Control de motores CC mediante microprocesador PIC16F84

Por: Alejandro Alonso
Fecha: 28/12/2002

Objetivo

El proyecto consiste en un montaje hardware y un programa en ensamblador para controlar la velocidad de un motor de corriente continua mediante PWM y verificar la velocidad que desarrolla mediante encoder, mostrando dicha información en barrera de leds.

Para indicar al procesador la velocidad a la que se quiere que funcione el motor, existen dos switches cuya combinación de posiciones permite establecer cuatro velocidades diferentes. El programa establece una secuencia de pulsos PWM para cada una de las combinaciones de los switches. Existe un pulsador cuya función consiste en indicar al procesador que tome la combinación establecida en los switches y adapte la velocidad a dicha combinación. De esta manera actúa como cargador de datos.

En este montaje no existe bucle de realimentación. Es decir que el procesador se limita a enviar la secuencia de pulsos PWM al driver del motor, sin verificar que la velocidad alcanzada sea la deseada.

La combinación PWM establecida según la posición de los pulsadores es la siguiente:

SW1	SW2	Trama PWM
0	0	100010000
0	1	100100100
1	0	110110110
1	1	111111110

Como puede verse, la trama es de 9 bits. Esto se debe a que la técnica utilizada para el envío de los pulsos PWM al driver del motor es la rotación de bits con acarreo, con lo que el bit noveno es el acarreo

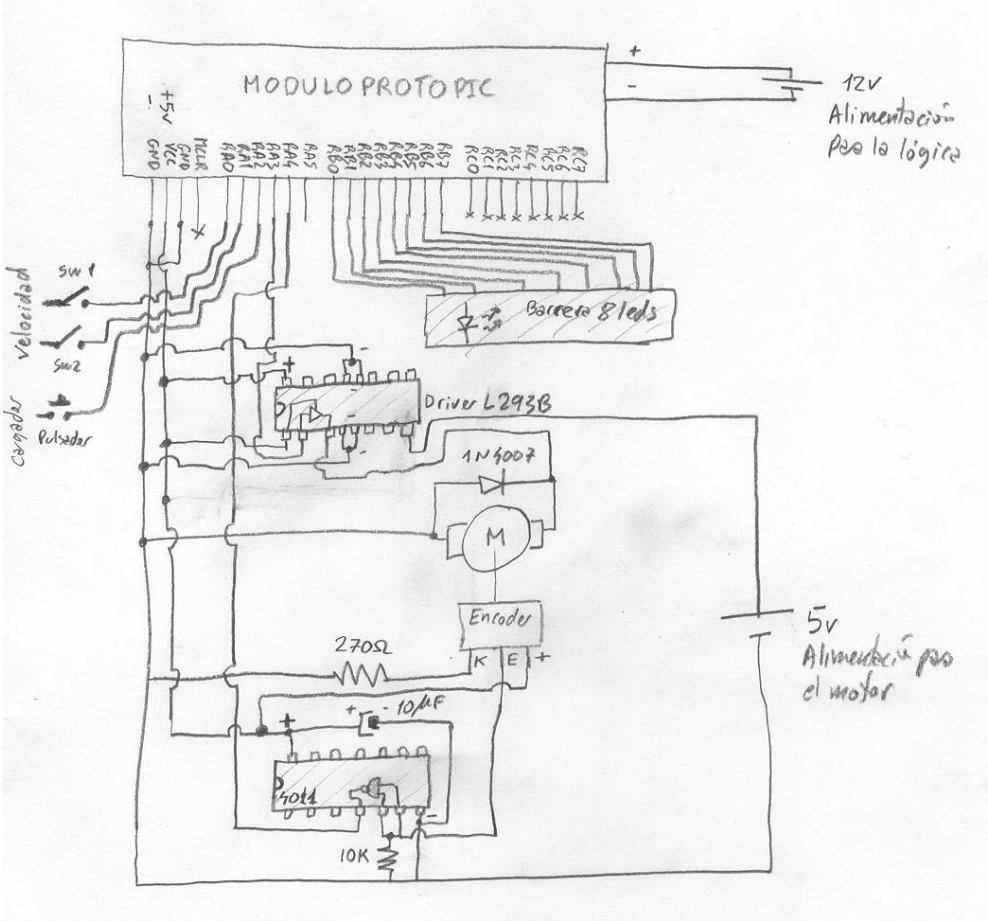
Por otra parte el procesador está programado para recibir pulsos del encoder y medir la velocidad del motor mostrándola en una barrera de 8 leds.

Insistimos en que no existe un enlace entre la medición de la velocidad y el envío de pulsos PWM. El programa trata ambas partes de forma independiente, aunque podrían haberse enlazado por programa. No obstante la idea de este montaje era experimentar sobre la medición de velocidades mediante encoders y el control de motores mediante PWM. La intención no era experimentar sobre los mecanismos de realimentación.

