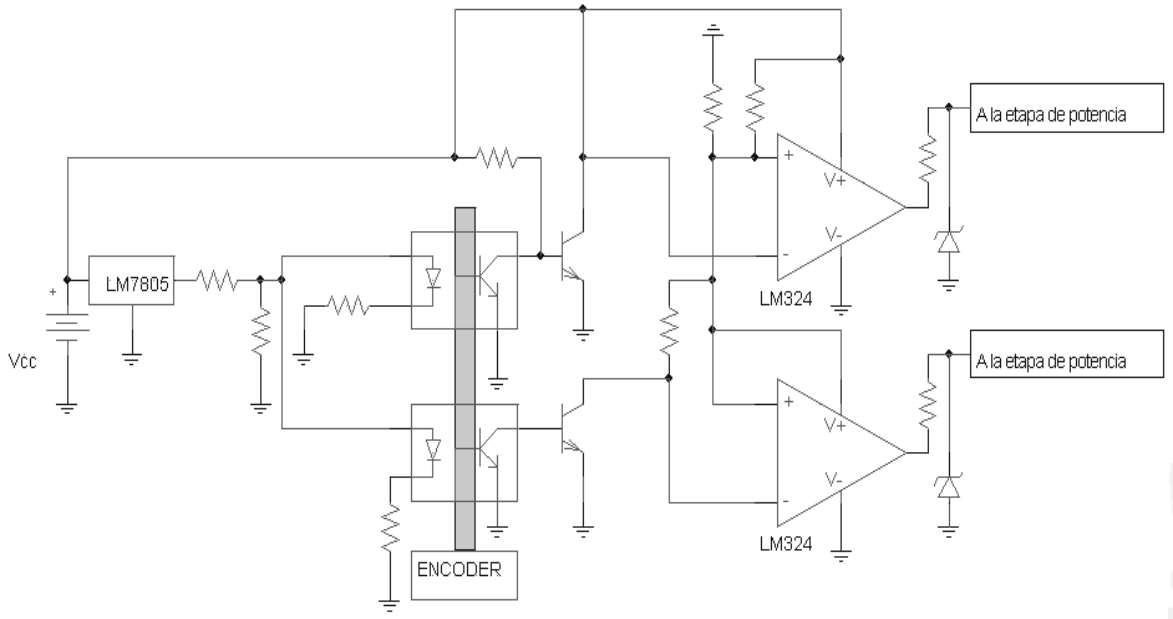


Figura. 4.15 Circuito electrónico de sensado del encoder

Para la construcción de la etapa de potencia y sensado en circuito impreso se montó previamente el circuito en “protoboard” y al obtener resultados satisfactorios, se fabricó en circuito impreso, tanto la etapa de potencia del motor como la etapa de sensado; y para ello fue necesario el uso del software P-CAD para reducir en medida de lo posible sus dimensiones. Figura 4.16

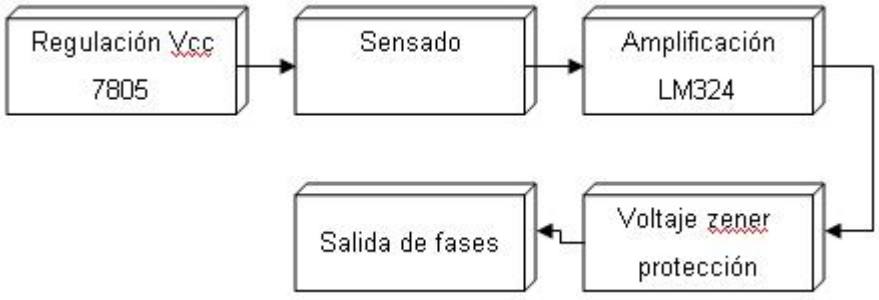


Figura 4.16 Diagrama de bloques de la etapa de sensado

ETAPA DE POTENCIA

Esta etapa es la encargada de suministrar el voltaje necesario a la señal emitida por la etapa de control, dado que recibe una señal de bajo nivel de voltaje, insuficiente de alimentar una bobina del motor. Así una etapa de potencia robustece de corriente y voltaje a una señal.

La inductancia del motor de pasos es el embobinado, y para alimentarlo se debe tener un nivel de corriente determinado, de tal manera que optimice el rendimiento del motor. El par de un motor de pasos depende del flujo de corriente suministrada por la etapa de potencia, mientras que la velocidad dependerá de la frecuencia a la que trabaje.

Para este tipo de motor se propusieron varios circuitos que tuviesen una etapa de potencia para robustecer la señal retroalimentada, originando un motor controlado a lazo cerrado.

Primer diseño.

El primer diseño consta básicamente de transistores Darlington en

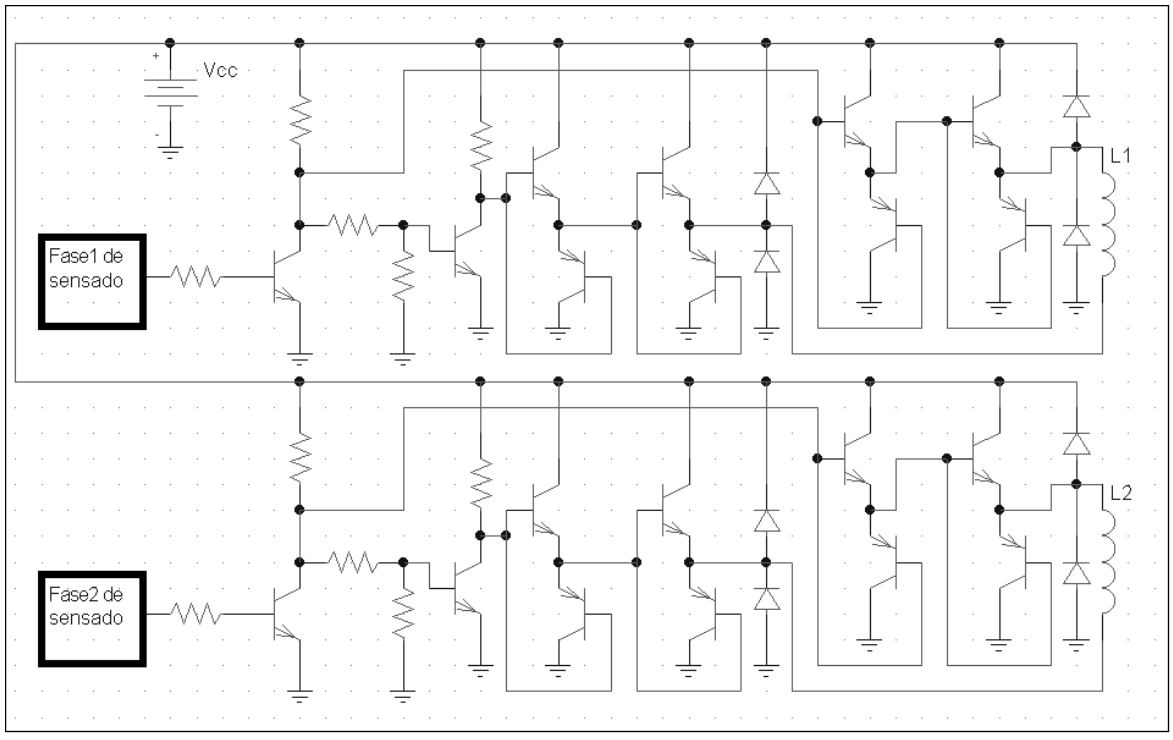


Figura 4.17 Etapa de potencia con transistores tipo Darlington

cascada en los cuales puedan suministrar la suficiente potencia a los embobinados, el sistema de encendido y apagado de las bobinas se realiza con transistores utilizándolos en forma de inversores analógico, como se muestra en la figura 4.17. Una de las experiencias que se tuvo con este circuito es que los TIP 41 y TIP 42 fueron relativamente costosos y el circuito final en impreso es muy grande, además es únicamente para la etapa de potencia de un motor.

Segundo diseño.

A diferencia del anterior se utilizaron transistores bipolares de menor costo que los TIP's y de un tamaño menor aunque no tan comerciales por que se buscaron en el mercado, de tal manera

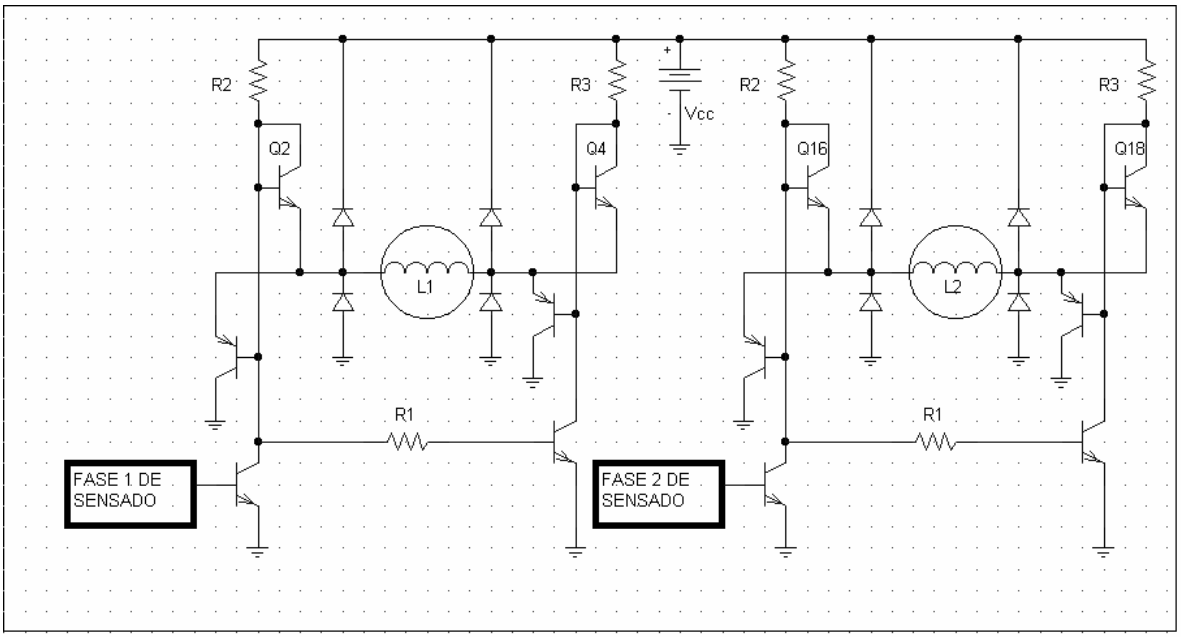


Figura 4.18 Etapa de potencia tipo puente H

fuesen capaces de soportar por lo menos un ampere de corriente (D1246 y A1246). Una de las de experiencias que tuvo de este circuito fue que al colocar todos los transistores en el PCB realizado en P-CAD, se presentó un calentamiento al dejarlo trabajar por algunos minutos en modo de lazo cerrado, aunque estuvieran dentro de su rango de corriente alcanzando una temperatura mayor a los 60°C que es la temperatura promedio que soportan los dispositivos electrónicos de estas características; cabe destacar que no se utilizaron disipadores de calor para no utilizar mas espacio. Figura 4.18.

Tercer diseño.

Se procedió en el último diseño a trabajar con el circuito integrado L293 conformado por etapas de potencia, que generalmente se utiliza para motores de pasos, este modelo es muy comercial debido a que la etapa de potencia proporciona la corriente suficiente para un buen par. Figura 4.19.

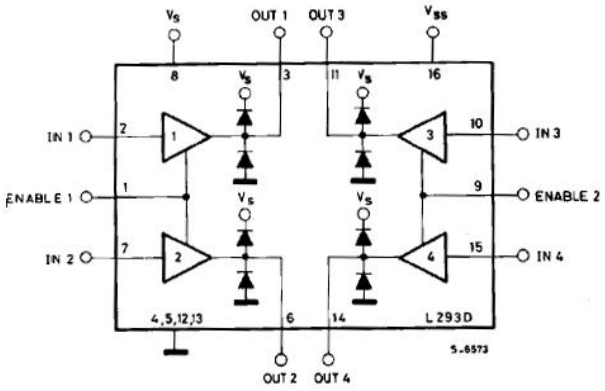


Figura 4.19 Etapa de potencia con el CI L293

Principalmente se tuvieron dos características por las cuales se decidió escogerlo como mejor diseño electrónico:

a) Dimensiones muy por debajo de sus antecesores, y

b) Bajos recursos financieros en su construcción.

Este chip trabaja con dos voltajes, Vss y Vcc. Con la configuración que se muestra en la figura 4.20 se observo que sería de mucha ayuda por la independencia que tiene cada fase, con ello y con un sistema de control de lógica programable, se puede apagar el motor con las cuatro fases a tierra.

