INGENIERIA EN ELECTRONICA

ROBOTICA

"Medidor de Velocidad y Posición con un Encoder"

Alumno:
Elfrich González Arévalo

Profesor:
F. Hugo Ramírez Leyva

Huajuapan de León, Oaxaca a 12 de Mayo de 2009

Resumen
En esta práctica se uso un encoder para medir la posición y velocidad de un motor de CD. Con una computadora y usando el software LABVIEW se creó e implemento un interfaz grafica (figura 9) para mostrar los valores gráficamente correspondientes al número de vueltas, sentido, posición y velocidad. Con el microcontrolador ATMEGA8 de ATMEL se implemento una interfaz física (figura 4) que sirvió de conexión para los datos entre el encoder y la interfaz grafica de la computadora.

Tabla de contenidos

Introducción
Sensores de Movimiento (Posición, Velocidad y Aceleración)
Electromecánicos: Una masa con un resorte y un amortiguador.

Piezo-eléctricos: Una deformación física del material causa un cambio en la estructura cristalina y así cambian las características eléctricas.

Piezo-resistivos: Una deformación física del material cambia el valor de las resistencias del puente.

Capacitivos: El movimiento paralelo de una de las placas del condensador hace variar su capacidad.

Efecto Hall: La corriente que fluye a través de un semiconductor depende de un campo magnético.

Los sensores de movimiento permiten la medida de la fuerza gravitatoria estática (cambios de inclinación), la medida de la aceleración dinámica (aceleración, vibración y choques), y la medida inercial de la velocidad y la posición (la velocidad midiendo un eje y la posición midiendo los dos ejes).

Midiendo la aceleración se puede determinar la velocidad y la posición. La Aceleración Integrada: una para velocidad, dos veces para la distancia. Medida Relativa desde una posición inicial:

Velocidad=A*t (ecu. 1) Distancia = (A/2)*t^2 (ecu. 2) donde, A es la aceleración y t el tiempo.

Puede ser exacta para periodos cortos de tiempo, pero la exactitud se degrada proporcionalmente al cuadrado del tiempo de integración. Es posible una exactitud Posicional de 2cm sobre un segundo. La exactitud Posicional se degrada a 20m después de 10 segundos de integración.

Giróscopos monolíticos

Analog Devices ha fabricado el primer giroscopio monolítico para realizar medidas angulares (mide la velocidad en que gira sobre su propio eje). Puede medir cambios de inclinación o cambios de dirección integrando la velocidad angular.

Medida de la velocidad angular

La velocidad angular mide la rapidez en que gira un objeto alrededor de un eje. Integrando la velocidad angular se miden los cambios de inclinación o cambios de dirección. La velocidad angular se mide midiendo la fuerza de Coriolis.

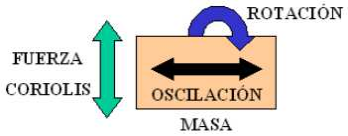


Figura 1. Fuerza de Coriolis.

¿Cuál es la fuerza de Coriolis?

Cuando un objeto se mueve de una manera periódica (oscilando o girando), girando el objeto en un plano ortogonal a su movimiento periódico causa una fuerza de traslación en la otra dirección ortogonal (figura 1).

La velocidad angular se determina por la medida de la aceleración de Coriolis (A_{cor}).
A_{cor} = 2*(ω velocidad de la masa) (ecu. 3) donde, ω es la velocidad angular aplicada

La Velocidad aplicada por medio una estructura rígida resonando a 18KHz acoplada a un marco de un acelerómetro. La aceleración de Coriolis estará en la misma frecuencia y fase que el resonador, con tal que la baja velocidad de vibración externa pueda cancelarse. La familia ADXRS150 usa dos traviesas (masas) resonando en antifase. El funcionamiento diferencial permite el rechazo de muchos errores.

Sensores de Posición

Los sensores de posición pueden dar según su construcción o montaje, una posición lineal o angular.

Electromecánicos: lo forman los Finales de Carrera o Microrruptores. Se sitúan en puntos estratégicos a detectar, en sistemas industriales y máquinas en general. Conmutan directamente cualquier señal eléctrica. Tienen una vida limitada. Solo pueden detectar posiciones determinadas, debido a su tamaño.

Magnéticos: lo forman los Detectores de Proximidad Magnéticos, que pueden ser los de Efecto Hall y los Resistivos, típicos en aplicaciones industriales.

Inductivos: lo forman los Detectores de Proximidad Inductivos, los Sincros y Resolvers, los RVDT (Rotatory Variable Differential Transformer) y LVDT (Lineal Variable Differential Transformer).