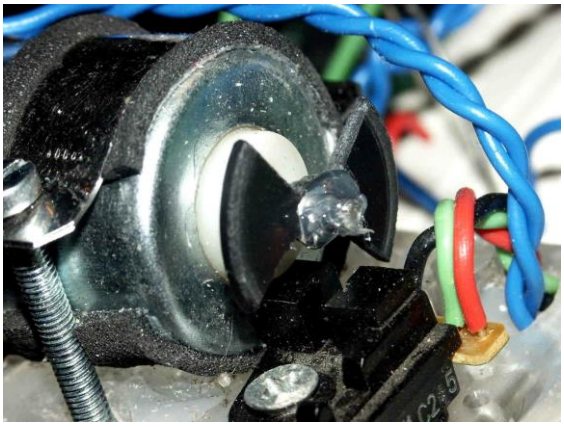
El montaje hardware

A continuación se muestra plano manual del montaje hardware. Básicamente está formado por 4 bloques:

- El módulo protopic, que no es más que un grabador de PICs.
- El bloque motor-encoder
- El driver del motor
- Una puerta NAND de tecnología CMOS para asegurar los niveles lógicos



La alimentación de la lógica y del motor está separada. Lo de la lógica proviene de la placa protopic que a su vez está alimentada externamente. Dicha placa estabiliza la tensión a 5v. Dicha tensión es utilizada para alimentar también al driver y al chip de puertas NAND. El motor está alimentado por otra fuente de corriente totalmente independiente de la anterior. Dicha corriente llega al motor a través del driver L293B

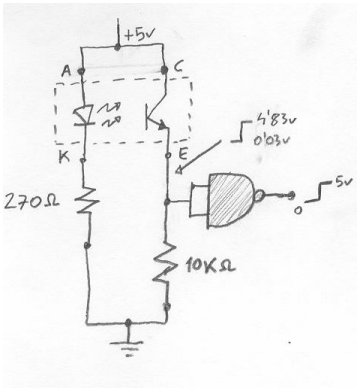


El motor tiene un diodo a la entrada de sus terminales para disipar la corriente de retorno producida al parar el motor. Esta corriente es producida debido a que el motor actúa como una bobina autoinducida.

El eje del motor está conectado a un encoder, formado por hélice de 4 estados (2 palas) y switch optoelectrónico.

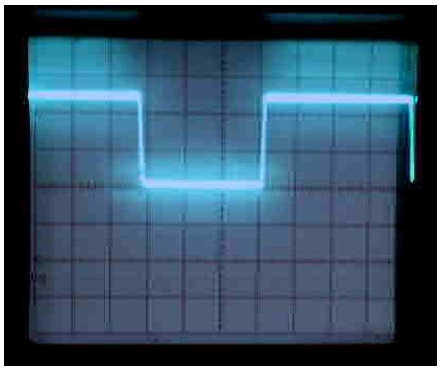
El corte del haz de luz que va del diodo al fotodiodo

es llevado a cabo por dicha hélice. Dichos cortes provocan pulsos de entre 1.8 v (corte) y 4.95 v (libre) con motor parado. Según aumentan las revoluciones, el nivel bajo se produce a voltajes más altos, de hasta 3 voltios. Además, la forma de onda producida no es puramente cuadrada como se puede ver en la imagen del osciloscopio (X: 2ms por cuadro; Y: 2V por cuadro. Cero calibrado en línea central).

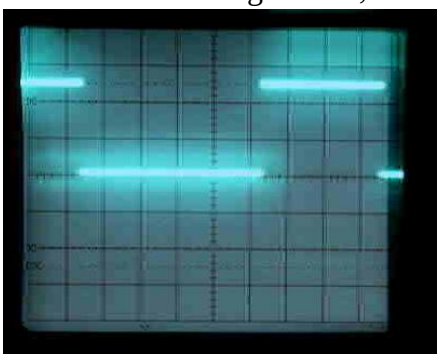


Estos valores son muy dependientes del modelo de switch fotoelectrónico utilizado y del aislamiento a interferencias de luz externa del mismo.

Para llevar estos voltajes a valores TTL válidos, la salida del fototransistor se ha llevado a una puerta NAND de tecnología CMOS (chip 4011), conectándola mediante resistencia de 10K a tierra para asegurar los niveles de entrada. A la izquierda se puede ver la imagen del osciloscopio a la entrada de la puerta NAND.