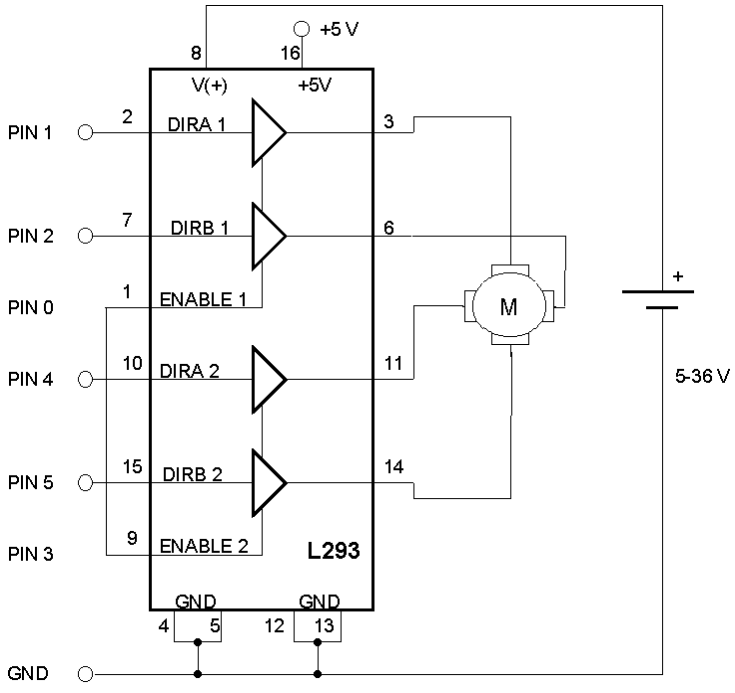


Figura 4.20 Configuración de la etapa de potencia

Una de las especificaciones que contiene este circuito integrado es su tamaño, excelente para los requerimientos que debe llevar el prototipo de control de todos los motores de la microcélula de producción.

Hecho lo anterior se prosiguió a reducir el tamaño de dicho motor dado que este mismo llegaría a estar montado en microequipo, entonces obligaría a reducir notablemente sus dimensiones.

Fue necesario al igual que con los demás prototipos diseñarle una etapa de potencia la cual tendría el mismo objetivo que las anteriores.

Diseño electrónico en MAX+PLUS II del control del motor bimodal. Al momento de realizar la grabación del circuito en la tarjeta se necesito conocer el software proporcionado por Altera. Entre sus diversas formas de programación de la UP1 se decidió por el modo gráfico, por sencillo y rápido.

A pesar de que el software de Altera maneja la opción de simular el circuito lógico, tiene la deficiencia de que, para analizar el comportamiento de las señales binarias generadas por dicho diseño electrónico, tiene una resolución únicamente de 1μs. Por ello se decidió emplear un software con mayor resolución en cuanto a frecuencia y tiempo de simulación, por tanto se opto por Electronic Workbench. En él, se elaboró el circuito que controla al motor bimodal en lazo cerrado y en lazo abierto por medio de contadores, flip-flop's, multiplexores, inversores, AND's y OR's; para más tarde copiarlo al software de Altera, el cual cumplía con los requerimientos establecidos de control, los cuales contienen bloques como los mostrados en la figura 4.21

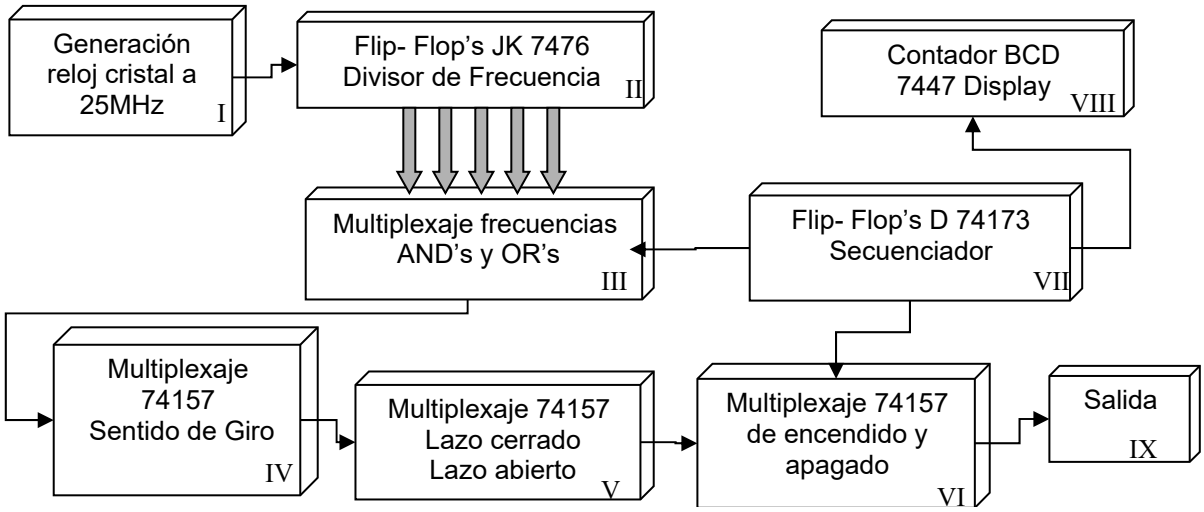


Figura 4.21 Diagrama de bloques del diseño electrónico en Max+PlusII

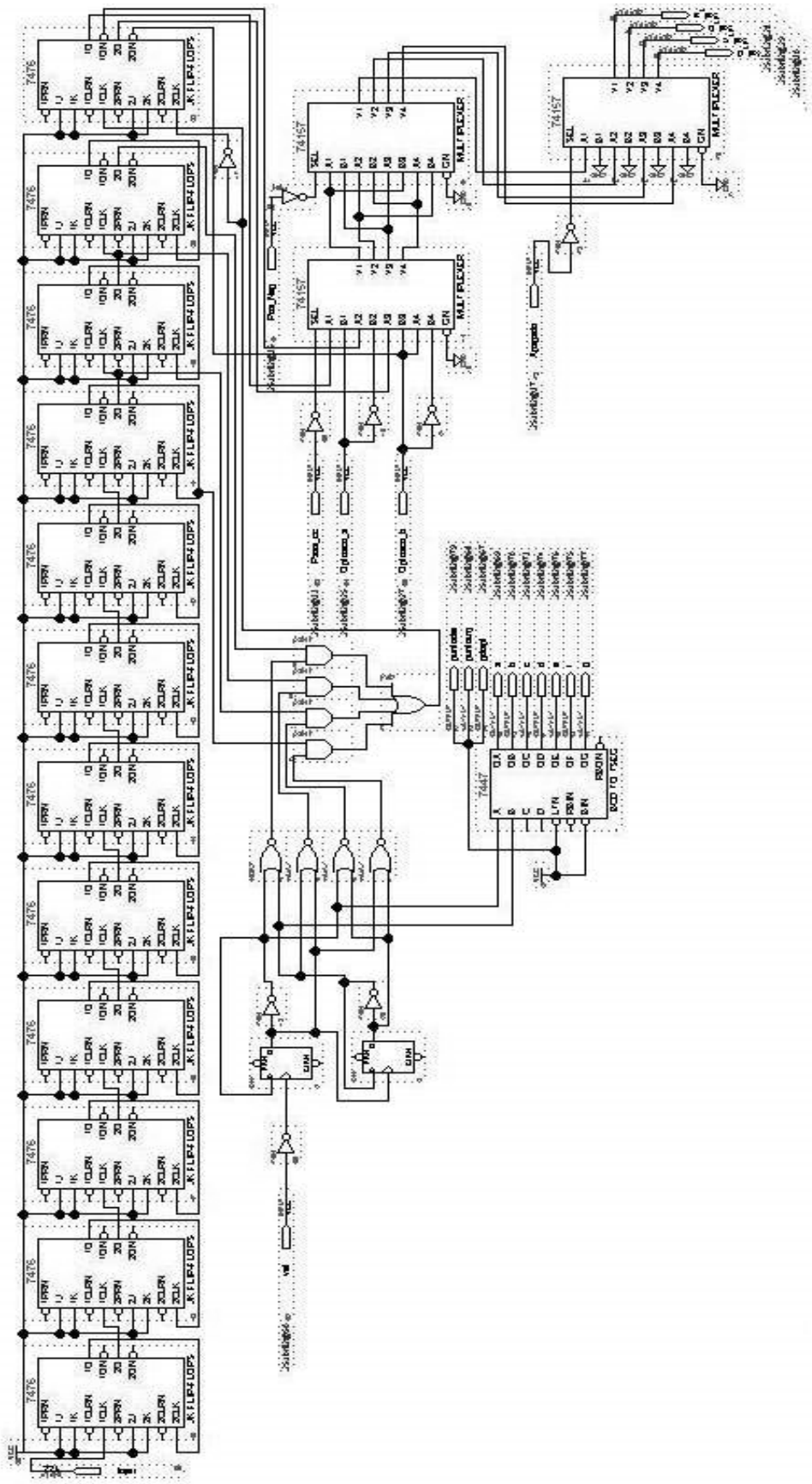


Figura 4.22 Diagrama gráfico en MAX+PLUS II del control de motores bimodales

Explicación del diagrama en MAX+PLUS II (Figura 4.22)			
I	La tarjeta UP1 proporciona un reloj a 25Mhz que es utilizado como entrada de pulsos	VI	Es la etapa de selección, donde por medio de la presencia de un pulso se permite emitir las señales
II	Con una conexión en serie de 12 flip flop's se obtiene un divisor de frecuencias	VII	Secuenciador lógico para cambiar de frecuencia y enviar la información binaria al BCD
III	Es donde se escoge la frecuencia deseada con AND y OR	VIII	El BCD 74173 es un decodificador para los display de 8 segmentos
IV	Es donde se escoge por medio de un pulso el sentido de giro	IX	Salida
V	Es donde se escoge por medio de un pulso si trabaja a lazo cerrado o a lazo abierto		

Tabla 4.1 Explicación del diagrama esquemático de Max+Plus II

Programación en Borland C++ Builder 4.

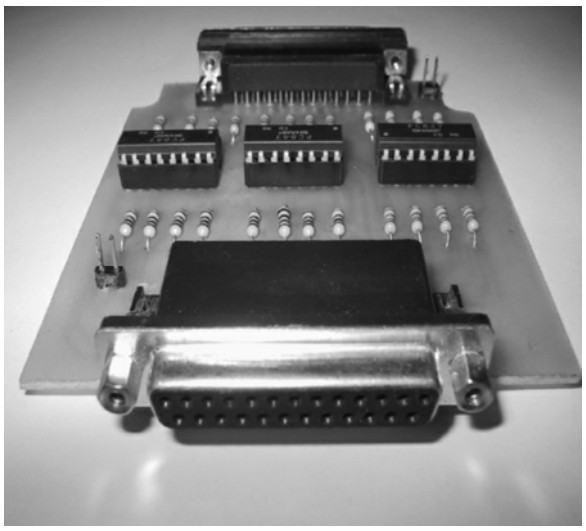


Figura 4.23 Optoprotección del puerto paralelo

Para que el FPGA tenga una mutiplexación entre control a lazo cerrado, lazo abierto, sentido, velocidad y número de pasos dados de cada motor, fue necesario alimentarla con señales de entradas binarias de 5 V.

Este tipo de alimentación se decidió que fuera proporcionada por una PC de escritorio, dado que sería una buena interfase por la facilidad de emisión de pulsos a través del puerto paralelo (LPT1 y LPT2) hacia los conectores del CPLD EPM7128S.

Cabe mencionar que para proporcionar la información lógica es necesario el uso del programa en Borland C++ Builder 4 y el cual tendrá justamente la tarea de seleccionar la formar de emisión de las señales requeridas.

Para la protección del puerto paralelo de la PC se requirió protegerlo de cualquier transitorio generado durante las pruebas, de esta manera se fabricó en Protel el circuito impreso de una tarjeta de protección para dicho puerto, la cual tiene como principal elemento: protecciones ópticas (PC847), el mostrado en la figura 4.23.

