Instituto Nacional de Astrofísica, Óptica y Electrónica.

Diseño y construcción de un Robot de 4 ejes para aplicaciones en óptica

Por:

Gabriel Gordiano Alvarado,
Jorge Castro Ramos, y
Sergio Vázquez Montiel

Coordinación de óptica
Reporte técnico.

Tonantzintla, Puebla, octubre 2010

INDICE

Diseño y construcción de un Robot de 4 ejes para aplicaciones en óptica

Gabriel Gordiano Alvarado,
Jorge Castro Ramos,
Sergio Vázquez y Montiel

Resumen. En este reporte se describe detalladamente la estructura de un robot cartesiano el cual ha sido construido en su totalidad en el laboratorio de metrología del departamento de Optica. Diseño mecánico, hardware firmware y software son incluidos en la descripción. Se deja pendiente la parte de identificación paramétrica necesaria en la obtención de un modelo.

Palabras clave. Robot, Firmware, Encoder, Elementos Ópticos Difractivos.

1. Introducción

Sin duda, el desarrollo de tecnología es uno de los factores principales en el crecimiento de una nación en todos sus aspectos. En este trabajo se ha centrado la atención en el desarrollo de partes de la mayor importancia en la operación de un robot, como son la tarjeta de procesamiento, la sección de potencia y el firmware residente en la memoria del micro controlador.

A nivel de bloques, el único componente adquirido en el mercado, ha sido el conjunto de motores idénticos de propósito general. Cada motor utilizado contiene su encoder de mediana resolución.

El artefacto electromecánico ha sido construido con el propósito de llevar a cabo experimentos que conduzcan a la construcción de Elementos Opticos Difractivos[1]:, utilizando un rayo Láser como elemento abrasivo. No obstante, la máquina ha sido diseñada para ser utilizada de manera general, en un número ilimitado de procesos.

En nuestro desarrollo es conveniente interpretar concepto de Robot como un conjunto de piezas eléctricas y mecánicas que es multifuncional y reprogramable.

2 Descripción a nivel modular

2.1 Arquitectura del Sistema

Los elementos necesarios en la integración del mecanismo construido son representados en la figura 2.11.

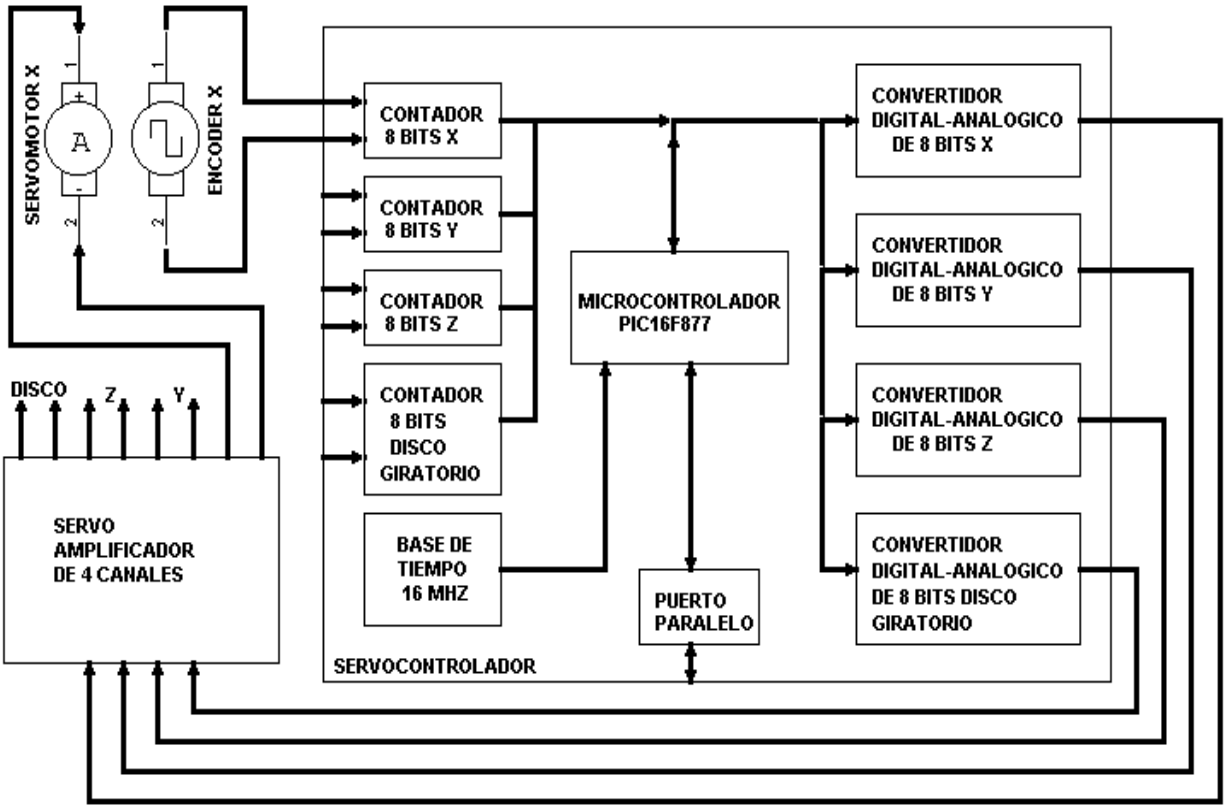

Figura 2.11. Diagrama del mecanismo construido

En el diagrama de bloques, solamente el grupo motor-encoder del eje X es representado en la esquina superior izquierda. Las dos líneas que llegan al contador de 8 bits representan las dos señales (que llamaremos A y B) desfasadas 90 grados, las cuales proveen la información necesaria para determinar la velocidad y el sentido de giro del motor.

Los bloques de conteo se construyen utilizando dispositivos GAL22v10. Previo a los contadores, las señales en cuadratura son registradas por dispositivos 74LS374 (registros octales).

Un micro controlador de la familia PIC16f877 es utilizado para efectuar todas las operaciones necesarias: calcula la posición de los rotores, el par necesario codificado a cada convertidor Digital-Analógico, y el mecanismo de comunicación con la computadora personal

por puerto paralelo. Es conveniente advertir que aun entre micro controladores de la misma familia (por ejemplo PIC16F877 y PIC16F877A) existen diferencias importantes, que el fabricante no menciona en las hojas de datos preliminares.

En cada bloque de conversión Digital-Analógica se provee a la salida un rango de +/- 5 volts, para excitar cada bloque amplificador de transconductancia (convertidor de voltaje a corriente).

2.2 Modelo de referencia

En esta sección, se efectúa el análisis de estabilidad de una ley de control Proporcional-Derivativa (PD), aplicada a manipuladores, y en particular provee un esquema posible en el control del mecanismo construido.

Consideremos la ecuación de balance de fuerza, la cual se deriva directamente de la ecuación de Euler-Lagrange, que modela la dinámica de un manipulador, y que en forma matricial puede escribirse como [2]:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \qquad (1)$$

Donde $H(q)$ es la matriz de inercia, la cual es simétrica y positiva definida,

$C(q,\dot{q})$ es el vector de fuerzas de Coriolis y centrípetas,

$g(q)$ son los vectores de pares gravitacionales.

A partir de la definición de Potencia (variación de energía con respecto al tiempo), podemos establecer la ecuación de balance del sistema en lazo cerrado de la siguiente manera[1]:

$$\frac{1}{2}\frac{d}{dt}\left[\dot{q}^{T} H \dot{q}\right] = \dot{q}^{T}\tau \qquad (2)$$

En el caso escalar, esta ecuación es:

[1] T denota transposición del vector.

$$\frac{1}{2}\frac{d}{dt}\left[mv^2\right] = Fv \tag{3}$$

La cual expresa que la variación de la energía (potencia) es equivalente a la fuerza aplicada multiplicada por la velocidad de la partícula.