Potenciométricos: lo forman los Potenciómetros lineales o circulares.

Ópticos: lo forman las Células fotoeléctricas y los Encoders.

Sensores de Posición Resistivos

Los potenciómetros se utilizan también como sensores de posición. Mediante una tensión DC de referencia muy estable, el cursor da una salida proporcional al ángulo del eje. Las consideraciones a tener en cuenta son el número de maniobras que va a realizar, para calcular la vida del mismo, ya que existe un contacto mecánico entre la resistencia y el terminal del cursor. Bourns fabrica una serie de potenciómetros especializados para este trabajo, utilizan como elemento resistivo el bobinado que es muy lineal, y también el plástico conductor que a parte de la linealidad ofrece una vida muy larga.

Existe en el mercado una variedad de elementos resistivos que se utilizan en los potenciómetros, el elemento más popular es el carbón, su mejor característica es el precio, pero como inconvenientes tiene las variaciones de temperatura y su vida; el *cermet* es una combinación de un material *CER*ámico y *MET*al que mejora muchísimo las características del carbón. Después se encuentra el bobinado, que sus principales ventajas son el bajo coeficiente de temperatura, su vida mecánica, bajo ruido, alta disipación, y estabilidad con el tiempo. Otro elemento utilizado es el plástico conductor que mejora en todas las características respecto a los demás elementos, pero tiene un precio superior.

Encoder

Es un dispositivo electromecánico, que convierte la posición angular de su eje en una señal digital eléctrica. Conectado a la electrónica adecuada y a través de los apropiados vínculos mecánicos, el encoder permite medir desplazamientos angulares, movimientos lineales y circulares, y velocidades rotacionales y aceleraciones. Varias técnicas pueden ser utilizadas para la detección de un movimiento angular: capacitiva, inductiva, potenciométrica y fotoeléctrica.

Encoders ópticos

Con los foto-interruptores y los reflectivos se pueden montar los encoders ópticos, formados por un disco que tiene dibujados segmentos para ser detectados por los sensores. Existen dos tipos de encoders, los encoders Incrementales y encoders Absolutos.

Encoder Incrementales: permiten que un sensor óptico detecte el número de segmentos que dispone el disco y otro sensor detecte la posición cero de dicho disco (figura 2).

Encoders Absolutos: permiten conocer la posición exacta en cada momento sin tener que dar una vuelta entera para detectar el punto cero del disco. La diferencia es que se necesitan varios sensores ópticos y el disco tendría una codificación tipo Manchester, por ejemplo (figura 3).

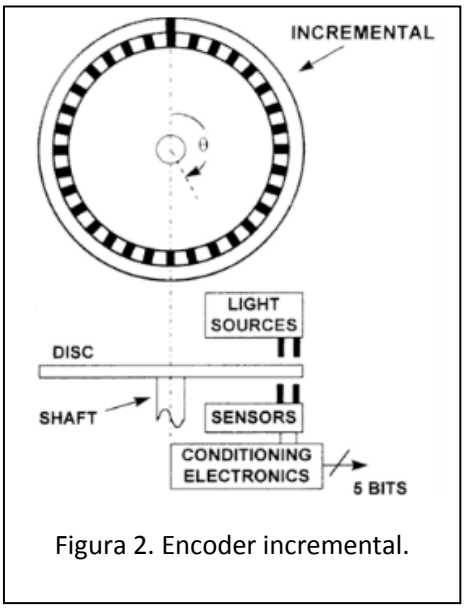


Figura 2. Encoder incremental.

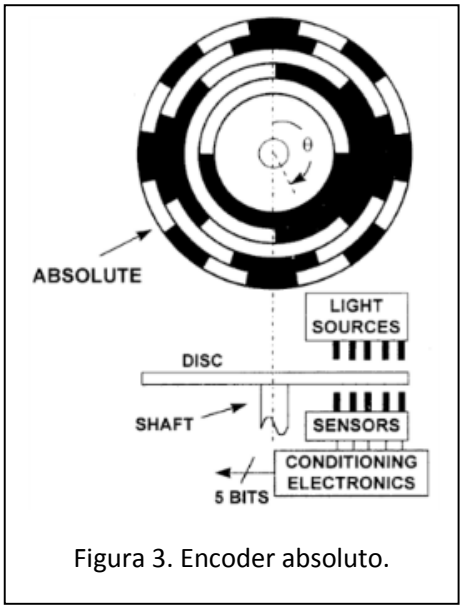


Figura 3. Encoder absoluto.