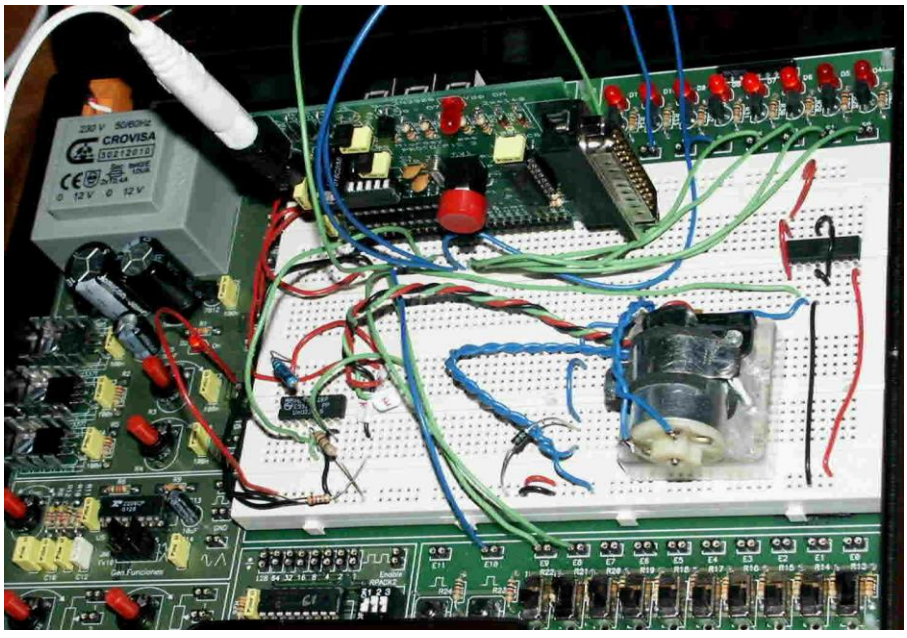


Al tener las entradas conectadas entre si, la puerta NAND se comporta como un inversor. Con esta configuración, los valores que se obtienen en la entrada de la puerta NAND son de 4,83v para el momento de no corte y 0,03v para el momento de corte incluso bajo interferencia de luz.



Esta puesta termina de dar la forma correcta a la señal como puede verse en la última imagen del osciloscopio.

El montaje queda como se muestra en la foto:



El Programa

El programa se ha realizado en ensamblador para el PIC 16F84. Básicamente funciona con dos funciones separadas pero solapadas:

- **Función de medición de velocidad**: Para ello se utiliza la variable "Velocidad", que se carga con unos. Tras esto se va ejecutando la subrutina "Retardo" para hacer una pausa temporal y se rota una posición a izquierdas la variable "Velocidad" metiendo el valor cero por la derecha (acarreo). El proceso se va repitiendo hasta que se produce una interrupción por desbordamiento del contador TMR0. Este contador está asociado a las señales del encoder. En dicha interrupción se vuelca la variable "Velocidad" al puerto RB en el que se encuentra conectado el bloque de 8 diodos, mostrándose una referencia a la velocidad. De esta manera, cuanto más rápido es el motor, antes se desbordará TMR0 y se producirá la interrupción, realizándose menos veces el bucle de tiempos que va desplazando bits a izquierdas, con lo que la barrera de leds mostrará más diodos encendidos.

- **Función de establecimiento de velocidad**: Simplemente lee la combinación de los switches 1 y 2 y si el pulsador está apretado, carga una trama de bits en la variable "PWM". El valor del bit 0 de PWM se irá volcando a la salida RA3 que está asociada al driver del motor. PWM irá rotando, con lo que RA3 ira mostrando en serie los bits de PWM, con lo que se consigue un auténtico control PWM del motor.