Si se propone una ley de control de la forma

$$\tau = -k_p\tilde{q} - k_d\dot{q} \tag{4}$$

con $\tilde{q}$ como vector de error, y $\dot{q}$ representando el vector de velocidades, y si además, se propone una función de Liapunov (energía cinética + energía potencial) de la siguiente forma:

$$V = \frac{1}{2}\left[\dot{q}^T H \dot{q}\right] + \frac{1}{2}\left[\tilde{q}^T H \tilde{q}\right] \tag{5}$$

Derivando (4), y notando que el primer termino de la derecha es $\dot{q}^T\tau$, obtenemos:

$$\dot{V} = \dot{q}^T\tau + \frac{1}{2}\left[\tilde{q}^T H \dot{\tilde{q}} + \dot{\tilde{q}}^T H \tilde{q}\right] \tag{6}$$

Sustituyendo el valor de τ se obtiene:

$$\dot{V} = -\dot{q}^T k_d \dot{q}$$

Esta variación de energía siempre negativa muestra que el sistema es asintóticamente estable.

Es conveniente mencionar que en este artefacto, solamente el primer término de (3) ha sido programado. Esto quiere decir que los esquemas de control de velocidad se han construido con base en ensayos experimentales, debido a que no se ha efectuado una identificación paramétrica.

2.3 Ejemplo de acoplamiento electromecánico

Entre el codificador de posición y el procesador, debe existir un elemento que registre la posición de cualquier eje. Esta operación se consigue mediante un contador ascendente/descendente de 8 bits. Las señales A y B de entrada al contador se encuentran desfasadas 90 grados; este desfasamiento es necesario porque el adelanto de una con respecto a la otra determina el sentido de giro, y de la frecuencia de pulsos se estima la velocidad angular.

El proceso de conteo de pulsos se lleva a cabo de la siguiente manera: Tomamos como referencia cualquiera de las dos señales, por ejemplo la señal B. El reloj de conteo es de una frecuencia muy superior a la máxima estimada cuando gira el eje; como ejemplo, supongamos que nuestra velocidad máxima permisible en un eje es de 1000 RPM, y utilizamos un encoder de 9600 Pulsos por revolución, esto implica que la frecuencia máxima de la señal generada en el encoder es de

f = ((1000 rev/min)/(60 seg/min))(9600 Pulsos/Rev) = 160 Khz

Si arbitrariamente fijamos nuestro tiempo de muestreo en 1000 muestras por segundo (sps), la longitud máxima de registro de posición es:
(160000 pps)/1000 sps = 160 pulsos/muestreo.

Es suficiente entonces, un contador de 8 bits como elemento de registro de posición.

3 ARQUITECTURA

3.1 DESCRIPCION FUNCIONAL DEL BLOQUE DE PROCESAMIENTO

El bloque de control se ha construido en torno a un micro controlador del tipo PIC16F877. Entre los recursos de este dispositivo que han sido utilizados tenemos: 2 convertidores Analógico-Digital, 1 entrada de conteo de eventos externos y 28 pines de Entrada-Salida digitales. Un esquema detallado es mostrado en la figura 3.11.

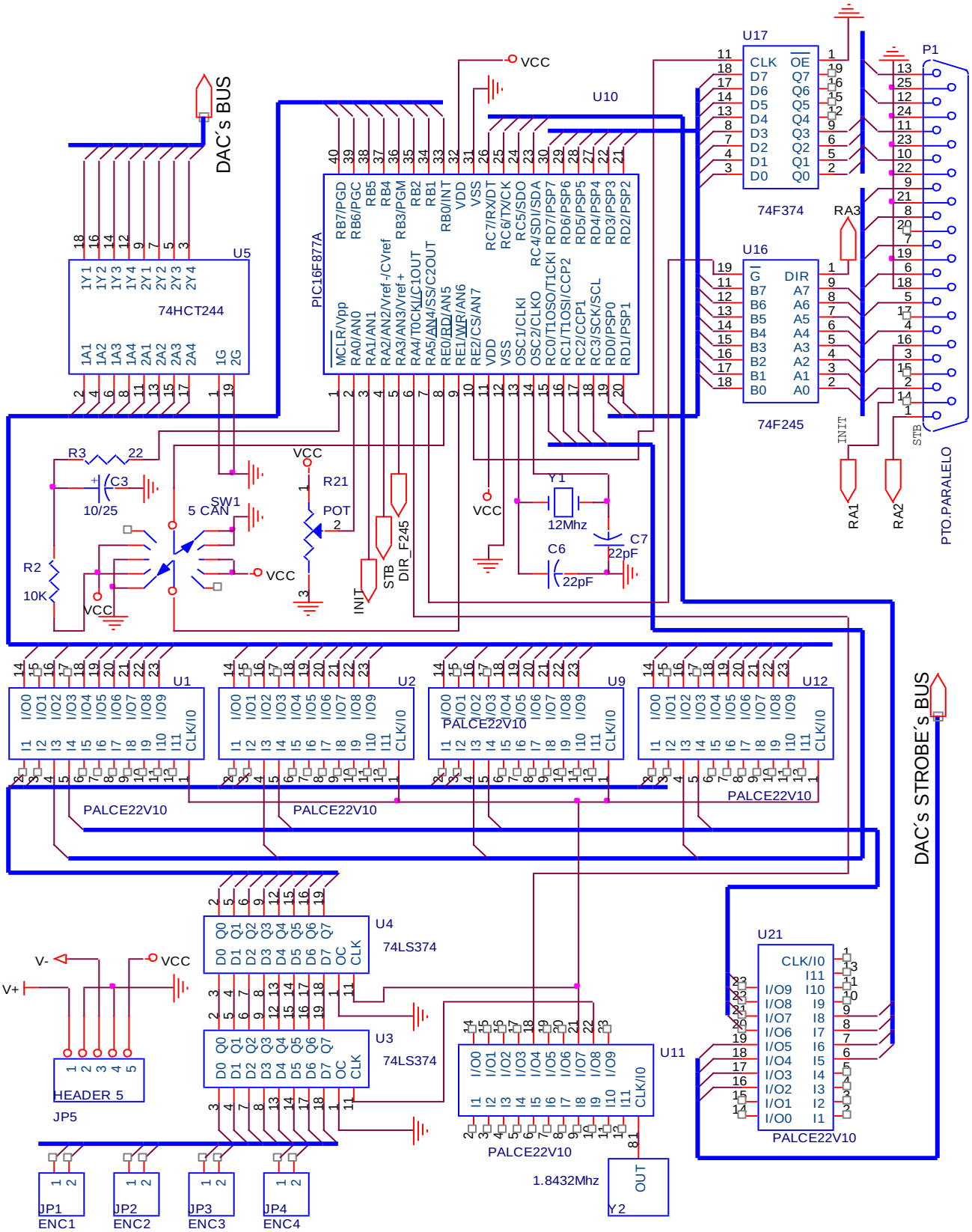

Figura 3.11. Control y medición de posición del sistema.