Objetivos

- Verificar que el encoder ENA1J-B28 de a la salida de la fase A 64 pulsos por vuelta (caracterización del encoder).
- Aprender la forma en la cual trabaja un enconder y usarlo para medir la posición y velocidad de un motor de CD.
- Mediante Labview crear una interfaz grafica que muestre la posición, velocidad del motor y graficar los resultados.
- Implementar en un micro una interfaz RS-232 para que se comunique con la computadora y el encoder.
- Determinar una ecuación teórica para determinar la velocidad en función de los pulsos y la unidad de tiempo.
- Hacer un sistema de montaje en el cual se acoplen la flecha del enconder con el eje de un motor de DC, de tal manera que ambos permanezcan estáticos mientras sus flechas giran.

Descripción del sistema

En la figura 4 se ven las conexiones físicas de los sistemas empleados en la práctica. Estos sistemas son el MAX232, ATMEGA8, encoder ENA1J-B28, motor de CD y una computadora.

Las salidas del encoder, la fase A y fase B son conectadas al microcontrolador el cual se hará cargo de calcular la posición, número de vueltas y velocidad del motor de CD. Una vez calculados estos valores el microcontrolador mediante una conexión serial manda los valores a la computadora para que la interfaz hecha en LABVIEW muestre los valores gráficamente. El circuito integrado MAX232 cambia los niveles TTL a los del estándar RS-232 cuando se hace una transmisión, y cambia los niveles RS-232 a TTL cuando se tiene una recepción. El circuito típico se muestra en la figura 4.

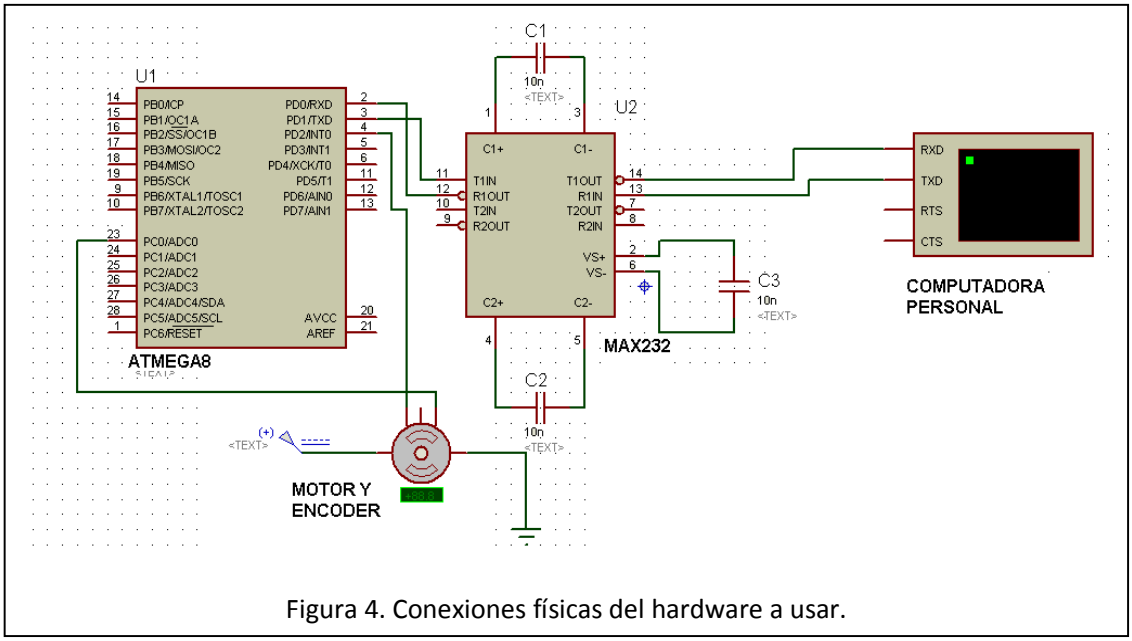


Figura 4. Conexiones físicas del hardware a usar.

Teoría

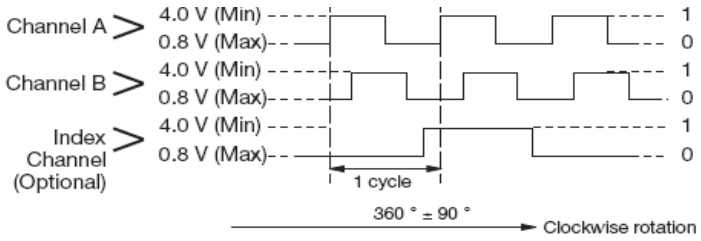
Los encoders de cuadratura usualmente tienen 2 salidas digitales llamadas fase A y fase B (figura 5). Cuando el encoder rota, la fase A y la fase B producen un pulso de onda cuadrado donde cada pulso representa una fracción de un giro del eje del encoder. Los encoders son medidos para un cierto número de pulsos (o cuentas) por revolución completa del eje. Comúnmente la relación cuenta/revolución están en 50, 100, 128, 200, 250, 256, 500, etc. Por el conteo del número de pulsos de salida en una de las fases empezando desde el tiempo 0, podemos calcular la distancia rotacional total que el encoder ha viajado.