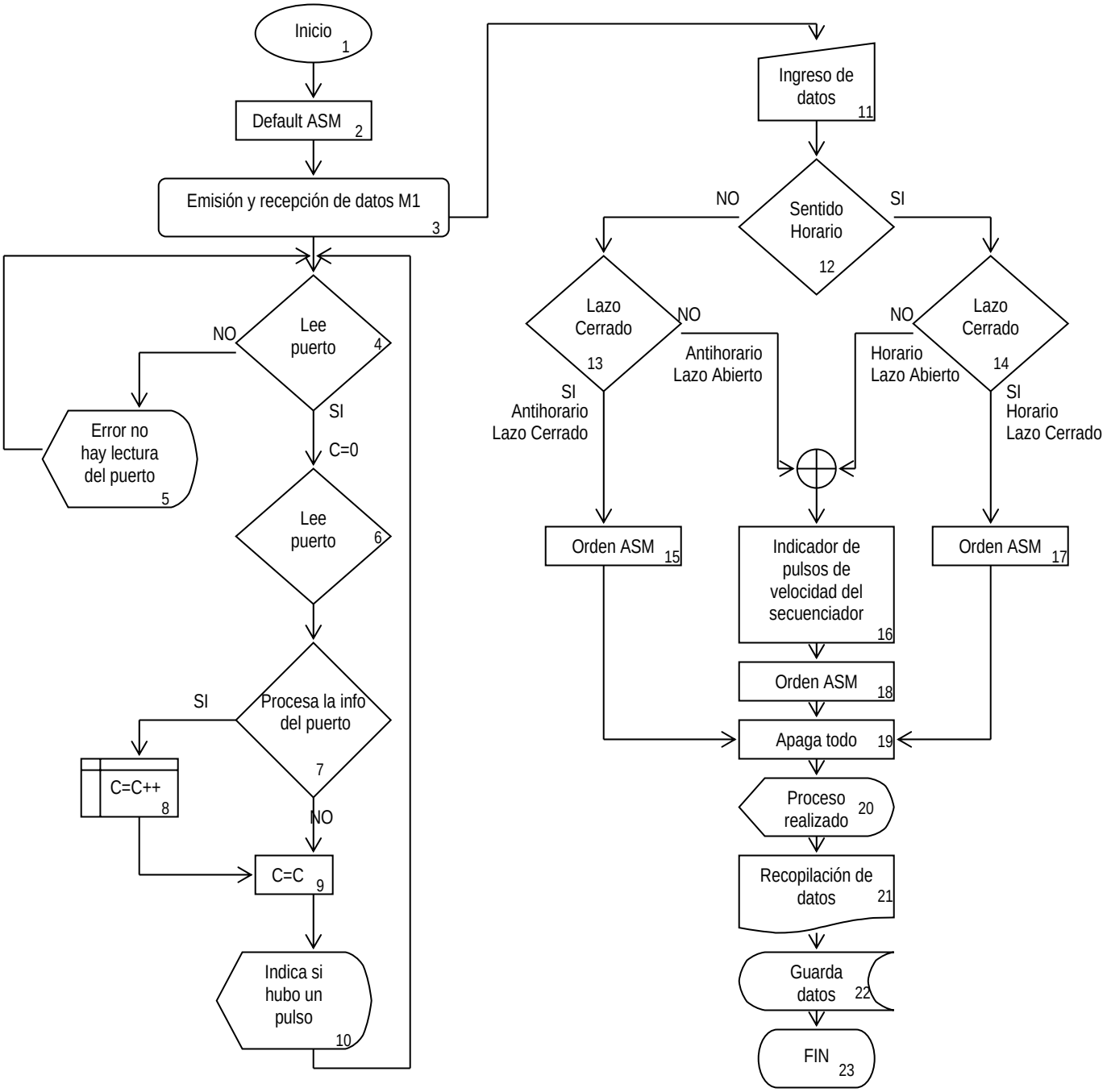


Figura 4.24 Diagrama de flujo para el control de un motor bimodal

El programa orientado objetos, que se realizó, controla dos motores, en el que se genera información del puerto paralelo (ya sea LPT1 o LPT2) previamente instalado, en el que se escoja la dirección del puerto y pueda leerla, ya sea binaria o hexadecimal, y sea necesaria para que en un ciclo “FOR” pueda manipular la información, y saber si el motor ha realizado un movimiento así como su dirección; posee la cualidad de que el usuario determine el modo de funcionamiento del motor bimodal (lazo cerrado o lazo abierto), puede además configurar el sentido en que el rotor girará; cabe resaltar que el mismo software tiene la posibilidad de regular la velocidad del motor, a la que el usuario requiera en modo de lazo abierto, que es el momento en el que se desea mayor precisión.

Explicación del diagrama de flujo del programa			
1	Inicializa el programa	13	Determina en caso de tener un sentido antihorario si trabaja a lazo cerrado o a lazo abierto
2	Reset del programa	14	Determina en caso de tener un sentido horario si trabaja a lazo cerrado o a lazo abierto
3	Emisión y recepción de datos de los motores al programa	15	Envía la información al puerto, de trabajar a lazo cerrado/antihorario
4	Realiza la comprobación de dispositivo conectado al LPT	16	Indica la cantidad de pulsos que se enviaran para determinar la frecuencia en lazo abierto
5	Imprime “Error de lectura de puerto”	17	Envía la información al puerto, de trabajar a lazo cerrado/horario
6	Inicializa variable y lee el puerto LPT	18	Envía la información de trabajar a lazo abierto/horario o antihorario
7	Procesa la información suministrada en el puerto LPT	19	Mediante un pulso se determina el encendido o apagado del motor
8	Incrementa la variable una unidad	20	Paro manual del trabajo del motor
9	Si no existe la presencia de un pulso, la variable permanece igual	21	Recopila y muestra la información en un cuadro de texto del programa
10	Imprime “la cantidad de pulsos acumulados”	22	Guarda la información obtenida a lo largo de la ejecución del programa
11	Personalización del trabajo del motor (modo, sentido y velocidad)	23	Finalización del programa
12	Elección de giro en sentido horario o antihorario		

Tabla 4.2 Explicación del diagrama de flujo

El circuito electrónico de la protección se muestra en el anexo 4. La forma de trabajo del software únicamente tendrá la oportunidad de sustituir cuatro interruptores externos del FPGA y pueda cuantificar en número de pasos que haga un motor, tratando de utilizar al mínimo el uso de la PC.

Los interruptores que se mencionan son los que controlan la retroalimentación, sentido de giro, indicación de cual motor debe trabajar, y la velocidad que debe de trabajar el motor en lazo abierto. Su diagrama de flujo quedó expresado como se muestra en la figura 4.24. Una forma de representar al motor bimodal se muestra en la figura 4.25.

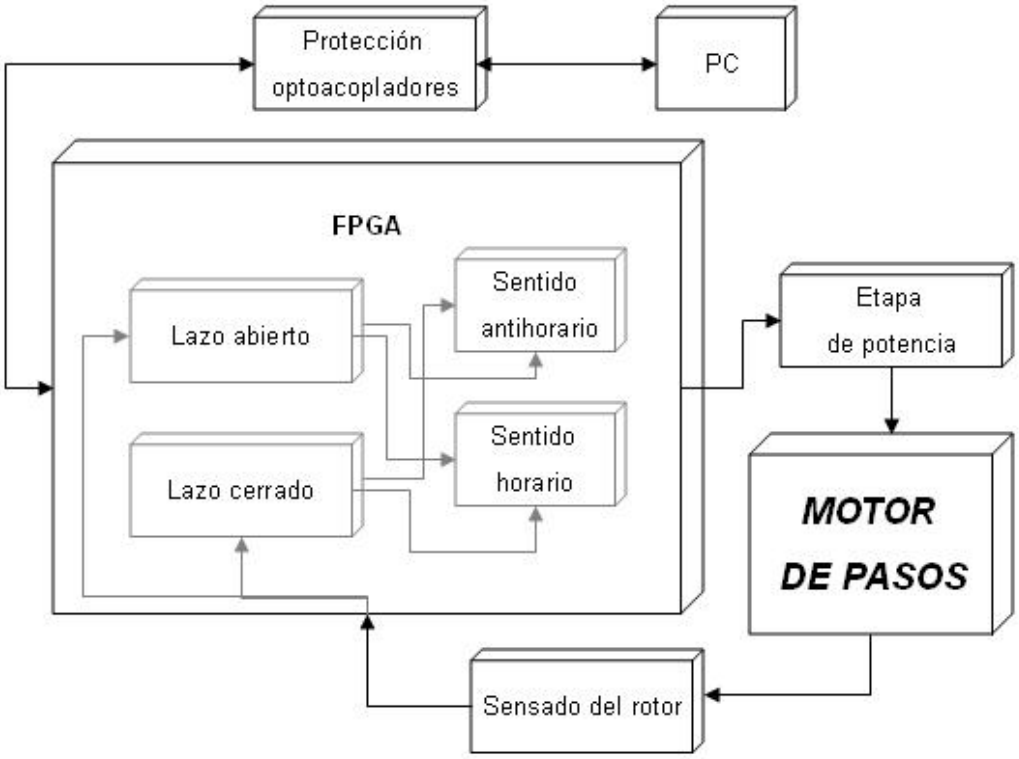


Figura 4.25 Diagrama de bloques del motor bimodal

MINIATURIZACIÓN DEL MOTOR Y DEL SENSADO.

Cuando se obtuvieron las r.p.m. esperados con los motores y la etapa de sensado, se continuó con la reducción de sus dimensiones.

Del primer prototipo de motor de pasos, se detectaron problemas de operación relacionados con el modo dinámico. Estos motores tenían un momento de inercia alto, por lo que fue necesario trabajar en la reducción del mismo. Para esto, se diseñó un rotor con un menor momento de inercia, reduciendo éste alrededor de dos tercios comparado con el primer diseño

Finalmente se obtuvieron dimensiones del motor y etapa de sensado, por debajo 4x4x4 cm^3, como se muestra en la figura 4.26.

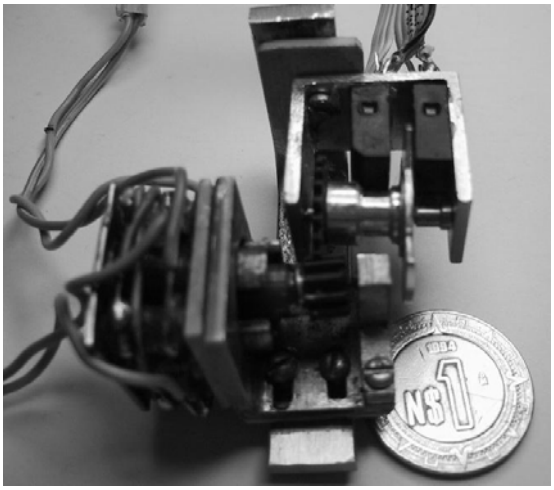


Figura 4.26 Miniaturización del motor y del sensado

PRUEBAS Y RESULTADOS

Capítulo 5

En esta sección se muestran tanto gráficas e imágenes de los resultados de las pruebas al motor bimodal, así como de la ejecución del programa de control hecho en C++.

El trabajo anteriormente descrito, resultó ser una solución para controlar los motores de pasos que se utilizarán en el microequipo de una microfábrica del LMM. Una de las pruebas que se realizaron fue el trabajar con el motor de pasos utilizando un control a lazo cerrado, donde su retroalimentación ayudaría a comprobar el aumento de velocidad, con respecto al del lazo abierto que se trabajaba antes. El motor mostrado en la siguiente figura 5.1 es uno de los primeros modelos que se desarrolló.

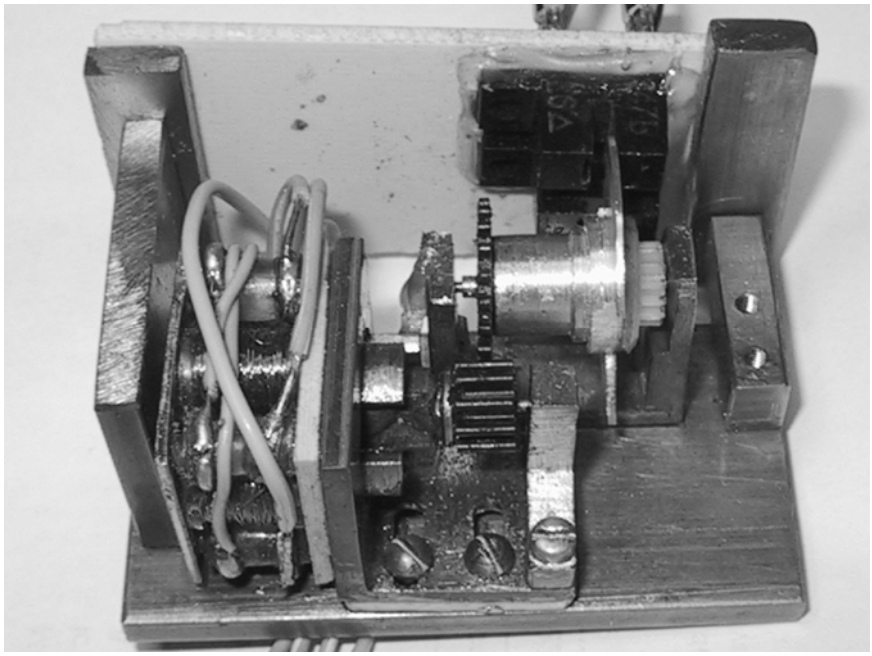


Figura 5.1 Primer prototipo del motor bimodal

Al estar hablando del motor a lazo cerrado las variables influyen en el voltaje y también en r.p.m. que el motor genera; la gráfica que se muestra en la figura 5.2 comprende los datos en donde se trabajó con un rango de voltaje entre 2 y 23 volts, obteniendo hasta 28,000 revoluciones por minuto, y un mínimo de 10,000 que, donde con las r.p.m. que el mismo motor genera en modo de lazo abierto solo se obtuvieron hasta 11,000 r.p.m. figura 5.3.

El número de r.p.m. máximo y mínimo que se obtuvieron en cada prueba se cuantificaba por medio de su señal digital, enviada a las bobinas en donde se catalogaban si eran aceptables mientras contuvieran un ciclo de trabajo del 50% y un defasamiento de 90° dentro de un análisis en el osciloscopio, además de la observación visual al motor, donde debería girar normalmente.