El sistema requiere una base de tiempo, la cual ha sido establecida inicialmente mediante un cristal de 16 Mhz. La señal que provee dicho reloj llega al dispositivo lógico programable U1, en el cual se ha programado un contador que divide la frecuencia entre 8; y esa señal de 2 Mhz. es utilizada para registrar en U3 las señales A y B de los 4 encoders incrementales. U4 es otro registro de 8 bits, en el cual se registran nuevamente las mismas señales de encoders, aunque con un reloj defasado. El propósito de este segundo registro es eliminar los posibles estados meta estables ocasionados por el registro asíncrono de las señales externas. El contador se construye como una maquina de 8 estados, en el que la salida en el pin 22 es invertida para obtener la señal de reloj del segundo registro, y esa misma señal es utilizada como reloj de los contadores medidores de posición U1, U1, U9 y U12, los cuales tambien han sido programados utilizando el GAL 22V10. En el apéndice A puede verse el codigo CNT3_2F2.PLD, escrito en el lenguaje del compilador universal de lógica programable (CUPL), del contador divisor por 8.

Los dispositivos medidores de posición son idénticos; básicamente, son contadores de 8 bits Up/Down con control de habilitación de salida y reset externos. En la programación de los contadores utilizados en la medición de posición, lo primero que se hace es sincronizar al reloj las dos señales A y B de cada encoder; las señales que se obtienen se han llamado QAS y QBS, y es conveniente mencionar que el bit menos significativo de los 8 bits de salida viene dado por la función or exclusivo de ambas señales, es decir:

D0 = QAS X QBS

El control de modo ya sea ascendente o descendente, se consigue retardando la señal QBS por un periodo de reloj; así, el control de modo es:

DIR = QBDL

El bit D1 en el canal de salida es simplemente QBS, y el problema que queda solamente consiste en hacer un contador de 6 bits. Como alternativa, y debido a que una maquina secuencial de 6 bits genera términos que exeden el número de puertas disponibles en el dispositivo programable, se ha optado por construir dos contadores de 3 bits en cascada con control de modo mediante los acarreos generados internamente. El código en lenguaje CUPL es mostrado en el apéndice B (CONTENC2.PLD).

Puede notarse que las salidas de los cuatro contadores (4 posibles ejes) son independientes. Esto se debe a que debe existir un mecanismo de lectura independiente para cada eje, en razón de que la velocidad del procesador es mucho mayor que cualquiera

de los dispositivos.

La habilitación de salida de cada uno de los contadores y registro de cada convertidor Digital-Analógico, es efectuada mediante el dispositivo U21. Este dispositivo GAL22V10 recibe las cuatro lineas mas significativas del puerto RC del microcontrolador, y activa en bajo cada línea de salida de acuerdo al codigo presente en los pines 6..9. El código en lenguaje CUPL es mostrado en el apéndice C con el nombre CNT3MOT2.PLD.

En la figura 3.11 puede observarse que el puerto RB del microcontrolador es compartido por los cuatro contadores y un buffer 74HCT244. Este buffer es necesario para reforzar las señales que son registradas en los 4 convertidores Digital-Analógico.

SW1 es un switch rotatorio que, en el modo manual, selecciona el canal que se mueve independientemente de los demás. En dicho modo manual. La velocidad del eje seleccionado es regulada mediante el potenciómetro R21.

Es conveniente mencionar que el transceiver U16 (74F245) ha sido utilizado para establecer comunicación bidireccional con la computadora personal, proceso en el que el dispositivo U17 establece 4 bits en el registro de estado del puerto paralelo.

Note la conexión entre la terminal 6 del microcontrolador y el divisor del reloj de 16 Mhz. Mediante la linea T0CKI se detecta la transición del reloj dividido, lo que origina una interrupción en la que se realiza la lectura de codificadores de posición de los 4 ejes.

Para el bloque de conversión Digital-Analógico, son utilizados los circuitos integrados DAC0800 los cuales proveen un rango de –5V hasta 5 Volts, para un rango de entrada desde 0H[2] hasta FFH, respectivamente (0 Volts corresponde a 7Fh). En la conversión de corriente a voltaje de la etapa de salida, amplificadores operacionales de alta frecuencia LF357 son utilizados. El diagrama esquemático detallado es mostrado en la figura 3.12.

[2] La letra H indica código hexadecimal.

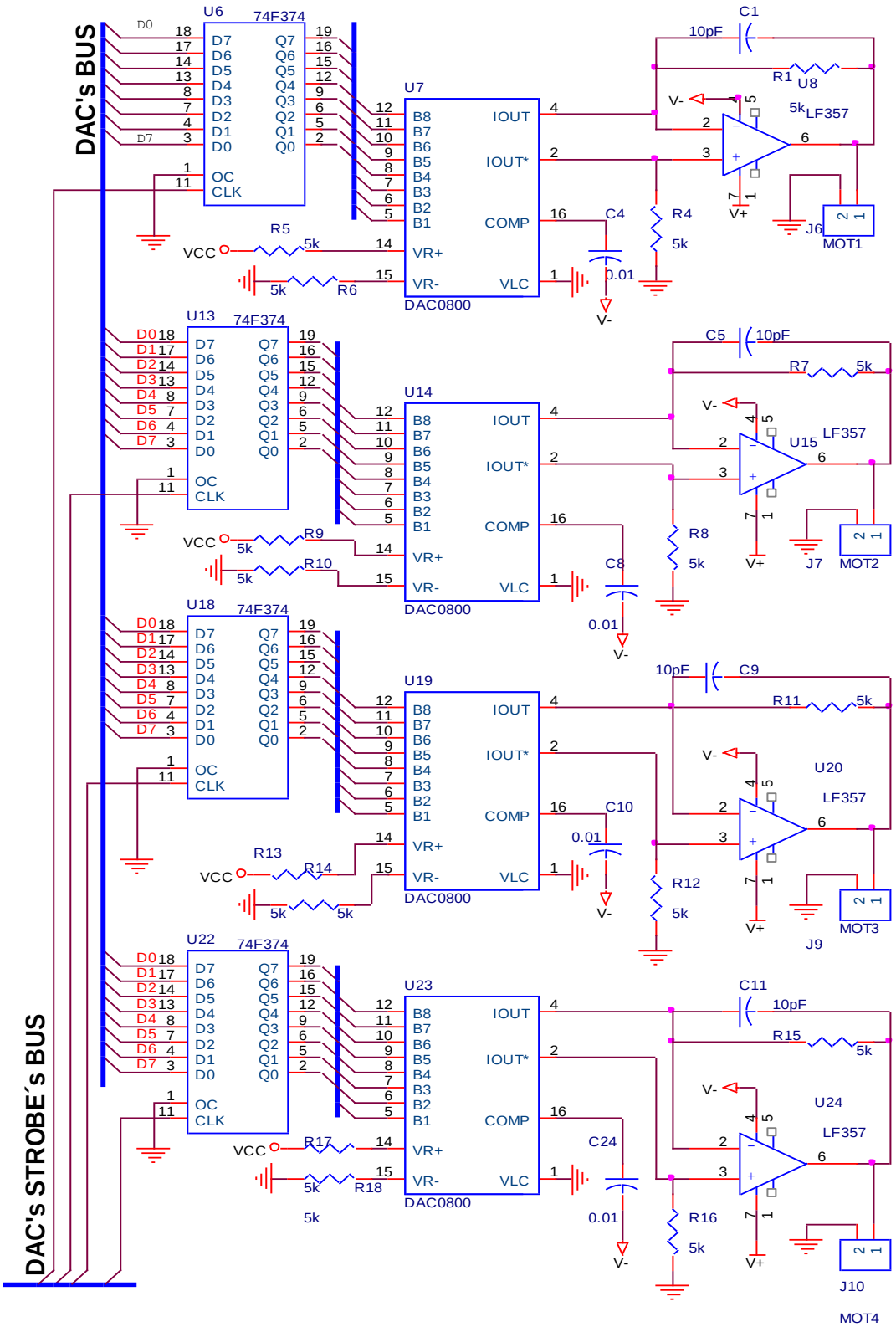

Figura 3.12. Etapa de conversión Digital-Analógica cuádruple.