A menudo, sin embargo, se quiere la posición actual y no solo la distancia total viajada. Para esto es necesario saber no solo cuán lejos el encoder ha viajado, sino también cual ha sido la dirección. Para hacer esto es necesario usar ambas salidas o fases del encoder de cuadratura.

El pulso desde la fase A y la fase B en el encoder de cuadratura estará siempre alineado 90 grados fuera de fase (dicho de otro modo: ¼ de longitud de onda aparte). Esta relación de fase especial nos permite saber la distancia y dirección que el eje del encoder ha rotado desde la el tiempo 0.

Para hacer esto, consideramos a la fase A como el contador de la distancia. Sobre cada flanco de subida de la fase A, contaremos uno “tic” de distancia, pero necesitamos saber la dirección. Observando la forma de onda en cuadratura de abajo. Advertimos que cuando nos movemos hacia delante en el tiempo (izquierda a derecha), la fase B esta siempre abajo (cero lógico) al flanco de subida de la fase A. Si nos movemos hacia atrás en el tiempo (derecha a izquierda) advertimos que es la misma situación. Entonces, si la fase A es nuestro contador, la fase B indica la dirección.

Ejemplo de una forma de onda de un encoder de cuadratura:



Tiempo: <----------->

Rotación

Adelante: >----------->

Atrás: <-----------<

Figura 5. Formas de onda de un encoder de cuadratura.

Velocidad angular

La velocidad angular es la rapidez con la que varía el ángulo (figura 6) en el tiempo y se mide en radianes / segundos. (2 π [radianes] = 360°)

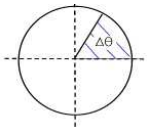


Figura 6. Variación del ángulo.

Por lo tanto si el ángulo es de 360 grados (una vuelta) y se realiza por ejemplo en un segundo, la velocidad angular es: 2 π [rad / s].
Si se dan dos vueltas en 1 segundo la velocidad angular es 4 π [rad / s].
Si se da media vuelta en 2 segundos es 1/2 π [rad / s].
La velocidad angular se calcula como la variación del ángulo sobre la variación del tiempo.

$$\omega = \frac{\Delta\theta}{\Delta t}$$ (ecu. 3) donde, teta es el ángulo y t el tiempo.

Análisis

Una vuelta en el eje del encoder equivale a 360 grados. 64 pulsos nos dan una vuelta. Así mediante esta relación existente podemos obtener que valor (en grados) le correspondería a un pulso:
64 pulsos -> 1 vuelta. 1 vuelta -> 360^o => 64 pulsos -> 360^o.
1 pulso -> X grados.
$X=360^o/64=5.625^o$

Con este valor sabemos que cada vez que el encoder de a la salida de la fase A un flanco de subida estaremos aumentando la posición 5.625^o.

Para saber la velocidad tomamos la medición de la posición (P0) y después de un determinado tiempo (muy corto, cada 50 ms.) tomamos la nueva posición (P1), así usando la ecuación 3 tenemos: $\Delta\theta=(P1-P0) \Rightarrow \omega = \Delta\theta/50$ [grados/ms].

Procedimiento

El primer paso consistió acoplar la flecha del enconder con el eje de un motor de DC. Esto fue sencillo usando un motor grande de CD y una banda de goma. El acoplamiento se observa en la figura 7.

Figura 7. Acoplamiento del eje del encoder y el eje del motor.

El segundo paso fue realizar el programa del microcontrolador e implementar los resultados obtenidos en el análisis, la siguiente figura muestra un diagrama de flujo. En el apéndice A se muestra el código fuente en C para micros.