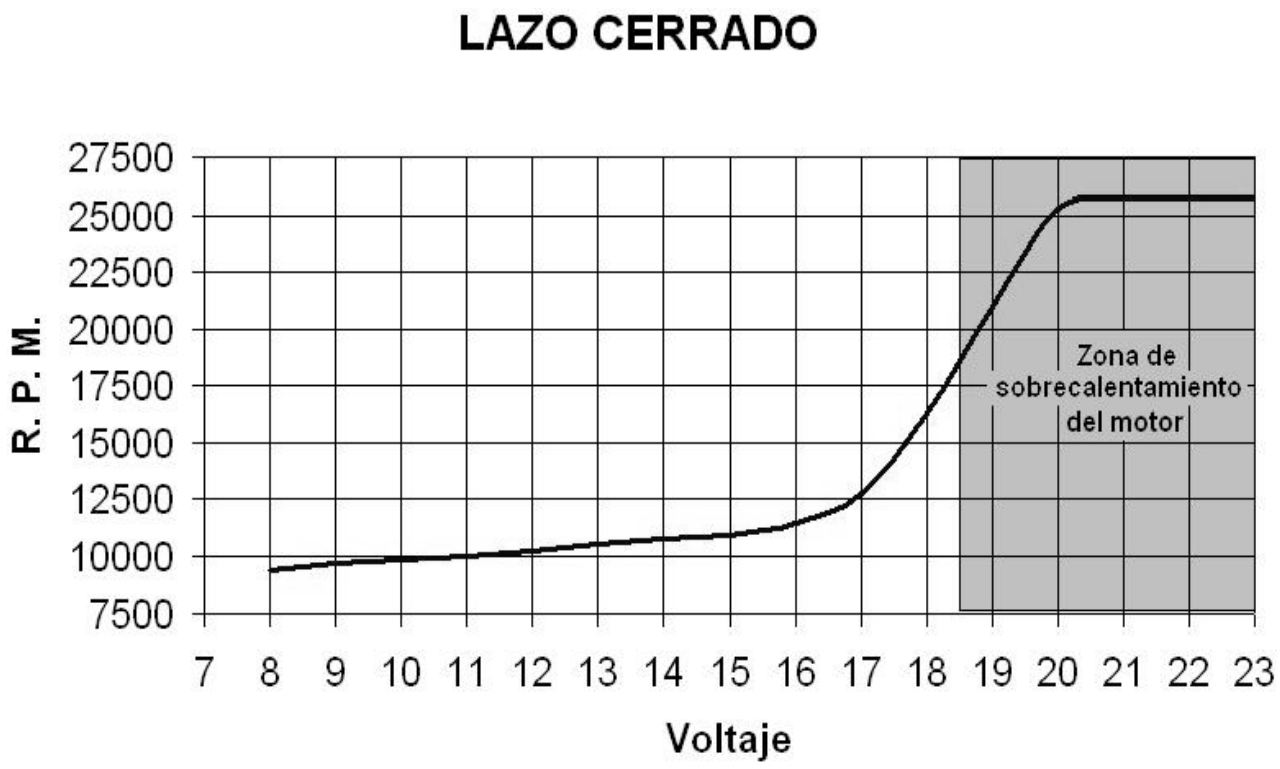


Figura 5.2 Grafica a lazo cerrado de voltaje vs r.p.m. del motor bimodal

Como se puede observar en las figuras 5.2, 5.3 y 5.4, las r.p.m. obtenidas en modo de lazo cerrado y modo de lazo abierto, son señales digitales emitidas por el FPGA, así deslindando el control y generación de la computadora, logrando tener un control indirecto con lógica discreta.

Figura 5.3 Grafica a lazo abierto del periodo vs r.p.m. del motor bimodal

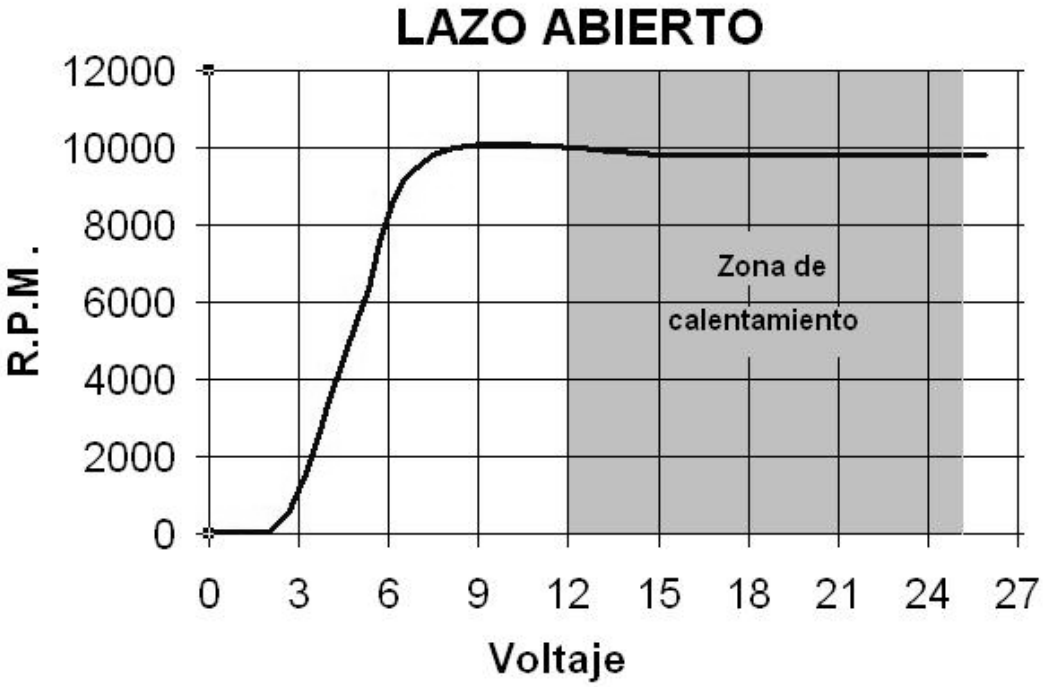


Figura 5.4 Grafica de lazo abierto de Voltaje vs r.p.m. del motor bimodal

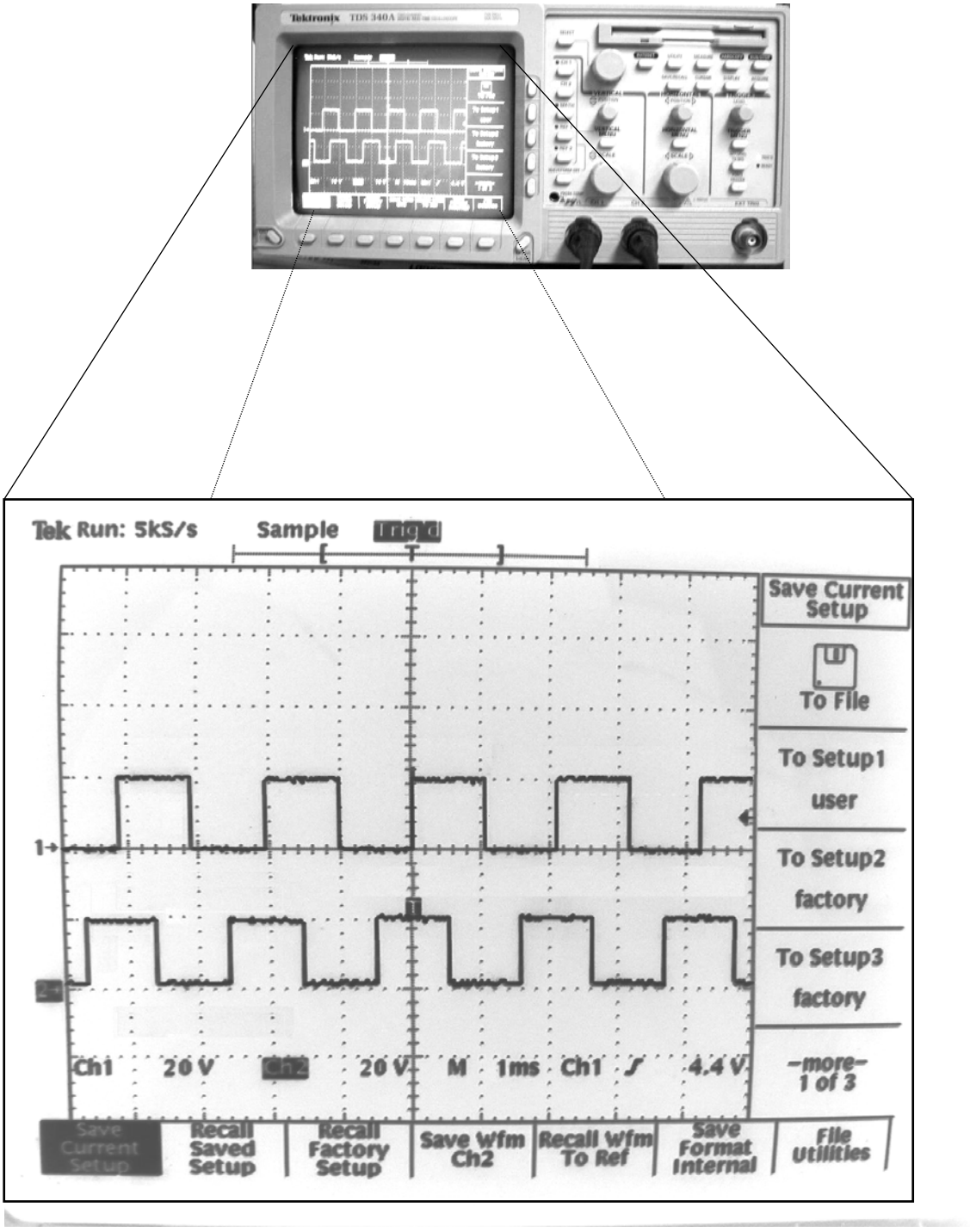


Figura 5.5 Señales defasadas 90° del motor bimodal a lazo cerrado

Cabe resaltar en las graficas mostradas en la figuras 5.2 y 5.4 se muestra la zona específica donde el motor se sobrecalienta, en donde

se toma como referencia para catalogarlo así cuando se podía tomar aún con la mano, siendo este un rango aproximado a los 60 °C.

También, se identifica en la gráfica de la figura 5.3 a lazo abierto la zona de precisión y la de inestabilidad donde el motor empieza a fallar mecánicamente por el traslape de pulsos enviados de información, debido a la alta frecuencia.

Las señales generadas siempre se catalogaron y se cuantificaron en el osciloscopio, así se garantizaba que no había pérdidas de información en altas revoluciones, como lo es para el caso en el modo de lazo cerrado, y que la forma de saber el voltaje máximo que soportaba los motores era al observar que no se reducía mas el periodo.

En la figura 5.5 se observan las señales que alimentan a un motor con retroalimentación, obteniendo un periodo de 2.1 [ms] y un voltaje de 20 volts, defasadas 90°, que es precisamente el nivel máximo de 28,500 r.p.m. que se pudo obtener.

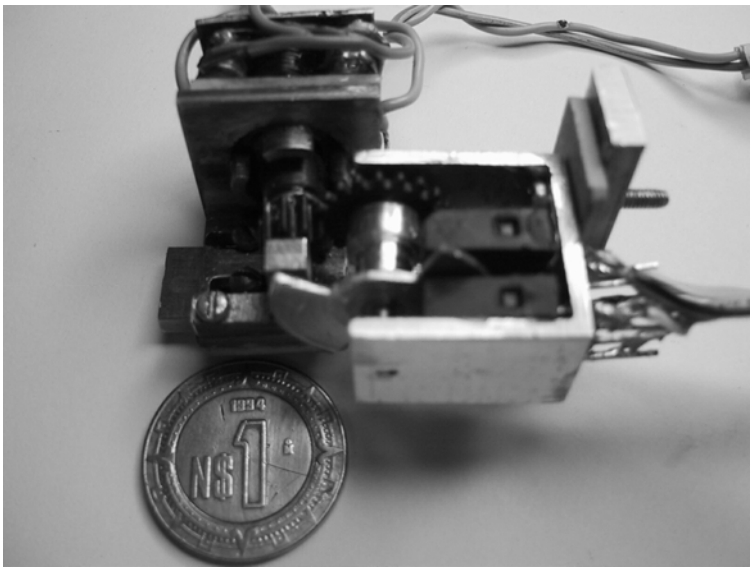


Figura 5.6 Motor de pasos bimodal

El motor de pasos bimodal finalmente diseñado cuenta con una dimensión aproximada de 4x4x4 cm^3, la superficie utilizada en circuito impreso por cada motor es de 4x3 cm^2, el sistema de sensado es de 2x1x0.5 cm^3. Figura 5.6.

Tomando en cuenta al motor de pasos, su etapa de potencia, su sensado, se podría hacer una estimación de costos, los cuales fueron de aproximadamente de 5 dólares por conjunto.

Los resultados obtenidos en la computadora fueron satisfactorios, dado que con las lecturas y el control de interruptores, se logró desplegar una pantalla con toda la información necesaria para cuantificar los movimientos del microequipo al que se desea instalar el motor bimodal. En la figura 5.7 podemos observar el archivo ejecutable del programa realizado en C++, el cuál se observa el conjunto de botones para control de los motores, seleccionando el sentido de giro, modo de control, velocidad en lazo abierto y poder obtener la información en tiempo real, de un puerto deseado únicamente con su dirección.