3.2 AMPLIFICADOR LINEAL

La etapa de amplificación, necesaria para alimentar los motores, se ha construido alrededor de un amplificador operacional añadiendo una etapa de potencia simétrica. La trayectoria de retroalimentación negativa se deriva de un elemento sensor de corriente de carga, lo que permite controlar corriente a partir de una fuente de voltaje variable. La razón por la que se construye un amplificador de transconductancia tiene su origen en el hecho de que el par de un motor es proporcional a la corriente que circula por su devanado. En la figura 3.21 pueden observarse los detalles de este bloque [3,4].

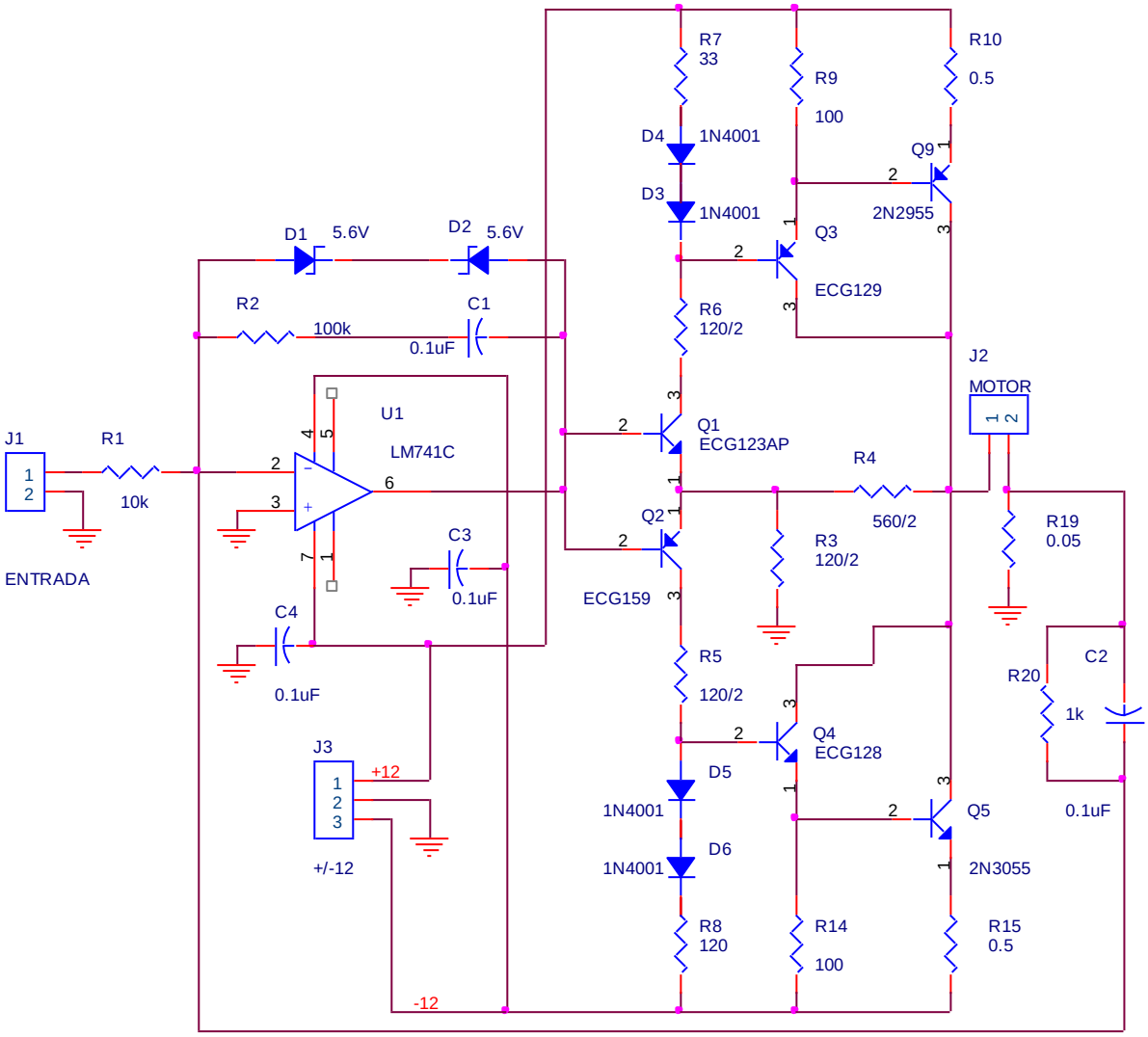

Figura 3.21. Amplificador de poder lineal de 5 Amperes.

4 IMPLEMENTACION

4.1 MECANISMO

El sistema electromecánico consta básicamente de 3 secciones: elementos de circuitos, piezas de software y mecanismo.

La parte mecánica está construida en forma de robot cartesiano, al cual se le ha agregado un eje en forma de plato giratorio para colocar piezas de algún material sobre el que se hace actuar la herramienta de trabajo. La carrera del eje X es de aproximadamente 32 cm, la del eje Y de 20 y el desplazamiento en el eje Z es de 8 cm. Se ha considerado como herramienta de trabajo un haz de Láser infrarrojo de 50 Watts, el cual puede trabajar tanto en modo continuo como pulsado.

Cada eje es accionado por un gusano con paso de 1mm; de modo que si consideramos que el encoder tiene 2000 estados en cuadratura por revolución, obtenemos una resolución de 0.5 micrones en el control de desplazamiento.

Para mover cada eje, se han utilizado 4 motores idénticos con las siguientes especificaciones técnicas:

Voltaje nominal: 24 VDC, 2700 ±270 RPM.
Torque; 11 in-lb. 12V DC, 1300 RPM.
Encoder incremental Optico: 2 fases, 500 pulsos/rev/fase.
Tamaño: 4" de diámetro.

Como alternativa para manejar los motores, se ha empleado el circuito LM12 de National Semiconductor, el cual es un amplificador operacional de alta potencia (80 Watts); aunque por el costo se ha preferido construir los bloques de manera discreta.