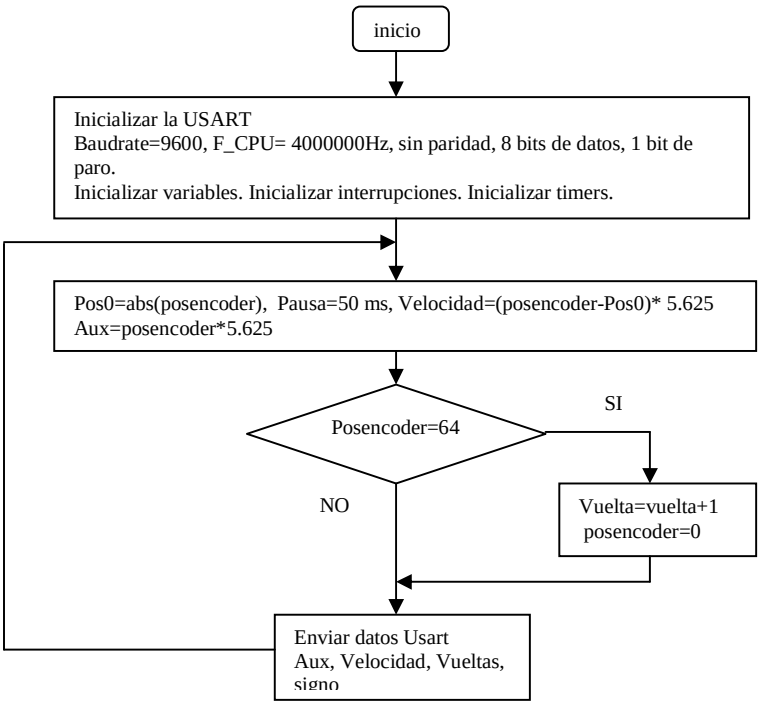


Figura 8. Diagrama de flujo del programa en C para el ATMEGA8.

En el tercer paso se creó la interfaz grafica en LABVIEW, la figura 9 es la interfaz grafica final y la figura 10 muestra el diagrama a bloques interno de la figura 9.

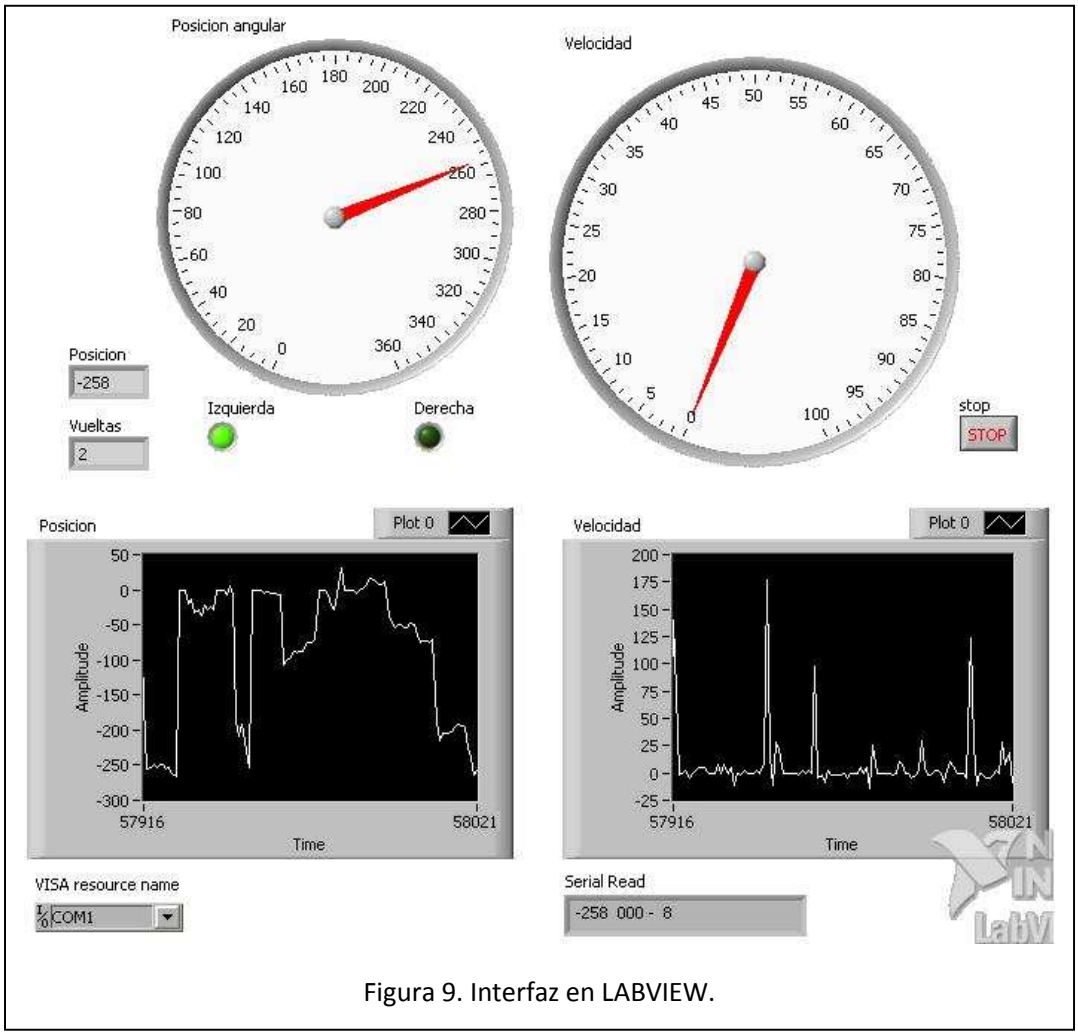


Figura 9. Interfaz en LABVIEW.