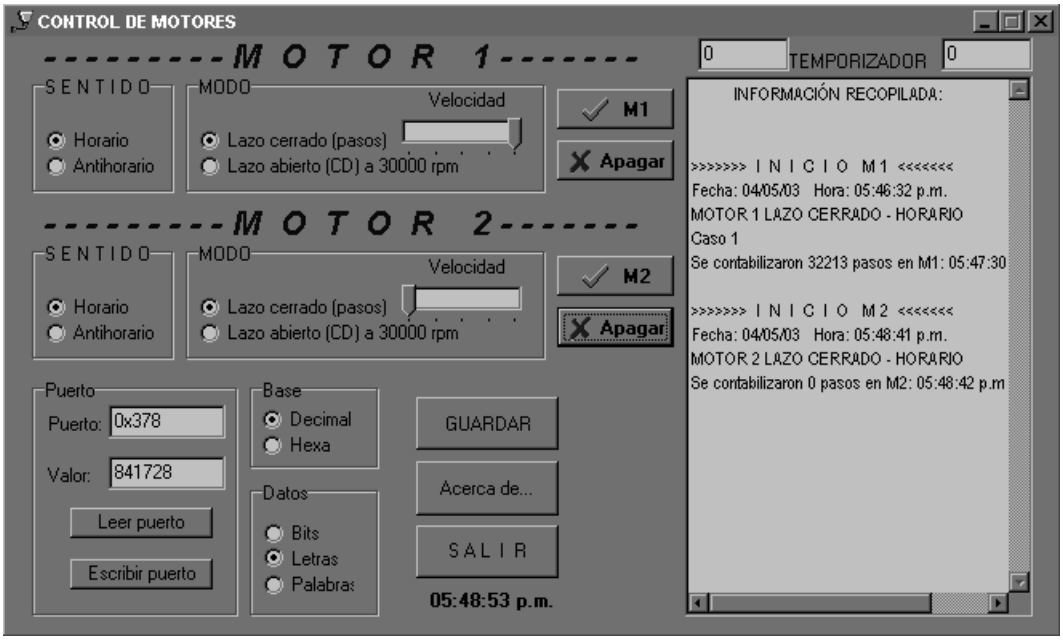


Figura 5.7 Pantalla desplegable del programa en C++.

CONCLUSIONES

Con este trabajo de tesis se consiguió:

- Diseñar e implantar un nuevo sistema de control basado en FPGA's para motores aplicados a microequipo.

- Controlar en modo de lazo cerrado y/o a lazo abierto, el sentido de giro, velocidad y posición de cada motor de forma independiente.

- Miniaturizar la etapa de potencia para cada de motor de forma independiente; así como la etapa de sensado ocupada por el encoder.

- Contar con un motor que trabaje aparoximadamente a 25,000 r.p.m., obteniedo una mayor optimización en comparación a los motores con los cuales se trabajaba en lazo cerrado.

- Implementar un sistema de control a lazo cerrado por medio de un encoder, proporcionando la retroalimentación.

- Generar una señal a partir de la información binaria obtenida por el encoder, mediante una etapa de amplificación y comparación; consiguiendo una señal cuadrada con un ciclo de trabajo del 50%.

- Diseñar un circuito con la posibilidad de generar pulsos a diferentes frecuencias para distintas fases mediante un FPGA.

- Desarrollar un programa orientado a objetos a traves del software de programación C++ , satisfaciendo las necesidades de programación del puerto paralelo (nivel ensamblador).

Cumpliendo con:

- El objetivo planteado al inicio de este trabajo de investigación, donde se emplee un FPGA para el sistema de control de motores de pasos bimodales en micromecánica.

- Las especificaciones requeridas por el LMM, para el control de motores de pasos.

- Tener un trabajo de investigación que sirva como antecedente para la continuación en el desarrollo de la primera microcélula de producción desarrollada en México.

- Contar con una interfaz amigable al usuario, para el control de los motores bimodales.

- Familiarización y conocimiento tanto de los FPGA's como con software de programación, enfocándose en el sistema ALTERA.

TRABAJO A FUTURO

Como trabajo a futuro se plantea la posibilidad de que estos nuevos motores bimodales, conjuntamente con su control; sean implantados en microequipo, formando una completa microcélula de producción de micropiezas. Las cuales mas tarde servirán para construir una generación mas adelantada de microfábricas de producción (cada vez de menores dimensiones).

Un aspecto importante a mejorar en el presente trabajo de tesis, es el diseñar algún tipo dispositivo o el rediseñar en forma mecánica los motores, para disminuir el exceso de energía disipada (calor) de los mismos, debidas a rozamientos entre sus piezas. Así como el reducir aún más en superficie, el circuito impreso.

Otra parte modificable, es el programa en C++, el cual requiere ser ampliado para el control de mas de dos motores bimodales independientes. Con ello acoplar los motores necesarios al microequipo y realizar las cuantificaciones necesarias para una programación de posiciones y tiempo de trabajo; para mas tarde llegar a prescindir de la tarjeta de programación UP1 y con tan solo el FPGA montado en un socket, sea posible leerlo y reprogramarlo para realizar micropiezas en serie y con ello eliminar la PC por completo. Así mismo llegar a una reducción de espacio-costo, para una mayor competitividad y eficiencia a nivel mundial con esta nueva microcélula de producción.

REFERENCIAS

Libros y Artículos empleados para la realización de este trabajo de tesis

ELECTRÓNICA

[1] Manual de *Motorola*, ***Fast and LS TTL Data***; Logic Integrated Circuits División; E.U.A. 1992.

[2] Ronald J. Tocci; **Sistemas Digitales**, *Principios y Aplicaciones*, Monroe Community College, 1985, pp. 235, 237, 258, 264.

[3] *Data Manual*; **Signetics Analog, Specifications, Applications, Military Summary**. Signetics Corporation, E.U.A. 1997.

[4] *ECG*, **Semiconductors Master Replacement Guide**. Philips, E.U.A. 1994.

[5] *Data Manual*; **Signetics Logic – TTL, Specifications, Military Summary**. Signetics Corporation, E.U.A. 1997

[6] Harry L. Helms; **Circuitos electrónicos**, *guía práctica*. McGraw Hill, México 1986. pp. 7, 43, 65, 115, 247.

[7] Robert Boylestad, Louis Nashelsky, **Electrónica** *Teoría de circuitos*. Prentice may, p.176.

[8] http://mailweb.udlap.mx/~lgojeda/apuntes/electronica1/4_1.htm

[9] http://www.creaturoides.com/index.html

MECÁNICA

[10] N.Ooyama, S.Kokaji, M. Tanaka; *Desktop Machining Microfactory;* **Microfactory, 2nd International Workshop on Microfactories (Papers).** Fribourg, Switzerland. October 9-10, 2000, pp.13-16.

[11] E. Westkämper, Fraunhofer – *Institute for Manufacturing Engineering and Automation*, **Microfactory, 2nd International Workshop on Microfactories (Papers),**IPA, Germany, October 9-10, 2000, pp.17-18

[12] L.Ruiz Huerta; *Diseño y Construcción de un microcentro de bajo costo*, **Tesis a nivel Maestría**; México, 2000.

[13] Takashi Kenjo, Akira Sugarawa; **Stepping Motors and their microprocessor controls**, Oxford Science Publications, 1994.

[14] E.M. Kussul, D.A. Rachkovskij, TN. Baidyk et al**. Micromechanical engineering: a basis for the low cost manufacturing of mechanical microdevices using microequipment**. *J.Micromech. Microeng*. – 1996.- 6.- P.410-425

[15] *Journal of Micromechanics and Microengineering*, **Structures, devices and systems;** Volume 12 Number 6 November 2002

[16] Kussul E., Ruiz L., Caballero A., Kasatkina I., Baydyk T., **CNC machine tools for low cost micro devices manufacturing; First International Conference on Mechatronics and Robotics; Saint Petesburg, Russia; May 29 – June 2, 2000; Proceedings Volume 1; pp98-103.**

[17] http://www.cs.uiowa.edu/~jones/step/types.html

[18] http://www.sapiens.itgo.com/motores_por_pasos/motores_por_pasos.htm

COMPUTACIÓN

[19] Kent Reisdorph, **Aprendiendo Borland C++ Builder 3 en 21 Días**, Person Educación 1999.

[20] Norma Elva Chávez Rodríguez, Jorge Valeriano Assem, Arturo Haro Ruiz; *Manual de consulta*, **Entorno de diseño Max+Plus II**. Facultad de Ingeniería, División de Ingeniería Eléctrica, Departamento de ingeniería en computación e ingeniería en electrónica, UNAM.; Febrero 2001.

[21] *Developer's Guide*; **C++ Builder 6 Borland**, Borland Software Corporation, E.U.A. 2002.

[22] Herbert Schildt; **Programación en Turbo C**, Borland – Osborne. México 1992.

[23] *User Guide*; **Altera University Program, Design Laboratory Package**; Altera Corporation 2001. http://www.altera.com

ROBOTICA

[24] Hayashi I., Iwatsuki N. **Micro Moving Robotics**, *International Symposium on Micromechatronics and Human Science*, 1998, pp.41-50