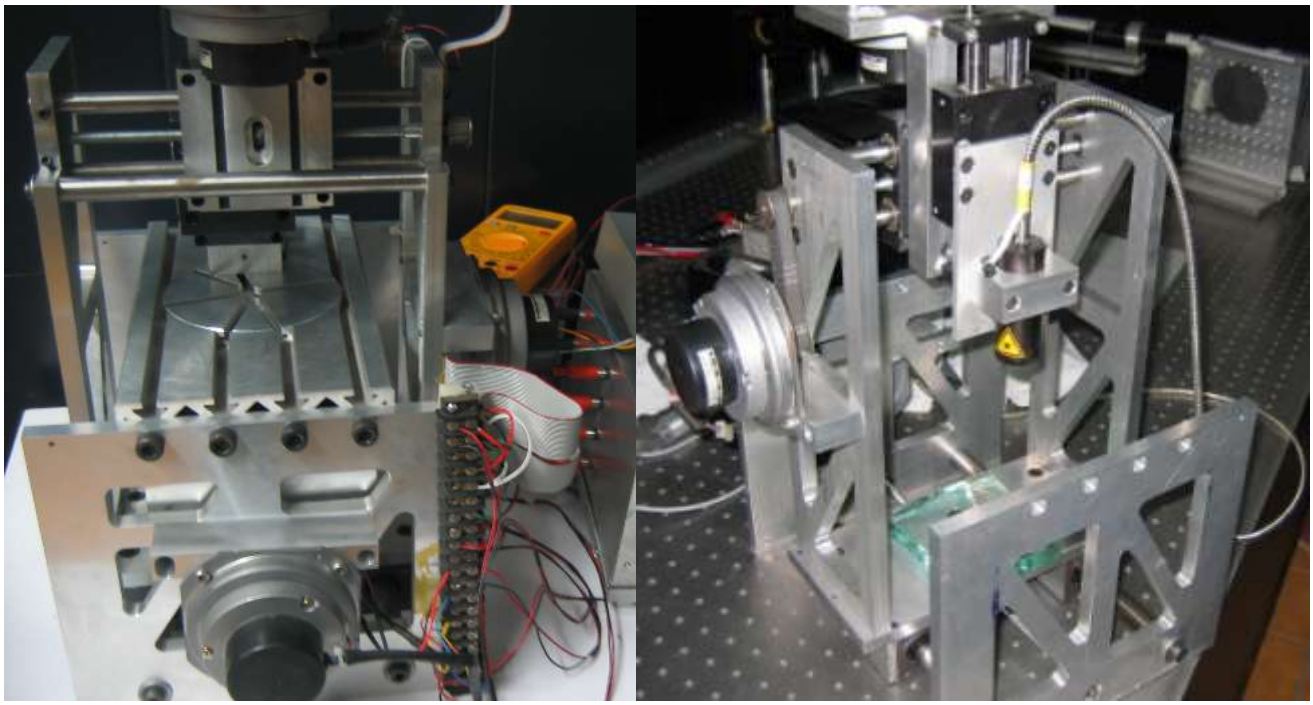

Figura 4.11 Dos vistas del sistema electromecánico de 4 ejes.

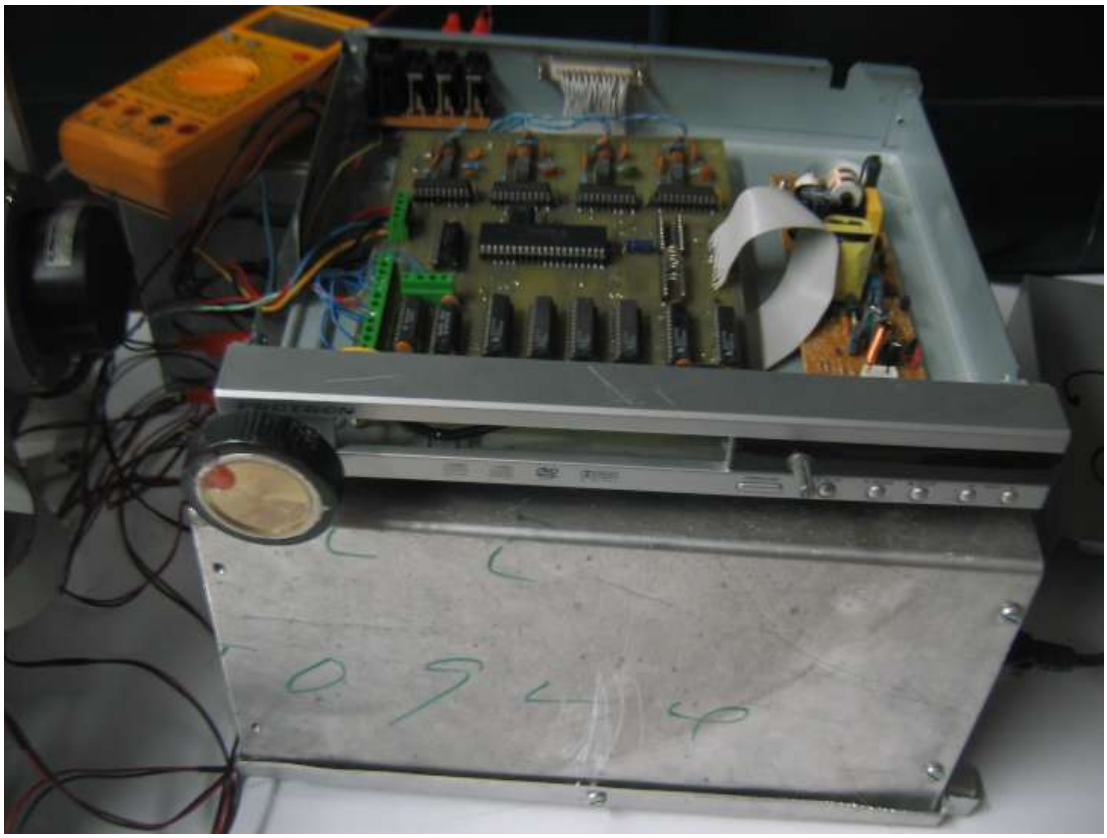

Figura 4.12 Tarjeta electrónica y amplificadores de alta potencia.

4.2 SOFTWARE, FIRMWARE

El sistema ha sido diseñado para operar de la manera más simple posible. Para el modo de control manual, este mecanismo tiene solamente un control potenciométrico, la posición central de éste marca el estado de reposo de algún motor en particular, o el comando CERO. Mediante un INTERRUPTOR de 4 posiciones se selecciona el eje que se desea mover. Una vez que se selecciona el eje que se desea girar, es posible mover el eje en el sentido deseado, a velocidad determinada por la posición del potenciómetro.

En modo automático, el sistema ha sido diseñado para trabajar en ambos modos; regulador o control de velocidad. Las referencias son enviadas a la computadora personal a través del puerto paralelo en una modalidad de handshake simple utilizando las líneas de control STROBE e INIT. Una vez que se han enviado las 4 referencias, mediante un botón se controla el estado del proceso.

Una versión del control manual es listada en el apéndice D con el nombre de TST3MOT_M.ASM.

Utilizando un ejemplo de la referencia [5], se han construido diversas piezas de software en VISUAL BASIC 6.0 para establecer la comunicación y diseñar una interfaz de manera conveniente. Un ejemplo de código utilizado es mostrado en el apéndice D. En particular, mediante el código mostrado se envían los SET POINTS a la tarjeta controladora, en los ejes X, Y y Z.

Conviene mencionar que en Windows XP el acceso al puerto paralelo no se puede realizar directamente con instrucciones de entrada-salida como en ediciones de compiladores y sistema operativo anteriores. Para conseguir el acceso es necesario utilizar la librería de enlace dinámico IO.DLL, obtenida previamente en la internet [6,7].

La apariencia del panel de control de motores es el que se muestra en la figura 4.11.