La figura 9 es una imagen con los valores obtenidos al probar el sistema completo usando un generador de onda cuadrada, esto para verificar que el código del programa implementado en el microcontrolador funcionará correctamente. La frecuencia de la onda cuadrada para probar el sistema fue de hasta 60 Hz.

La figura 10 en uno de sus bloques muestra que la velocidad es tomada como un valor entero, esto es porque al probar un tipo de valor diferente al entero en el microcontrolador, el mismo no respondía con la suficiente velocidad para hacer las mediciones. Aunque en el diagrama de flujo (figura 8) se realizaron multiplicaciones con valores que tienen hasta 3 digitos después del punto, no significa que el micro aceptara del todo los decimales, lo único que hace es truncar el valor al entero más cercano.

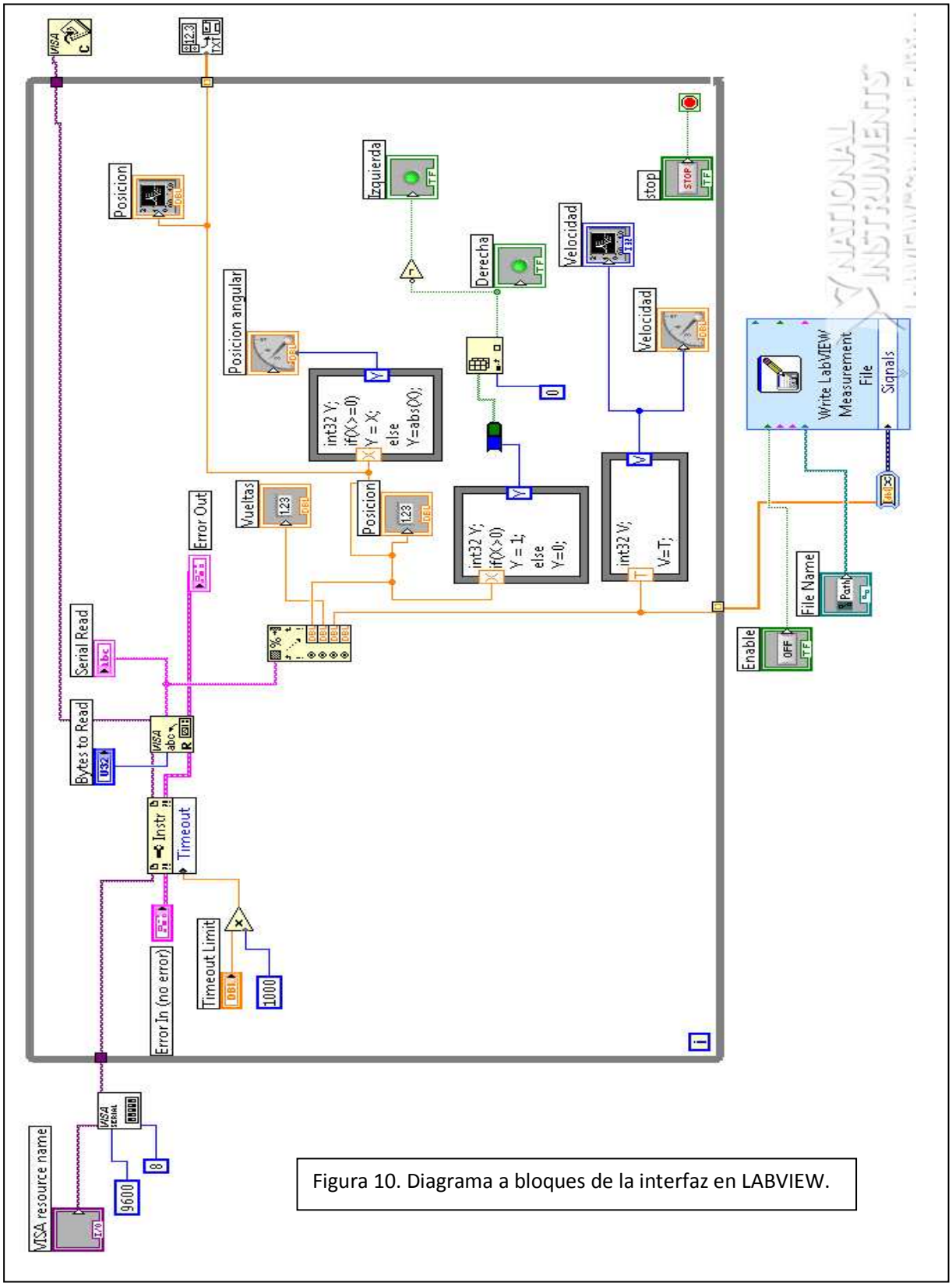


Figura 10. Diagrama a bloques de la interfaz en LABVIEW.