[25] Karel Chapek**, La robotica en nuestros días.** Barcelona 1920**.**

ANEXO 1

Programa en Borland C++ Builder 4, para el control independiente (velocidad, número de pasos, sentido de giro, lazo abierto y lazo cerrado) de 2 motores bimodales.

```
//         Programa:           CONTROL DE MOTORES
//                          Laboratorio de Mecatrónica. U. N. A. M.
//         Programadores   Carlos Alberto Muñoz Leines
//                            Cesar Augusto Santos Carrasco
//              CCADET. Centro de Ciencias Aplicadas y Desarrollo Tecnológico
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Main.h"
#pragma package(smart_init)
#pragma link "TDLPortIO"
#pragma resource "*.dfm"
TMain_Win *Main_Win;
int Valor, r, t, s, u;
//---------------------------------------------------------------------------
__fastcall TMain_Win::TMain_Win(TComponent* Owner) : TForm(Owner)
{
  FBaseMode=bHex;   // Modo hexadecimal
  FDataType=dtByte; // Modo de bit

  PortEdit->Text="0x378"; // LPT1:
  DataEdit->Text="0x00";

  DLPortIO->DriverPath=ExtractFileDir(ParamStr(0));
  // El comando DriverLINX
  DLPortIO->OpenDriver();
  if (!DLPortIO->ActiveHW)
  {
    MessageDlg("No puede abrir el comando Driver!!!",
            mtError, TMsgDlgButtons() << mbOK, 0);
    PortGroup->Enabled=false;
    BaseGroup->Enabled=false;
    TypeGroup->Enabled=false;
  }
  }
//---------------------------------------------------------------------------
void __fastcall TMain_Win::Base10RadioClick(TObject *Sender)
{
  if (FBaseMode!=bDecimal)
  {
    // En un modo decimal
    FBaseMode=bDecimal;
    try {
      PortEdit->Text=IntToStr(StrToInt(PortEdit->Text));
    } catch (...) {
      PortEdit->Text="0";
```

```
    }
    try {
      DataEdit->Text=IntToStr(StrToInt(DataEdit->Text));
    } catch (...) {
      DataEdit->Text="0";
    }
  }
}
//---------------------------------------------------------------------------
void __fastcall TMain_Win::Base16RadioClick(TObject *Sender)
{
  if (FBaseMode!=bHex)
  {
    // En modo hexadecimal
    FBaseMode=bHex;
    try {
      PortEdit->Text="0x"+IntToHex(StrToInt(PortEdit->Text),0);
    } catch (...) {
      DataEdit->Text="0x0000";
    }
    try {
      DataEdit->Text="0x"+IntToHex(StrToInt(DataEdit->Text),0);
    } catch (...) {
      DataEdit->Text="0x00";
    }
  }
}
//---------------------------------------------------------------------------
void __fastcall TMain_Win::ReadButtonClick(TObject *Sender)
{
  WORD DataPort;  // Para leer el puerto
  DWORD DataRead; // Para leer la direeción deseada
  try {
    DataPort=(WORD)StrToInt(PortEdit->Text);
  } catch (...) {
    MessageDlg("No se ha especificado la direción.\n Salga y ejecute otra vez el programa.",
            mtError, TMsgDlgButtons() << mbOK, 0);
    return;
  }
  switch (FDataType)
  {
    case dtByte:  DataRead=DLPortIO->Port[DataPort];  break;
    case dtWord:  DataRead=DLPortIO->PortW[DataPort]; break;
    case dtDWord: DataRead=DLPortIO->PortL[DataPort]; break;
  }
  if (FBaseMode==bDecimal)
    DataEdit->Text=IntToStr(DataRead);
  else
    DataEdit->Text="0x"+IntToHex(int(DataRead), 0);
        }

//---------------------------------------------------------------------------
void __fastcall TMain_Win::WriteButtonClick(TObject *Sender)
{
  WORD DataPort;   // Escribe en edit el puerto
  DWORD DataWrite; // Toma en cuenta el valor
  try {
    DataPort=(WORD)StrToInt(PortEdit->Text);
  } catch (...) {
   MessageDlg("No se ha especificado la direción.\n Salga y ejecute otra vez el programa",
            mtError, TMsgDlgButtons() << mbOK, 0);
    return;
```

```
  }

  try {
    DataWrite=(DWORD)StrToInt(DataEdit->Text);
  } catch (...) {
   MessageDlg("No se ha especificado la direción.\n Salga y ejecute otra vez el programa.",
              mtError, TMsgDlgButtons() << mbOK, 0);
    return;
  }
  switch (FDataType)
  {
    case dtByte:  DLPortIO->Port[DataPort]=(BYTE)(DataWrite&0xFF);    break;
    case dtWord:  DLPortIO->PortW[DataPort]=(WORD)(DataWrite&0xFFFF); break;
    case dtDWord: DLPortIO->PortL[DataPort]=DataWrite;                break;
  }
}
//---------------------------------------------------------------------------
void __fastcall TMain_Win::ByteRadioClick(TObject *Sender)
{
  FDataType=dtByte;
}

//---------------------------------------------------------------------------

void __fastcall TMain_Win::WordRadioClick(TObject *Sender)
{
  FDataType=dtWord;
}

//---------------------------------------------------------------------------
void __fastcall TMain_Win::DWordRadioClick(TObject *Sender)
{
  FDataType=dtDWord;
}

//---------------------------------------------------------------------------
void __fastcall TMain_Win::AboutButtonClick(TObject *Sender)
{
  MessageDlg(
    "Programa:   CONTROL DE MOTORES \n\n"
     "Laboratorio de Mecatrónica.UNAM.\n\n"
     "Carlos Alberto Muñoz Leines.\n\n"
     "Cesar Augusto Santos Carrasco\n\n"
     "CCADET. Centro de Ciencias Aplicadas y Desarrollo Tecnológico",
    mtInformation,
    TMsgDlgButtons() << mbOK,
    0);
}

//---------------------------------------------------------------------------
void __fastcall TMain_Win::Button1Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
void __fastcall TMain_Win::Timer1Timer(TObject *Sender)
{
reloj->Caption=Time();
}
//---------------------------------------------------------------------------
void __fastcall TMain_Win::BitBtn2Click(TObject *Sender)
{
```

```
 AnsiString as;
   if (Memo1->Modified){

   SaveDialog1->Title = "Los datos se guardarán en:";
   if (SaveDialog1->Execute())
     {
    MessageDlg(" El archivo tipo texto será guardado en:   "+SaveDialog1->FileName+".txt",
               mtInformation    , TMsgDlgButtons() << mbOK, 0);
         as = ChangeFileExt(SaveDialog1->FileName,".txt");
         Memo1->Lines->SaveToFile(as);
         Memo1->Modified = false;
     }
     }
   else
  MessageDlg("No se puede guardar porque no se ha corrido el programa,  ni extraido alguna información !!! ",
               mtWarning , TMsgDlgButtons() << mbOK, 0);
}

//---------------------------------------------------------------------------
void __fastcall TMain_Win::FormCreate(TObject *Sender)
{
//para inicializar el programa que este todo apagado

       asm {
          mov DX, 0x378
          mov AX, 0xF0
          out DX, AX
          }
       asm {
          mov DX, 0x278
          mov AX, 0xF000
          out DX, AX
          }
}
//---------------------------------------------------------------------------
void __fastcall TMain_Win::BitBtn1Click(TObject *Sender)
{
r=1;
t=0;
int ahora, i, h, p, rpm;
Memo1->Lines->Add(">>>>>>> I N I C I O  M 1 <<<<<<<");
Memo1->Lines->Add("Fecha: "+Date()+"   Hora: "+Time());
//COMANDOS PARA EL MEMO EN MODO DE USO
  if(abierto->Checked)
   if (horario->Checked)    //lazo abierto-horario
      {
Timer2->Enabled=False;
Timer2->Enabled=True;
       Memo1->Lines->Add("MOTOR 1 LAZO ABIERTO - HORARIO ");
      }
   else {                 //lazo abierto-antihorario
    {
Timer2->Enabled=False;
Timer2->Enabled=True;
            Memo1->Lines->Add("MOTOR 1 LAZO ABIERTO - ANTIHORARIO ");
    }
        }
else{
    if (horario->Checked)  //lazo cerrado-horario
      {
Timer2->Enabled=False;
Timer2->Enabled=True;
```

```
      Memo1->Lines->Add("MOTOR 1 LAZO CERRADO - HORARIO ");
      }
 else{             //lazo cerrado-antihorario
      {
       Timer2->Enabled=False;
       Timer2->Enabled=True;
       Memo1->Lines->Add("MOTOR 1 LAZO CERRADO - ANTIHORARIO ");
      }
    }
    }
//Comandos para los pulsos y orden de bits
if(abierto->Checked){
   if (horario->Checked){
      asm        //lazo abierto-horario
      {
       mov DX, 0x378
       mov AX, 0x03
       out DX, AX
      }
                    }
   else {
      asm           //lazo abierto-antihorario
    {
      mov DX, 0x378
      mov AX, 0x02
      out DX, AX
    }
       }
                 }
else{
    {
   if (horario->Checked)
     asm             //lazo cerrado-horario
      {
       mov DX, 0x378
       mov AX, 0x00
       out DX, AX
      }
   else{             //lazo cerrado-antihorario
       asm
    {
      mov DX, 0x378
      mov AX, 0x01
      out DX, AX
    }
    }
    //sistema de control de velocidad en pulsos de un bit
  switch (TrackBar1->Position)
   {
        case 0:    {
       Memo1->Lines->Add("Caso 0");

             asm {
                  mov DX, 0x378
                  mov AX, 0xF0
                  out DX, AX
                 }
                break;
                 }
         case 1:    {
        Memo1->Lines->Add("Caso 1");
             ahora=GetTickCount();
```

```
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x378
        mov AX, 0x00
        out DX, AX
        }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x378
        mov AX, 0x04
        out DX, AX
       }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x378
        mov AX, 0x00
        out DX, AX
        }
        break;
        }
case 2:    {
      Memo1->Lines->Add("Caso 2");
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x378
        mov AX, 0x00
        out DX, AX
        }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x378
        mov AX, 0x04
        out DX, AX
       }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x378
        mov AX, 0x00
        out DX, AX
        }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x378
        mov AX, 0x04
        out DX, AX
       }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x378
        mov AX, 0x00
        out DX, AX
        }
break;
   }
case 3:    {
```

```
        Memo1->Lines->Add("Caso 3");
     ahora=GetTickCount();
     do;while (GetTickCount()-ahora<80);
     asm {
          mov DX, 0x378
          mov AX, 0x00
          out DX, AX
          }
     ahora=GetTickCount();
     do;while (GetTickCount()-ahora<80);
     asm {
          mov DX, 0x378
          mov AX, 0x04
          out DX, AX
         }
     ahora=GetTickCount();
     do;while (GetTickCount()-ahora<80);
     asm {
          mov DX, 0x378
          mov AX, 0x00
          out DX, AX
          }
     ahora=GetTickCount();
     do;while (GetTickCount()-ahora<80);
     asm {
          mov DX, 0x378
          mov AX, 0x04
          out DX, AX
         }
     ahora=GetTickCount();
     do;while (GetTickCount()-ahora<80);
     asm {
          mov DX, 0x378
          mov AX, 0x00
          out DX, AX
          }
     ahora=GetTickCount();
     do;while (GetTickCount()-ahora<80);
     asm {
          mov DX, 0x378
          mov AX, 0x04
          out DX, AX
         }
     ahora=GetTickCount();
     do;while (GetTickCount()-ahora<80);
     asm {
          mov DX, 0x378
          mov AX, 0x00
          out DX, AX
          }
   break;
     }
   case 4:   {
   Memo1->Lines->Add("Caso 4");
     asm {
          mov DX, 0x378
          mov AX, 0x00
          out DX, AX
        }
     ahora=GetTickCount();
     do;while (GetTickCount()-ahora<80);
     asm {
```

```
                mov DX, 0x378
                mov AX, 0x04
                out DX, AX
               }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                mov DX, 0x378
                mov AX, 0x00
                out DX, AX
                }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                mov DX, 0x378
                mov AX, 0x04
                out DX, AX
               }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                mov DX, 0x378
                mov AX, 0x00
                out DX, AX
                }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                mov DX, 0x378
                mov AX, 0x04
                out DX, AX
               }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                mov DX, 0x378
                mov AX, 0x00
                out DX, AX
                }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                mov DX, 0x378
                mov AX, 0x04
                out DX, AX
               }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                mov DX, 0x378
                mov AX, 0x00
                out DX, AX
               }
           break;
            }
      }
    }
 }
}
//---------------------------------------------------------------------------
void __fastcall TMain_Win::BitBtn4Click(TObject *Sender)
{
```

```
Timer3->Enabled=False;
Memo1->Lines->Add("Se contabilizaron "+Edit2->Text+" pasos en M2: "+Time());
//sistema de control de velocidad en pulsos de un bit
int ahora;
  switch (TrackBar2->Position)
    {
          case 4:    {
             ahora=GetTickCount();
             do;while (GetTickCount()-ahora<80);
             asm {
                  mov DX, 0x278
                  mov AX, 0x0000
                  out DX, AX
                  }
             ahora=GetTickCount();
             do;while (GetTickCount()-ahora<80);
             asm {
                  mov DX, 0x278
                  mov AX, 0x0400
                  out DX, AX
                 }
             ahora=GetTickCount();
             do;while (GetTickCount()-ahora<80);
             asm {
                  mov DX, 0x278
                  mov AX, 0x0000
                  out DX, AX
                  }
                  break;
                  }
          case 3:    {
             ahora=GetTickCount();
             do;while (GetTickCount()-ahora<80);
             asm {
                  mov DX, 0x278
                  mov AX, 0x0000
                  out DX, AX
                  }
             ahora=GetTickCount();
             do;while (GetTickCount()-ahora<80);
             asm {
                  mov DX, 0x278
                  mov AX, 0x0400
                  out DX, AX
                 }
             ahora=GetTickCount();
             do;while (GetTickCount()-ahora<80);
             asm {
                  mov DX, 0x278
                  mov AX, 0x0000
                  out DX, AX
                  }
             ahora=GetTickCount();
             do;while (GetTickCount()-ahora<80);
             asm {
              $    mov DX, 0x278
                  mov AX, 0x0400
                  out DX, AX
                 }
             ahora=GetTickCount();
             do;while (GetTickCount()-ahora<80);
             asm {
```

```
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
break;
   }
case 2:    {
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0400
        out DX, AX
       }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0400
        out DX, AX
       }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0400
        out DX, AX
       }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
break;
   }
case 1:    {

   asm {
        mov DX, 0x278
```

```
          mov AX, 0x0000
          out DX, AX
         }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
          mov DX, 0x278
          mov AX, 0x0400
          out DX, AX
         }

    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
          mov DX, 0x278
          mov AX, 0x0000
          out DX, AX
          }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
          mov DX, 0x278
          mov AX, 0x0400
          out DX, AX
         }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
          mov DX, 0x278
          mov AX, 0x0000
          out DX, AX
          }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
          mov DX, 0x278
          mov AX, 0x0400
          out DX, AX
         }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
          mov DX, 0x278
          mov AX, 0x0000
          out DX, AX
          }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
          mov DX, 0x278
          mov AX, 0x0400
          out DX, AX
         }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
          mov DX, 0x278
          mov AX, 0x0000
          out DX, AX
         }
 break;
    }
```

```
      }
   //Para apagar
        asm {
          mov DX, 0x278
          mov AX, 0xF000
          out DX, AX
          }
    }
//---------------------------------------------------------------------------
void __fastcall TMain_Win::Timer3Timer(TObject *Sender)
{
  for(int g=0;g<100;g++)
{
  FBaseMode=bDecimal;
  FDataType=dtDWord;
              PortEdit->Text="0x278";
              DataEdit->Text="0x00";
  DLPortIO->DriverPath=ExtractFileDir(ParamStr(0));
  DLPortIO->OpenDriver();
  if (!DLPortIO->ActiveHW)
  {
    MessageDlg("No hay configuracion en el puerto.",
            mtError, TMsgDlgButtons() << mbOK, 0);
    PortGroup->Enabled=false;
    BaseGroup->Enabled=false;
    TypeGroup->Enabled=false;
  }
   WORD DataPort;
  DWORD DataRead;
  try {
    DataPort=(WORD)StrToInt(PortEdit->Text);
  } catch (...) {
    MessageDlg("No se especifico el puerto.",
            mtError, TMsgDlgButtons() << mbOK, 0);
    return;
  }
  switch (FDataType)
  {
    case dtByte:  DataRead=DLPortIO->Port[DataPort];  break;
    case dtWord:  DataRead=DLPortIO->PortW[DataPort]; break;
    case dtDWord: DataRead=DLPortIO->PortL[DataPort]; break;
  }
  if (FBaseMode==bDecimal)
    DataEdit->Text=IntToStr(DataRead);
  else
    DataEdit->Text="0x"+IntToHex(int(DataRead), 0);
              WORD DataPort2;
              DWORD DataRead1;
                try {
                  DataPort2=(WORD)StrToInt(PortEdit->Text);
              } catch (...) {
              MessageDlg("No se ha especificado la dirección del puerto.\nNo se puede ejecutar  el programa.",
                        mtError, TMsgDlgButtons() << mbOK, 0);
                return;
              }
                DataRead1=DLPortIO->PortW[DataPort2];
               Edit2->Text=IntToStr(DataRead1);
              WORD DataEdit2;
                try {
                  DataEdit2=(WORD)StrToInt(Edit2->Text);
              } catch (...) {
                MessageDlg("No es valida la información",
```

```
                    mtError, TMsgDlgButtons() << mbOK, 0);
                        return;
                }

Valor=(DWORD)StrToInt(DataEdit->Text);
if (Valor>809960){
    s=s++;
              }
    else{
    s=0;
       }
if (s==1){
   u=u++;
        }
  else {
   u=u;
      }
if(s>=1){
      u=u;
       }
     else{
      s=s;
     }
Edit2->Text=u;
}
}
//---------------------------------------------------------------------------
void __fastcall TMain_Win::BitBtn3Click(TObject *Sender)
{
s=1;
u=0;
int ahora, i, h, p, rpm;
Memo1->Lines->Add(">>>>>>> I N I C I O  M 2 <<<<<<<");
Memo1->Lines->Add("Fecha: "+Date()+"   Hora: "+Time());
//COMANDOS PARA EL MEMO EN MODO DE USO

  if(abierto->Checked)
   if (horario->Checked)    //lazo abierto-horario
       {
Timer3->Enabled=False;
Timer3->Enabled=True;
        Memo1->Lines->Add("MOTOR 2 LAZO ABIERTO - HORARIO ");
       }
   else {                  //lazo abierto-antihorario
    {
Timer3->Enabled=False;
Timer3->Enabled=True;
             Memo1->Lines->Add("MOTOR 2 LAZO ABIERTO - ANTIHORARIO ");
     }
         }
else{
     if (horario->Checked)  //lazo cerrado-horario
       {
Timer3->Enabled=False;
Timer3->Enabled=True;
        Memo1->Lines->Add("MOTOR 2 LAZO CERRADO - HORARIO ");
        }
 else{              //lazo cerrado-antihorario
       {
         Timer3->Enabled=False;
         Timer3->Enabled=True;
         Memo1->Lines->Add("MOTOR 2 LAZO CERRADO - ANTIHORARIO ");
```

```
      }
    }
    }
//Comandos para los pulsos y orden de bits
if(abierto->Checked){
  if (horario->Checked){
     asm        //lazo abierto-horario
     {
      mov DX, 0x278
      mov AX, 0x0300
      out DX, AX
     }
                  }
  else {
     asm           //lazo abierto-antihorario
   {
     mov DX, 0x278
     mov AX, 0x0200
     out DX, AX
   }
       }
              }
else{
    {
  if (horario->Checked)
    asm             //lazo cerrado-horario
     {
      mov DX, 0x278
      mov AX, 0x0000
      out DX, AX
     }
  else{             //lazo cerrado-antihorario
      asm
   {
     mov DX, 0x278
     mov AX, 0x0100
     out DX, AX
   }
   }
   //sistema de control de velocidad en pulsos de un bit

 switch (TrackBar2->Position)
  {
       case 0:    {
      Memo1->Lines->Add("Caso 0");
            asm {
                 mov DX, 0x278
                 mov AX, 0xF0
                 out DX, AX
                }
               break;
                }
        case 1:    {
       Memo1->Lines->Add("Caso 1");
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                 mov DX, 0x278
                 mov AX, 0x0000
                 out DX, AX
                 }
            ahora=GetTickCount();
```

```
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0400
        out DX, AX
       }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
        break;
        }
case 2:    {
      Memo1->Lines->Add("Caso 2");
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0400
        out DX, AX
       }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0400
        out DX, AX
       }
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
break;
   }
case 3:    {
      Memo1->Lines->Add("Caso 3");
   ahora=GetTickCount();
   do;while (GetTickCount()-ahora<80);
   asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
```

```
        }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
        mov DX, 0x278
        mov AX, 0x0400
        out DX, AX
       }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
        mov DX, 0x278
        mov AX, 0x0400
        out DX, AX
       }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
        mov DX, 0x278
        mov AX, 0x0400
        out DX, AX
       }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
        }
break;
    }
case 4:    {
Memo1->Lines->Add("Caso 4");
    asm {
        mov DX, 0x278
        mov AX, 0x0000
        out DX, AX
      }
    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
    asm {
        mov DX, 0x278
        mov AX, 0x0400
        out DX, AX
       }

    ahora=GetTickCount();
    do;while (GetTickCount()-ahora<80);
```

```
            asm {
                 mov DX, 0x278
                 mov AX, 0x0000
                 out DX, AX
                 }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                 mov DX, 0x278
                 mov AX, 0x0400
                 out DX, AX
                }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                 mov DX, 0x278
                 mov AX, 0x0000
                 out DX, AX
                 }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                 mov DX, 0x278
                 mov AX, 0x0400
                 out DX, AX
                }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                 mov DX, 0x278
                 mov AX, 0x0000
                 out DX, AX
                 }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                 mov DX, 0x278
                 mov AX, 0x0400
                 out DX, AX
                }
            ahora=GetTickCount();
            do;while (GetTickCount()-ahora<80);
            asm {
                 mov DX, 0x278
                 mov AX, 0x0000
                 out DX, AX
                }
           break;
            }
       }
     }
 }
}
//---------------------------------------------------------------------------
void __fastcall TMain_Win::BitBtn5Click(TObject *Sender)
{
Timer2->Enabled=False;
   Memo1->Lines->Add("Se contabilizaron "+Edit1->Text+" pasos en M1: "+Time());
//Para apagar
       asm {
         mov DX, 0x378
         mov AX, 0xF0
```

```
          out DX, AX
          }
}
//---------------------------------------------------------------------------
void __fastcall TMain_Win::Timer2Timer(TObject *Sender)
{
for(int f=0;f<100;f++)
{
 PortEdit->Text="0x378";
                 WORD DataPort1;
                 DWORD DataRead1;
                   try {
                    DataPort1=(WORD)StrToInt(PortEdit->Text);
                 } catch (...) {
                MessageDlg("No se ha especificado la dirección del puerto.\nNo se puede ejecutar  el programa.",
                          mtError, TMsgDlgButtons() << mbOK, 0);
                    return;
                 }
                    DataRead1=DLPortIO->PortW[DataPort1];
                  Edit1->Text=IntToStr(DataRead1);
                 WORD DataEdit1;
                   try {
                    DataEdit1=(WORD)StrToInt(Edit1->Text);
                 } catch (...) {
                    MessageDlg("No es valida la información",
                       mtError, TMsgDlgButtons() << mbOK, 0);
                           return;
                 }
if (DataEdit1>24620){
    r=r++;
               }
    else{
    r=0;
        }
if (r==1){
   t=t++;
         }
  else {
    t=t;
       }
if(r>=1){
      t=t;
        }
     else{
      r=r;
      }
Edit1->Text=t;
}
}
//---------------------------------------------------------------------------
```