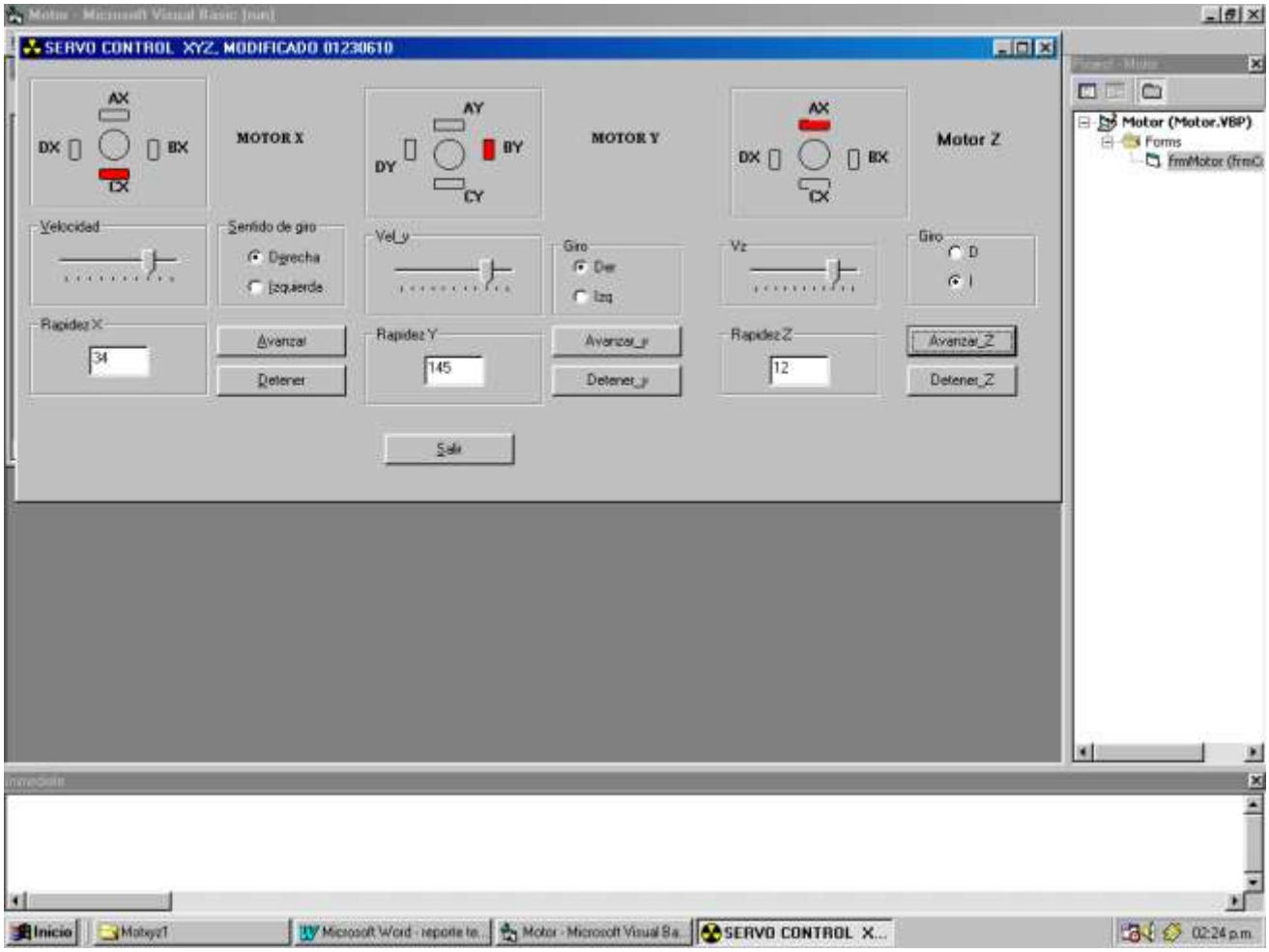

Figura 4.21. Entorno visual.

5 TRABAJO PENDIENTE

Uno de los propósitos que se han establecido después de que el primer prototipo ha llegado a ser funcional, es sustituir el mecanismo de comunicación de puerto paralelo por una interfaz USB, debido a que pocas computadoras personales de generaciones recientes han heredado el estándar de comunicación mediante puerto paralelo. Afortunadamente, hay al menos un dispositivo con interfaz USB compatible pin a pin con el micro controlador usado, por ejemplo el PIC18F4550.

La estructura del programa ensamblador es conveniente para probar diversos algoritmos de control tanto de regulación como de velocidad. El mecanismo hecho con tornillos sin fin ha sido maquinado en forma deficiente, lo que hace útil el sistema en la comprobación del desempeño de dichos algoritmos, especialmente en baja velocidad. Cabe mencionar que solamente ha sido programado el término proporcional en la función (o ley) de control.

En la construcción de la interfaz se han presentado muchos problemas de sincronización debido a las restricciones del lenguaje orientado a objetos por un lado, y a las limitaciones del puerto paralelo por el otro. Es necesaria una interfaz en la que se pueda observar toda la información de interés del proceso, sin problemas de sincronización.

6 CONCLUSIONES

Tal vez el punto principal de reflexión ha cerca de este trabajo esta relacionado al hecho de que el mecanismo ha sido diseñado en su totalidad en el instituto, sin necesidad de recurrir a la integración de componentes foráneos. Se tiene la certeza de que este primer prototipo servirá de mucho en la obtención de un sistema sub micrométrico necesario en la elaboración de elementos ópticos difractivos.

AGRADECIMIENTOS

JCR Agradece al CONACYT el apoyo otorgado para la realización de este dispositivo mediante el proyecto J-50614.“Diseño y Construcción de Sistemas Ópticos Difractivos: “Lentes híbridas, placa cúbica de fase”. Asimismo agradecemos las sugerencias y ayuda en la construcción del robot mecánico al Ing. Marco Antonio de Jesús Ortiz.

REFERENCIAS

[1] J. Castro-Ramos, S. Vazquez-Montiel, J. Hernandez-de-la-Cruz, O. Garcıa-Lievanos y W. Calleja-Arriaga, “Optica difractiva: una revision al diseño y construcción de sistemas ópticos empleando lentes difractivas”, Rev. Mex. Fis.,. 52 (6) 479–500, Dic. 2006.

[2] Lewis C. Eggebrecht, “Interfacing to the IBM Personal Computer”, Sams; 2nd edition (July 1990).

[3] José Fernando Reyes Cortes, “Control de un Robot de Transmisión Directa de dos Grados de Libertad”, Tesis Doctoral, Centro de Investigación Científica y de Educación Superior de Ensenada, Oct. 1997.

[4] José Luis Durán G., J. Eduardo Acosta C., “Diseño de un Servoamplificador Lineal de Potencia del tipo de Transconductancia Para el Control de Motores DC”, Mexicón 92, Guadalajara Jal., México.

[5] Francisco J. Ceballos, “Visual Basic 6, Curso de Programación”, Alfa Omega – Rama, 2002.

[6] http://www.geekhideout.com/iodll.shtml

[7] http://www.latticesemi.com/lit/docs/datasheets/pal_gal/p22v10.pdf

Apéndice A

Divisor de Reloj

```
Name      cnt3_2F2.pld;
Partno    PLAs;
Date      16/05/08;
Revision  01;
Designer  G.G.;
Company   Inaoe;
Assembly  A1;
Location  U11;
Device    g22v10;
    /* contador de 3 bits, con una salida invertida  */
    /* para segundo sincronizador de conteo          */
    /* hay una salida adicional para T0CKIN, y otra  */
    /* aun no usada */
/**  Entradas  **/
pin 1 = clk;            /* 16 Mhz   */

/**  Salidas  **/
pin 23 = QA;
pin 22 = QB;
pin 21 = QC;
pin 20 = Q0;
pin 19 = Q1;
pin 17 = Q2;
pin 18 = QD;

Q0.ar = 'b'0;
Q0.sp = 'b'0;
Q0.oe = 'b'1;
Q1.ar = 'b'0;
Q1.sp = 'b'0;
Q1.oe = 'b'1;
Q2.ar = 'b'0;
Q2.sp = 'b'0;
Q2.oe = 'b'1;

/** Ecuaciones lógicas **/

Q0.d = !Q0;
Q1.d = (!Q1 & Q0) # (Q1 & !Q0);
Q2.d = (Q2 & !Q1) # (Q2 & !Q0) # (!Q2 & Q1 & Q0);
QA = clk;
QB = Q2;
QC = !Q2;
QD = Q2;
```