Resultados experimentales

La figura 11 muestra los resultados obtenidos para la caracterización del encoder. En esta imagen el encoder estaba en la posición 132 con 68 pulsos, los cursores indican los 64 pulsos necesarios para completar una vuelta completa (360 grados).

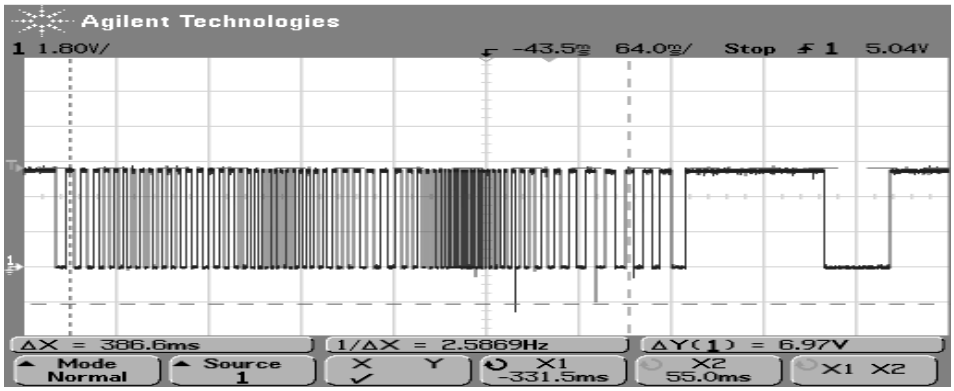


Figura 11. Pulsos de salida del encoder de la fase A.

Como ya se había mencionado antes, la figura 9 muestra la grafica de posición y velocidad, estas graficas se obtuvieron al quitar el encoder y probar el sistema haciendo uso de un generador de funciones.

Conclusiones

El uso de encoders facilita el control sobre el posicionamiento y velocidad de un motor. Su uso es fácil y no requiere de ningún tipo de acondicionamiento previo para las ondas de salida de las fases A y B. El análisis para las relaciones vueltas-grados fue bastante directo, lo que permitió sin problemas la implementación del cálculo para la posición y velocidad en un microcontrolador, excepto por el tipo de variables, ya que a pesar de que se intento usar una variable del tipo long para la velocidad, esto no funciono. La causa de este problema no se logro hallar. Pero la salida a este problema fue el truncamiento al valor entero más próximo que el mismo micro realiza al pasarle a un entero el valor de una variable con decimas.

En cuanto a la implementación de la interfaz grafica en LABVIEW, no hubo ningún problema, ya que todo fue cuestión de tiempo y de saber buscar en la ayuda del mismo software, el cómo pasar datos y convertirlos a otro tipo de dato como booleanos, a un array, etc.

Bibliografía

- *Mayné Jord.* Sensores Acondicionadores y Procesadores de señal.
- McComb Gordon. The Robot Builder's Bonanza, tercera edición, McGraw-Hill.
- http://www.elcis.com/SPAGNOLO/generalita/generalitacentro.html, 9/05/09.
- http://www.fisicapractica.com/velocidad-angular-mcu.php, 9/05/09.
- www.avrfreaks.com, 28/04/09.
- www.avrtutor.com, 21/04/09.
- http://www.sc.ehu.es/sbweb/fisica/cinematica/circular/circular.htm, 9/05/09.
- http://es.farnell.com/bourns/ena1j-b28-l00064/rotary-encoder-output-type-square/dp/1783828, 9/05/09.