ANEXO 2

TABLA #1

Nomenclatura del conector JTAG_IN

Display de 7 segmentos	Pin para digito 1	Pin para Digito 2
A	58	69
B	60	70
C	61	73
D	63	74
E	64	76
F	65	75
G	67	77
Punto Decimal	68	79

TABLA #2

Posiciones de los tres jumpers

TDI	TDO	DEVICE	BOARD
C1	C1	C1	C1
C2	C2	C2	C2
C3	C3	C3	C3

TABLA #3

Acción según la posición de los jumpers

Desired Action	TDI	TDO	DEVICE	BOARD
Programación únicamente del EPM7128S	C1&C2	C1&C2	C1&C2	C1&C2
Configuración únicamente del EPF10K20	C2&C3	C2&C3	C1&C2	C1&C2
Programación / configuración ambos dispositivos	C2&C3	C1&C2	C2&C3	C1&C2
Conexión de múltiples tarjetas	C2&C3	OPEN	C2&C3	C2&C3

TABLA #4

Conectores alrededor del dispositivo EPM7128S

P1		P2		P3		P4	
Outside	**Inside**	**Outside**	**Inside**	**Outside**	**Inside**	**Outside**	**Inside**
75	76	12	13	33	34	54	55
77	78	14	15	35	36	56	57
79	80	16	17	37	38	58	59
81	82	18	19	39	40	60	61
83	84	20	21	41	42	62	63
1	2	22	23	43	44	64	65
3	4	24	25	45	46	66	67
5	6	26	27	47	48	68	69
7	8	28	29	49	50	70	71
9	10	30	31	51	52	72	73
10	X	32	X	53	X	74	X

TABLA #5

Conexiones de los pines del dispositivo hacia los displays

Display de 7 segmentos	Pines para el digito 1	Pines para el digito 2
a	58	69
b	60	70
c	61	73
d	63	74
e	64	76
f	65	75
g	67	77
Punto decimal	68	79

TABLA #6

Nombres de las señales contra los pines conectados en cada orificio

Número de orificio	Señal/pin	Número de orificio	Señal/pin
1	RAW	2	GND
3	Vcc	4	GND
5	Vcc	6	GND
7	Sin conexión	8	Sin conexión
9	Sin conexión	10	Sin conexión
11	Sin conexión	12	GCLRn/1
13	OE1/84	14	OE2/GCLK2
15	4	16	5

17	6	18	8
19	9	20	10
21	11	22	12
23	15	24	16
25	17	26	18
27	20	28	21
29	22	30	24
31	25	32	27
33	28	34	29
35	30	36	31
37	33	38	34
39	35	40	36
41	37	42	39
43	40	44	41
45	44	46	45
47	46	48	48
49	49	50	50
51	51	52	52
53	54	54	55
55	56	56	57
57	Vcc	58	GND
59	Vcc	60	GND

TABLA #7

Asignación de pines para los switches FLEZ_SW1

Switch	EPF10J20 Pines
FLEX_SWITCH-1	41
FLEX_SWITCH-2	40
FLEX_SWITCH-3	39
FLEX_SWITCH-4	38
FLEX_SWITCH-5	36
FLEX_SWITCH-6	35
FLEX_SWITCH-7	34
FLEX_SWITCH-8	33

TABLA #8

Asignación de pines para los displays de siete segmentos

Display de 7 segmentos	Pines para el digito 1	Pines para el digito 2
a	6	17
b	7	18
c	8	19
d	9	20
e	11	21
f	12	23
g	13	24
Punto decimal	14	25

TABLA #9

Nombres de las señales contra los pines conectados en cada orificio del *FLEX_EXPAN_A*

Número de orificio	Señal/pin	Número de orificio	Señal/pin
1	RAW	2	GND
3	Vcc	4	GND
5	Vcc	6	GND
7	Sin conexión	8	DI1/99
9	DI2/92	10	DI3/210
11	DI4/212	12	DEV_CLR/209
13	DEV_OE/213	14	DEV_CLK2/211
15	45	16	46
17	48	18	49
19	50	20	51
21	53	22	54
23	55	24	56
25	61	26	62
27	63	28	64
29	65	30	66
31	67	32	68
33	70	34	71
35	72	36	73
37	74	38	75
39	76	40	78
41	79	42	80
43	81	44	82
45	83	46	84
47	86	48	87
49	88	50	94
51	95	52	97
53	98	54	99
55	100	56	101
57	Vcc	58	GND
59	Vcc	60	GND

TABLA #10

Asignación de los pines del conector Centronics

Conector Centronics	Función en la PC
2	DataBit 1
3	DataBit 2
7	DataBit 6
8	DataBit 7
9	DataBit 8
10	NACK
11	Busy
12	Paper end
13	Select
14	nAuto_FEED
19-30,32	GND
1,4..6,15..18,31,33..36	Sin conectar

TABLA #11

Asignación de los pines del conector ISP

Pata	Señal JTAG
1	TCK
2	GND
3	TDO
4	Sin conectar
5	TMS
6	Sin conectar
7	Sin conectar
8	Sin conectar
9	TDI
10	GND