Apéndice B

Contador para medición de posición.

```
Name      Mod64;
Partno    002;
Date      01190508;
Revision  01;
Designer  G.G.;
Company   INAOE;
Assembly  PIC16F877;
Location  U1, U2, U9, U12;
Device    g22v10;

/*****************************************************************/
/* Este es un contador de 6 bits UP/DOWN  sincrono, con         */
/* habilitaci¢n de cuenta, y control de modo.                    */
/* se construyen dos de 3 bits en cascada                        */
/* GAL22V10                                                      */
/* MINIMIZACION QUICK                                            */
/*****************************************************************/

/**  Entradas  **/

pin 1 = clk;            /* Reloj           */
pin 2 = DA;
pin 3 = DB;             /* Entradas de encoder   */
pin 4 = INICIO;
pin 5 = ENAB;

/**  Salidas  **/

pin [21..23] = [QC,QB,QA];         /* Salidas */
pin [18..20] = [Q2,Q1,Q0];

pin 17 = QAS;
pin 16 = QBS;
pin 15 = QBDL;
pin 14 = QASxQBS;

dir = QBDL;

mode = dir;          /* campo de modo */
up = mode:1;         /* modo ascendente   */
down = mode:0;       /* modo descendente  */

field flips = [QASxQBS,QBDL,Q2,Q1,Q0,QC,QB,QA,QAS,QBS];
```

```
flips.ar = !INICIO;
flips.oe = !ENAB;

QA.sp = 'b'0;
QB.sp = 'b'0;
QC.sp = 'b'0;
Q0.sp = 'b'0;
Q1.sp = 'b'0;
Q2.sp = 'b'0;
QAS.sp = 'b'0;
QBS.sp = 'b'0;
QBDL.sp = 'b'0;
QASxQBS.sp = 'b'0;

/** Ecuaciones lógicas **/

QAS.D = DA;

QBS.D = DB;

QBDL.D = QBS;

QASxQBS =  QAS $ QBS;

cin1 = QAS # !(QBS $ QBDL);

QC.D = (!QC & QB & QA & up & !cin1)#(QC & !QB & !QA & up & !cin1)
#(QC & !QB & QA & up & !cin1)#(QC & QB & !QA & up & !cin1)
#(!QC & !QB & !QA  & down & !cin1)#(QC & !QB & QA  & down & !cin1)
#(QC & QB & !QA  & down & !cin1)#(QC & QB & QA  & down & !cin1)
#(QC & !QB & !QA  & up & cin1)#(QC & !QB & QA  & up & cin1)
#(QC & QB & !QA  & up & cin1)#(QC & QB & QA  & up & cin1)
#(QC & !QB & !QA  & down & cin1)#(QC & !QB & QA  & down & cin1)
#(QC & QB & !QA  & down & cin1)#(QC & QB & QA  & down & cin1);

QB.D = (!QC & !QB & QA  & up & !cin1)#(!QC & QB & !QA  & up & !cin1)
#(QC & !QB & QA  & up & !cin1)#(QC & QB & !QA  & up & !cin1)
#(!QC & !QB & !QA  & down & !cin1)#(!QC & QB & QA  & down & !cin1)
#(QC & !QB & !QA  & down & !cin1)#(QC & QB & QA  & down & !cin1)
#(!QC & QB & !QA  & up & cin1)#(!QC & QB & QA  & up & cin1)
#(QC & QB & !QA  & up & cin1)#(QC & QB & QA  & up & cin1)
#(!QC & QB & !QA  & down & cin1)#(!QC & QB & QA  & down & cin1)
#(QC & QB & !QA  & down & cin1)#(QC & QB & QA  & down & cin1);

QA.D = (!QC & !QB & !QA  & up & !cin1)#(!QC & QB & !QA  & up & !cin1)
#(QC & !QB & !QA  & up & !cin1)#(QC & QB & !QA  & up & !cin1)
#(!QC & !QB & !QA  & down & !cin1)#(!QC & QB & !QA  & down & !cin1)
#(QC & !QB & !QA  & down & !cin1)#(QC & QB & !QA  & down & !cin1)
```

```
#(!QC & !QB & QA  & up & cin1)#(!QC & QB & QA  & up & cin1)
#(QC & !QB & QA  & up & cin1)#(QC & QB & QA  & up & cin1)
#(!QC & !QB & QA  & down & cin1)#(!QC & QB & QA  & down & cin1)
#(QC & !QB & QA  & down & cin1)#(QC & QB & QA  & down & cin1);

carry1 = !((QC & QB & QA  & up & !cin1)#(!QC & !QB & !QA  & down & !cin1));

Q2.D = (!Q2 & Q1 & Q0 & up & !carry1)#(Q2 & !Q1 & !Q0 & up & !carry1)
#(Q2 & !Q1 & Q0 & up & !carry1)#(Q2 & Q1 & !Q0 & up & !carry1)
#(!Q2 & !Q1 & !Q0  & down & !carry1)#(Q2 & !Q1 & Q0  & down & !carry1)
#(Q2 & Q1 & !Q0  & down & !carry1)#(Q2 & Q1 & Q0  & down & !carry1)
#(Q2 & !Q1 & !Q0  & up & carry1)#(Q2 & !Q1 & Q0  & up & carry1)
#(Q2 & Q1 & !Q0  & up & carry1)#(Q2 & Q1 & Q0  & up & carry1)
#(Q2 & !Q1 & !Q0  & down & carry1)#(Q2 & !Q1 & Q0  & down & carry1)
#(Q2 & Q1 & !Q0  & down & carry1)#(Q2 & Q1 & Q0  & down & carry1);

Q1.D = (!Q2 & !Q1 & Q0  & up & !carry1)#(!Q2 & Q1 & !Q0  & up & !carry1)
#(Q2 & !Q1 & Q0  & up & !carry1)#(Q2 & Q1 & !Q0  & up & !carry1)
#(!Q2 & !Q1 & !Q0  & down & !carry1)#(!Q2 & Q1 & Q0  & down & !carry1)
#(Q2 & !Q1 & !Q0  & down & !carry1)#(Q2 & Q1 & Q0  & down & !carry1)
#(!Q2 & Q1 & !Q0  & up & carry1)#(!Q2 & Q1 & Q0  & up & carry1)
#(Q2 & Q1 & !Q0  & up & carry1)#(Q2 & Q1 & Q0  & up & carry1)
#(!Q2 & Q1 & !Q0  & down & carry1)#(!Q2 & Q1 & Q0  & down & carry1)
#(Q2 & Q1 & !Q0  & down & carry1)#(Q2 & Q1 & Q0  & down & carry1);

Q0.D = (!Q2 & !Q1 & !Q0  & up & !carry1)#(!Q2 & Q1 & !Q0  & up & !carry1)
#(Q2 & !Q1 & !Q0  & up & !carry1)#(Q2 & Q1 & !Q0  & up & !carry1)
#(!Q2 & !Q1 & !Q0  & down & !carry1)#(!Q2 & Q1 & !Q0  & down & !carry1)
#(Q2 & !Q1 & !Q0  & down & !carry1)#(Q2 & Q1 & !Q0  & down & !carry1)
#(!Q2 & !Q1 & Q0  & up & carry1)#(!Q2 & Q1 & Q0  & up & carry1)
#(Q2 & !Q1 & Q0  & up & carry1)#(Q2 & Q1 & Q0  & up & carry1)
#(!Q2 & !Q1 & Q0  & down & carry1)#(!Q2 & Q1 & Q0  & down & carry1)
#(Q2 & !Q1 & Q0  & down & carry1)#(Q2 & Q1 & Q0  & down & carry1);
```