El código del programa es el siguiente:

```
; Programa para control de motores cc con encoder directo
; Por: Alejandro Alonso
; Fecha: 26/12/2002
; Función:
;
; Mide revoluciones y las muestra en una barrera de leds
; Selecciona cuatro posibles velocidades según la combinación
; de dos switches, cargando el patrón PWM adecuado en caso de
; presionar el pulsador.
; Asegura mediante trenes PWM el funcionamiento del motor a la
; velocidad seleccionada.


      list        p=16f84
      include     "P16F84a.INC"


      Contador    EQU   0x0C  ;Contador multiuso
      Contador2   EQU   0x0E  ;Contador multiuso
      Velocidad   EQU   0x0D  ;Cálculos de velocidad real
      PWM         EQU   0x0F  ;Cadena PWM a enviar al motor

      Backup      EQU   0xA0  ;Bits de backup para casos varios...
                              ;...Definidos como sigue:
      Acarreo     EQU   0


      ; Definiciones bits del registro RA

      Switch1     EQU   0     ;RA0 - Switches que definen patrón...
      Switch2     EQU   1     ;RA1 - de PWM (Velocidad)
```

```
      Pulsador    EQU   2     ;RA2 - Permiso de carga de patrón.
      Motor       EQU   3     ;RA3 - Salida PWM control velocidad
      Encoder     EQU   4     ;RA4 - Entrada señal encoder motor
      ;Atención, RA4 en modo salida trabaja en colector abierto


      org   0
      goto  INICIO
      org   4                 ; Vector de interrupción
      goto  INTERRUPT

INTERRUPT
      movf  Velocidad,W
      movwf PORTB             ;Muestra leds velocidad
      movlw b'11111111'       ;Inicializamos contador
      movwf Velocidad         ;de velocidad
      movlw b'11110000'       ;Inicializamos contador pulsos
      movwf TMR0
      bcf   INTCON,T0IF       ; quita flag
      RETFIE




; ------------------------------------------------------------------------
INICIO            ;Inicio del cuerpo del programa

      bsf   STATUS,RP0        ;Apunta a banco 1
      movlw b'11110111'       ;Establece puerta A como ENTRADA...
      movwf TRISA             ;...excepto RA3 (Motor)
      movlw b'00000000'       ;Establece puerta B como SALIDA (Leds)
      movwf TRISB             ;
      movlw b'00100000'       ;Configuración OPTION para TMR0
      movwf OPTION_REG
      bcf   STATUS,RP0        ;Apunta a banco 0

      movlw b'10100000'       ;Establece interrupciones
      movwf INTCON            ;para overflow TMR0

      clrf  PORTA
      clrf  PORTB

      movlw b'11111111'       ;Inicializamos contador
      movwf Velocidad         ;de velocidad

      movlw b'11110000'       ;reiniciamos contador pulsos
      movwf TMR0

      clrf  Backup            ;Limpiamos variable de backups

BUCLE ;Bucle principal del programa

      btfss PORTA,Pulsador    ;Vemos si hay que cargar patrón PWM
      Goto  Ciclo             ;No, continuamos ciclo
      ;Si, Chequeamos switches para asignar velocidad (cadena PWM)

      btfss PORTA,Switch1     ;Vemos valor Switch1
      goto  Sw_0x             ;Sw1=0
      btfss PORTA,Switch2     ;Sw1=1. Vemos valor Switch2
      goto  Sw_10             ;Sw1=1, Sw2=0
      goto  Sw_11             ;Sw1=1, Sw2=1
Sw_0x btfss PORTA,Switch2     ;Sw1=0. Vemos valor Switch2
      goto  Sw_00             ;Sw1=0, Sw2=0
      goto  Sw_01             ;Sw1=0, Sw2=1

Sw_00 movlw b'10001000'
```

```
      goto  Sw_Ok
Sw_01 movlw b'10010010'
      goto  Sw_Ok
Sw_10 movlw b'11011011'
      goto  Sw_Ok
Sw_11 movlw b'11111111'
      goto  Sw_Ok

Sw_Ok movwf PWM               ;Cargamos el valor seleccionado en PWM
      bcf   STATUS,C          ;Reiniciamos valores
      bcf   Backup,Acarreo    ;Reiniciamos valores


Ciclo Movlw d'230'            ;
      movwf Contador          ;...se carga en "contador"
      call  Retardo
      bcf   STATUS,C          ;Ponemos a 0 acarreo
      rrf   Velocidad,F


      GOTO BUCLE



; ------------------------------------------------------------------------


; Subrutinas


Retardo     ;Provoca un retardo según el valor de "Contador"
      btfss Backup,Acarreo    ;Vemos cual era el valor anterior de acarreo
      goto  C_Cero            ;..y lo ponemos de nuevo como acarreo
      bsf   STATUS,C
      goto  C_Ok
C_Cero      bcf   STATUS,C
C_Ok  NOP
Bucle1      movlw 255         ;Inicialización bucle interno
      movwf Contador2
Bucle2      btfss PWM,0       ; Chequea bit 0 de cadena PWM
      goto  EsCero            ; Si es cero pone cero en Salida control motor
      bsf   PORTA,Motor       ; Si es uno pone cero en Salida control motor
      goto  Sigue
EsCero      bcf   PORTA,Motor
Sigue rrf   PWM,F             ; Rota PWM
      btfss STATUS,C          ;Vemos cual es el valor de acarreo y hacemos
backup
      goto  Cc_Cero           ;..y lo ponemos de nuevo como acarreo
      bsf   Backup,Acarreo
      goto  Cc_Ok
Cc_Cero     bcf   Backup,Acarreo
Cc_Ok NOP
      decfsz      Contador2,F ;Decrementar contador bucle externo
      goto  Bucle2            ;y repetir bucle interno hasta fin
      decfsz      Contador,F  ;Decrementar contador bucle externo
      goto  Bucle1            ;y repetir bucle externo hasta fin

      RETURN



Fin
      END
```