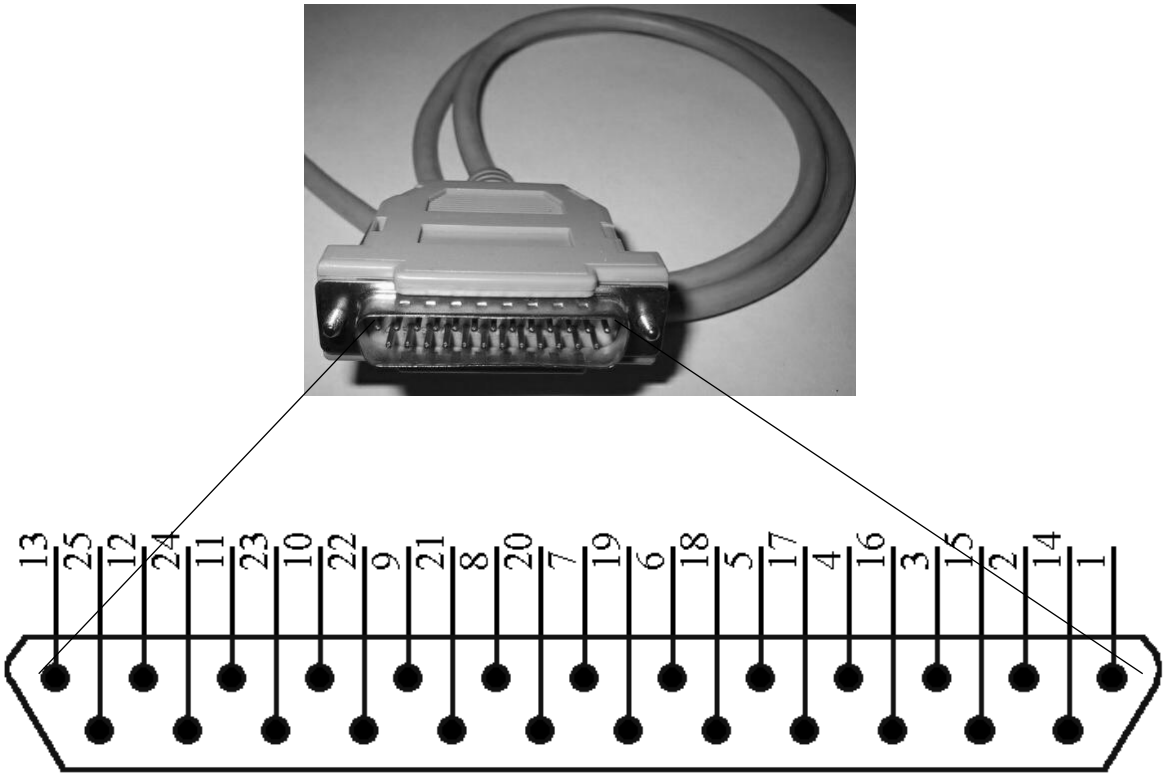
TABLA #12

Asignación de los pines del conector de alimentación

Pata	Función
1	GND
2	V+
3	GND

ANEXO 3

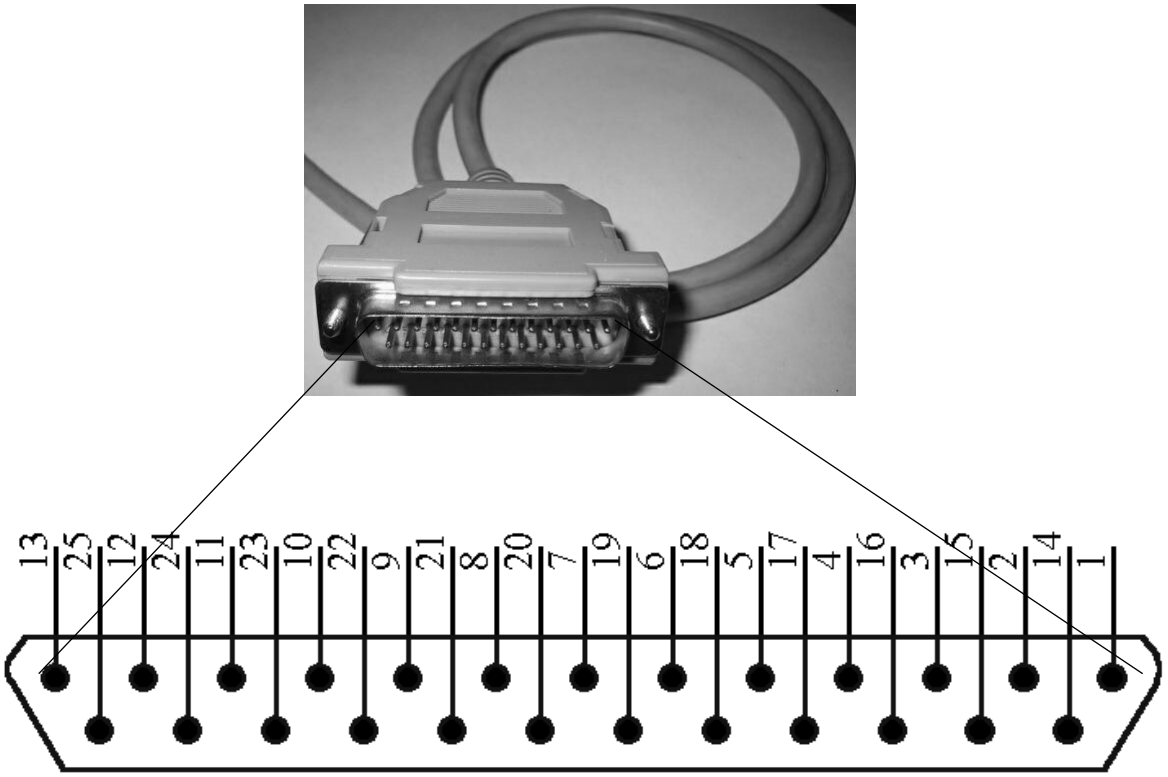
Configuración de pines del puerto paralelo (LPT).



Pin	E/S	Polaridad Activa	Descripción
1	Salida	0	Strobe
2~9	Salida	-	Líneas de datos (bit 0/pin 2, bit 7/pin9)
10	Entrada	0	Línea acknowledge (activa cuando el sistema remoto toma datos)
11	Entrada	0	Línea busy (si está activa, el sistema remoto no acepta datos)
12	Entrada	1	Línea falta de papel (si esta activa, falta papel en la impresora)
13	Entrada	1	Línea Select (si está activa, la impresora no se ha seleccionado)
14	Salida	0	Línea Autofeed (si está activa, la impresora inserta una nueva línea por cada retorno de carro)
15	Entrada	0	Línea Error (si está activa, hay un error en la impresora)
16	Salida	0	Línea INIT (Si se mantiene activa por al menor 50 microsegundos, está señal auto inicializa la impresora)
17	Salida	0	Línea Select Input (Cuando está inactiva, obliga a la impresora a salir de línea)
18~25	-	-	Tierra eléctrica

ANEXO 3

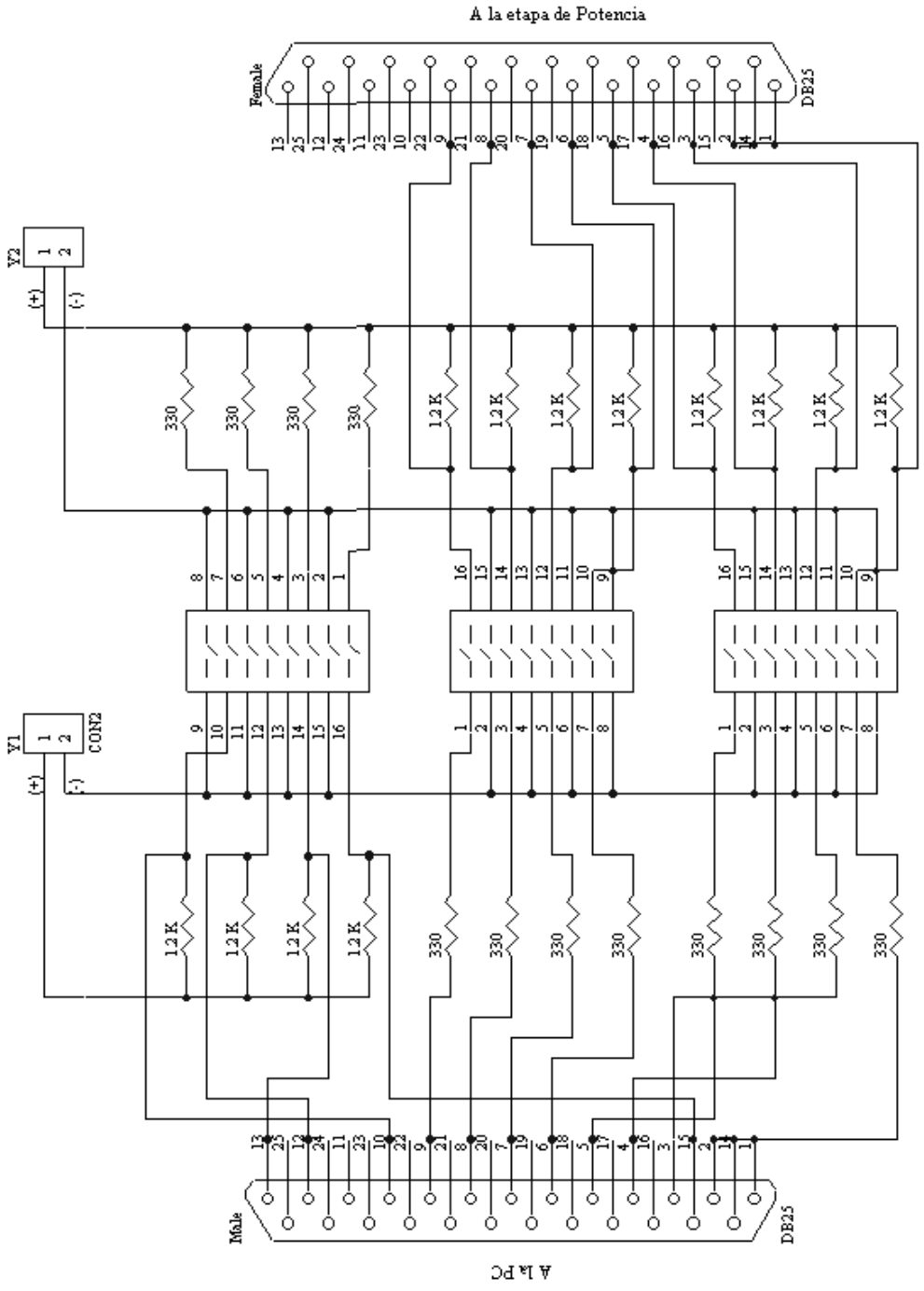
Configuración de pines del puerto paralelo (LPT).



Pin	E/S	Polaridad Activa	Descripción
1	Salida	0	Strobe
2~9	Salida	-	Líneas de datos (bit 0/pin 2, bit 7/pin9)
10	Entrada	0	Línea acknowledge (activa cuando el sistema remoto toma datos)
11	Entrada	0	Línea busy (si está activa, el sistema remoto no acepta datos)
12	Entrada	1	Línea falta de papel (si esta activa, falta papel en la impresora)
13	Entrada	1	Línea Select (si está activa, la impresora no se ha seleccionado)
14	Salida	0	Línea Autofeed (si está activa, la impresora inserta una nueva línea por cada retorno de carro)
15	Entrada	0	Línea Error (si está activa, hay un error en la impresora)
16	Salida	0	Línea INIT (Si se mantiene activa por al menor 50 microsegundos, está señal auto inicializa la impresora)
17	Salida	0	Línea Select Input (Cuando está inactiva, obliga a la impresora a salir de línea)
18~25	-	-	Tierra eléctrica

ANEXO 4

Protección contra transitorios de entrada hacia el puerto paralelo (LPT).



GLOSARIO

AND	Compuerta lógica multiplicadora
BCD	Contador Binario-Decimal (*Binary Counter Decimal*)
CC	Corriente Continua
CCADET	Centro de Ciencias Aplicadas y Desarrollo Tecnológico
CI	Circuito Integrado
CLB	Bloque de Lógica Configurable (*Configurable Logic Block*)
CMOS	Semiconductor con Oxido de Metal Complementario (*Complement Metal Oxid Semiconductor*)
CONACyT	Consejo Nacional de Ciencia y Tecnología
CPLD	Dispositivos de Lógica Programables Complejos (*Complex Programmable Logic Device*)
EDA	Automatización del Diseño Electrónico (*Electronic Design Automation*)
EEPROM	Memoria Programable Exclusiva para Lectura Borrable Eléctricamente (*Erase Electrical Programmable Read Only Memory*)
EPLD	Dispositivo lógico programable borrable (*Erasable Programmable Logic Device*)
EPROM	Memoria Programable Exclusiva para Lectura Borrable (*Erase Programmable Read Only Memory)*
Flip Flop	Dispositivo biestable síncrono
FPD	Dispositivo de campo programable (*Field Programmable Device*)
FPGA	Arreglos de Compuertas Programables de Campo (*Field Programmable Gate Array*)
GND	Tierra eléctrica (Ground)
HDL	Lenguaje descriptivo de hardware (*Hardware Description Language*)
IOB	Bloque de entrada-salida (*Input Output Block*)
JTAG	Joint Test Action Group
LCA	Arreglo de celdas lógicas (*Logic Cell Array*)
LED	Diodo Emisor de Luz (*Light Emisor Diode*)
LMM	Laboratorio de Micromecánica y Mecatrónica
LPT	Impresor en línea (*Line PrinTer*)
LUT	Look Up Table
MAX+PLUS II	Múltiple Array Matrix Programmable Logic User System II
MPGA	Arreglos de Matrices de Compuertas Programables (*Matrix Programmable Gate Array*)
NASA	National Aeronautics and Space Administration
OR	Compuerta lógica sumadora
OTP	Únicamente de tipo programable (*Only Type Programmable*)

PAL	Matriz Lógica Programable (*Programmable Array Logic*)
PC	Computadora personal (*Personal Computer*)
PCB	Circuito Impreso
PFU	Unidades de Función Programables (*Programmable Function Unit*)
PIC	Controlador de interrupciones programables (*Programmable Interrupt Controller*)
PLA	Arreglo Lógico Programable (Programmable Logic Array)
PLD	Dispositivos de Lógica Programable (*Programmable Logic Device*)
PROM	Memoria Programable únicamente de escritura (*Programmable Read-Only Memory*)
RAM	Memoria de Acceso Aleatorio (*Random Access Memory*)
RPM	Revoluciones Por Minuto
SPLD	Dispositivos de Lógica Programable Simple (*Simple Programmable Logic Device*)
SRAM	Memoria Acceso Aleatorio Estática (*Static RAM*)
TBJ	Transistor de Unión Bipolar (*Transistor Bipolar Juncture*)
UNAM	Universidad Nacional Autónoma de México
USB	Universal Serial Bus
Vcc	Voltaje de fuente
VHDL	Circuitos Integrados de Muy Alta Velocidad (*Very High Speed Integrated Circuit*)
WYSIWYG	Lo que ves es lo que tienes (*What You See Is What You*)