Apéndice C

Selector de dispositivos

```
Name         decoder CNT3MOT2.pld;
Partno       01;
Revision     02;
Date         19/05/08;
Designer     GABO;
Company      INAOE;
Assembly     Motores DC;
Location      U21;
Device       G22V10;

/*****************************************/
/*   habilitacion de  Salida  (OE) de      */
/*  chips medidores de posicion,           */
/*  y salidas analogicas de la tarjeta     */
/*  de control de 4 servos  de DC          */
/*****************************************/

/**  Entradas  **/

pin 6 = DEC0;    /* De RC4  */
pin 7 = DEC1;    /* De RC5  */
pin 8 = DEC2;    /* De RC6  */
pin 9 = DEC3;    /* De RC7  */
pin 10 = PULSO;  /* Del divisor de reloj */
/**  Salidas  **/
pin  16 = DAC3;
pin  17 = DAC2;
pin  18 = DAC1;
pin  19 = DAC0;
pin  20 = CNT3;
pin  21 = CNT2;
pin  22 = CNT1;
pin  23 = CNT0;
/**  Ecuaciones logicas  **/
!CNT0 = !DEC3 & !DEC2 & !DEC1 & !DEC0;
!CNT1 = !DEC3 & !DEC2 & !DEC1 & DEC0;
!CNT2 = !DEC3 & !DEC2 & DEC1 & !DEC0;
!CNT3 = !DEC3 & !DEC2 & DEC1 & DEC0;
!DAC0 = !DEC3 & DEC2 & !DEC1 & !DEC0;
!DAC1 = !DEC3 & DEC2 & !DEC1 & DEC0;
!DAC2 = !DEC3 & DEC2 & DEC1 & !DEC0;
!DAC3 = !DEC3 & DEC2 & DEC1 & DEC0;
```

Apéndice D

Interfaz grafica (Visual Basic)

```
Option Explicit
Private Declare Sub PortOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Byte)
Dim salidasx(4) As Byte
Dim Estadox As Integer
Dim salidasy(4) As Byte
Dim Estadoy As Integer
Dim salidasz(4) As Byte
Dim Estadoz As Integer
Dim Conteox As Integer
Dim Variablex As Integer
Dim Direccionx As Integer
Dim Conteoy As Integer
Dim Variabley As Integer
Dim Direcciony As Integer
Dim Conteoz As Integer
Dim Variablez As Integer
Dim Direccionz As Integer

Const MAXX = 4
Const Derechax = 1
Const Reversax = 2
Const INTERVALO_INICIALX = 10
Const COLOR_APAGADOX = &HC0C0C0

Const MAXY = 4
Const Derechay = 1
Const Reversay = 2
Const INTERVALO_INICIALY = 10
Const COLOR_APAGADOY = &HC0C0C0

Const MAXZ = 4
Const Derechaz = 1
Const Reversaz = 2
Const INTERVALO_INICIALZ = 10
Const COLOR_APAGADOZ = &HC0C0C0

Private Sub cmdAvanzarx_Click()
    tmrx.Interval = Slider1.Value * INTERVALO_INICIALX
     If optDerecha = True Then
        Direccionx = Derechax
    Else
        Direccionx = Reversax
    End If
```

```
    Estadox = 0
    Conteox = 0
End Sub

Public Sub Moverx(ByVal Direx As Byte)

    salidasx(0) = 6
    salidasx(1) = 5
    salidasx(2) = 9
    salidasx(3) = 10
    Ledsx(Estadox).BackColor = COLOR_APAGADOX
    If Direx = Derechax Then
        Estadox = (Estadox + 1) Mod MAXX
      Else
        Estadox = (Estadox + MAXX - 1) Mod MAXX
    End If
     Conteox = Conteox + 1
    Call PortOut(890, 8)
    Call PortOut(888, Variablex)
     Ledsx(Estadox).BackColor = &HFF
    If Conteox = Variablex Then
    tmrx.Interval = 0
    End If
End Sub
Public Sub Movery(ByVal Direy As Byte)

    salidasy(0) = 6
    salidasy(1) = 5
    salidasy(2) = 9
    salidasy(3) = 10
    Ledsy(Estadoy).BackColor = COLOR_APAGADOY
    If Direy = Derechax Then
        Estadoy = (Estadoy + 1) Mod MAXY
       Else
        Estadoy = (Estadoy + MAXY - 1) Mod MAXY
    End If
        Conteoy = Conteoy + 1
    Call PortOut(890, 12)
    Call PortOut(888, Variabley)
    Ledsy(Estadoy).BackColor = &HFF
    If Conteoy = Variabley Then
    Tmry.Interval = 0
    End If
End Sub
Public Sub Moverz(ByVal Direz As Byte)
    salidasz(0) = 6
    salidasz(1) = 5
    salidasz(2) = 9
    salidasz(3) = 10
```

```
    Ledsz(Estadoz).BackColor = COLOR_APAGADOZ
    If Direz = Derechaz Then
       Estadoz = (Estadoz + 1) Mod MAXZ
      Else
       Estadoz = (Estadoz + MAXZ - 1) Mod MAXZ
    End If
       Conteoz = Conteoz + 1
    Call PortOut(890, 0)
    Call PortOut(888, Variablez)

    Ledsz(Estadoz).BackColor = &HFF
    If Conteoz = Variablez Then
    Tmrz.Interval = 0
    End If
End Sub

Private Sub CmdAvanzary_Click()
    Tmry.Interval = Slider2.Value * INTERVALO_INICIALY
     If optDery = True Then
       Direcciony = Derechay
    Else
       Direcciony = Reversay
    End If
     Estadoy = 0
     Conteoy = 0
End Sub

Private Sub CmdAvanzarz_Click()
    Tmrz.Interval = Slider2.Value * INTERVALO_INICIALZ
     If Optd = True Then
       Direccionz = Derechaz
    Else
       Direccionz = Reversaz
    End If
     Estadoz = 0
     Conteoz = 0
End Sub

Private Sub cmdDetenerx_Click()
    tmrx.Interval = 0
    Conteox = 0
    Variablex = 127
End Sub

Private Sub CmdDetenery_Click()
    Tmry.Interval = 0
    Conteoy = 0
    Variabley = 127
End Sub
```

```
Private Sub CmdDetenerz_Click()
    Tmrz.Interval = 0
    Conteoz = 0
    Variablez = 127
End Sub

Private Sub cmdSalir_Click()
      End
End Sub


Private Sub Contadorx_LostFocus()
Conteox = 0
Variablex = Contadorx.Text
Variablex = Variablex + 127
End Sub

Private Sub Contadory_LostFocus()
Conteoy = 0
Variabley = Contadory.Text
End Sub

Private Sub Contadorz_LostFocus()
Conteoz = 0
Variablez = Contadorz.Text
End Sub

Private Sub Form_Load()
     Conteox = 0
    Conteoy = 0

End Sub

Private Sub optIzquierda_Click()

End Sub


Private Sub HScroll1_Change()

End Sub

Private Sub Slider1_Click()
    tmrx.Interval = Slider1.Value * INTERVALO_INICIALX
    Conteox = 0
End Sub

Private Sub Slider2_Click()
```

```
   Tmry.Interval = Slider2.Value * INTERVALO_INICIALY
   Conteoy = 0
End Sub

Private Sub Tmrz_Timer()
Moverz (Direccionz)
End Sub

Private Sub Slider3_Click()
 Tmrz.Interval = Slider3.Value * INTERVALO_INICIALY
End Sub

Private Sub Tmry_Timer()
Movery (Direcciony)
End Sub

Private Sub tmrx_Timer()
    Moverx (Direccionx)
End Sub
```