Apéndice

Código fuente del programa principal

```
// File Name            : encodertest.c
#include <avr/io.h>                 // include I/O definitions (port names, pin
names, etc)
#include <avr/interrupt.h>          // include interrupt support
#include <math.h>
#include <stdlib.h>
#include "global.h"                 // include our global settings
#include "uart.h"                   // include uart function library
#include "rprintf.h"                // include printf function library
#include "timer.h"                  // include timer function library (timing,
PWM, etc)
#include "vt100.h"                  // include VT100 terminal support
#include "encoder.h"
void encoderTest(void);
//s32 position0;
s32 speed0;

int main(void)
{
        uartInit();
        uartSetBaudRate(9600);
        timerInit();
        rprintfInit(uartSendByte);
        vt100Init();
        // clear the terminal screen
        vt100ClearScreen();
        // run the test
        encoderTest();
        return 0;
}
```

```
void encoderTest(void)
{
        s32 startpos;
        int revolucion=0;
        int aux=0
                // initialize the encoders
        encoderInit();
        rprintf("\r\nPractica uno, prueba del encoder \r\n");
        while(1)
        {
startpos = abs(encoderGetPosition(0));
timerPause(50);
speed0 = (abs(encoderGetPosition(0)) - startpos)*2.8125;
                aux=(encoderGetPosition(0)*2.8125);



                if (abs(encoderGetPosition(0))==64)

        {revolucion=revolucion+1;encoderSetPosition(0,0);
                }

                rprintfProgStrM(" ");
                rprintfNum(10, 4, TRUE, ' ', aux);
                rprintfProgStrM(" ");
                rprintfNum(10, 4, TRUE, '0', revolucion);
                rprintfProgStrM(" ");
                rprintfNum(10, 4, TRUE, ' ', speed0);
                rprintfCRLF();
        }
}
```

```
// File Name            : global.h

#ifndef GLOBAL_H
#define GLOBAL_H

#include "avrlibdefs.h"
#include "avrlibtypes.h"

// CPU clock speed
#define F_CPU       4000000                     // 4MHz processor
#define CYCLES_PER_US ((F_CPU+500000)/1000000)      // cpu cycles per microsecond

#endif
```

```
// File Name            : 'encoder.c'

#include <avr/io.h>
#include <avr/interrupt.h>
#include "global.h"
#include "encoder.h"

volatile EncoderStateType EncoderState[NUM_ENCODERS];

void encoderInit(void)
{
        u08 i;

        // initialize/clear encoder data
        for(i=0; i<NUM_ENCODERS; i++)
        {
                EncoderState[i].position = 0;

        }


#ifdef ENC0_SIGNAL
                // set interrupt pins to input and apply pullup resistor
                cbi(ENC0_PHASEA_DDR, ENC0_PHASEA_PIN);
                sbi(ENC0_PHASEA_PORT, ENC0_PHASEA_PIN);
                // set encoder direction pin for input and apply pullup resistor
                cbi(ENC0_PHASEB_DDR, ENC0_PHASEB_PIN);
                sbi(ENC0_PHASEB_PORT, ENC0_PHASEB_PIN);
                // configure interrupts for any-edge triggering
                sbi(ENC0_ICR, ENC0_ISCX0);
                cbi(ENC0_ICR, ENC0_ISCX1);
                // enable interrupts
                sbi(IMSK, ENC0_INT);            // ISMK is auto-defined in
encoder.h
        #endif
// enable global interrupts
        sei();
}


// encoderOff() disables hardware and stops encoder position updates
void encoderOff(void)
{
        // disable encoder interrupts
        #ifdef ENC0_SIGNAL
                // disable interrupts
                sbi(IMSK, INT0);      // ISMK is auto-defined in
encoder.h
        #endif
        #ifdef ENC1_SIGNAL
                // disable interrupts
                sbi(IMSK, INT1);      // ISMK is auto-defined in
encoder.h
        #endif
        #ifdef ENC2_SIGNAL
                // disable interrupts
                sbi(IMSK, INT2);      // ISMK is auto-defined in
encoder.h
        #endif
        #ifdef ENC3_SIGNAL
                // disable interrupts
                sbi(IMSK, INT3);      // ISMK is auto-defined in
encoder.h
        #endif
}

s32 encoderGetPosition(u08 encoderNum)
{

        if(encoderNum < NUM_ENCODERS)
                return EncoderState[encoderNum].position;
        else
                return 0;
}

void encoderSetPosition(u08 encoderNum, s32 position)
{
        // sanity check
        if(encoderNum < NUM_ENCODERS)
                EncoderState[encoderNum].position = position;
        // else do nothing
}

#ifdef ENC0_SIGNAL
//! Encoder 0 interrupt handler
SIGNAL(ENC0_SIGNAL)
{

        if( ((inb(ENC0_PHASEA_PORTIN) &
(1<<ENC0_PHASEA_PIN)) == 0) ^
                ((inb(ENC0_PHASEB_PORTIN) &
(1<<ENC0_PHASEB_PIN)) == 0) )
        {
                EncoderState[0].position++;
        }
        else
        {
                EncoderState[0].position--;
        }
}
#endif
```