BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA
MAESTRÍA EN INGENIERÍA ELECTRÓNICA, OPCIÓN
INSTRUMENTACIÓN ELECTRÓNICA

Sistemas Programables

Practica 2

Control de Motor a pasos

Presentan:

Ing. Luis Efraín López García
Ing. Alejandro Sánchez Mendoza
Lic. Luis Carlos López Guerra

Puebla, Pue., Marzo 2021

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Motor paso a paso, es un motor de corriente continua cuya característica más destacada es que su movimiento se produce mediante pequeñas traslaciones o pasos. Tiene la posibilidad de rotar un número determinado de grados y quedarse enclavados en una posición, ejerciendo fuerza contraria a la acción que intente moverlos, o bien quedar totalmente libres. A diferencia de un motor convencional de corriente continua que no poseen esta cualidad, pues al aplicar un voltaje giran de manera continua y al dejar de aplicar dicho voltaje dejarán de girar, pero este motor no es capaz de girar un número de grados específicos ni quedarse enclavados en una posición

El control de este tipo de motor suele realizarse en cadena abierta pues brinda la ventaja de no tener que utilizar ningún tipo de retroalimentación para llevar el motor a una posición determinada. El movimiento se produce gracias a la excitación consecutiva de los diferentes devanados que componen las fases de estator; estas fases atraen al rotor, el cual a su vez está formado por múltiples polos, produciendo el movimiento relativo del rotor respecto al estator, Fig. 1.1.

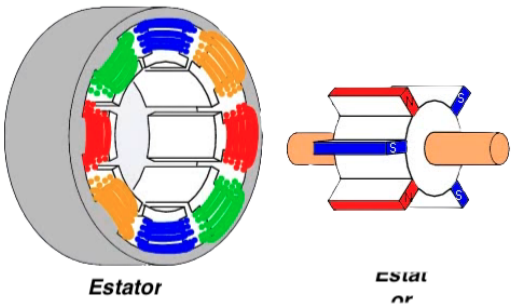

Figura 1.1: Motor paso a paso con estator y rotor.

Dependiendo de la construcción interna del motor, los grados que realiza por cada paso paso pueden ser desde 90 hasta pequeños pasos de 1,8. Cuanto más pequeño sea el paso más precisón poseerá el motor pero más despacio girará a una misma frecuencia de fncionamiento. Por lo cual, los motores paso a paso se pueden clasificar en función de sus características constructivas o en función del tipo de alimentación de fases del estator.

El motor $28BYJ-48$ es un motor de cuatro fases, pues tiene cuatro pares de bobinas a 180 entre si, Fig 1.1 La representación en diagrama de este motor se muestra en la Fig.1.2, donde el circulo representa el rotor, en la parte superior izquierda se representan dos pares de bobinas e inmediatamente abajo otro par; de manera análoga se tiene otros dos pares de bobinas en el lado inferior derecho. Los extremos de cada bobina se unen para formar un nodo comun, es por esto que se llaman unipolar de cinco hilos.

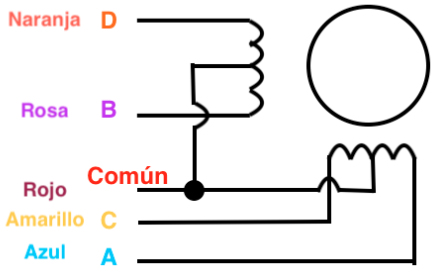

Figura 1.2: Diagrama de motor $28BYJ-48$.

Este motor requiere una fuente de voltaje DC de $5V$. Además tiene un sistema de reducción mecánica de $1:64$, es decir, en eje del rotor se encuentra un engrane, el cual se conecta a un sistema de engranajes para reducción de velocidad pero con un aumento de torque; la reducción significa que el eje del rotor deberá dar 64 vueltas para que el eje externo de un paso.

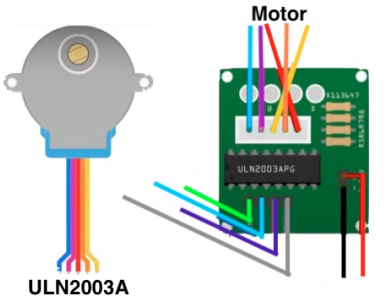

Figura 1.3: Motor $28BYJ-48$ con driver y conexiones.

El consumo de cada bobina de este motor es de $40mA$ y puede trabajar a una frecuencia máxima de $100MHz$, es decir la aplicación de pulsos a cada bobina deberá ser mayor a $10ms$. Para el uso de este motor se emplea el driver $ULN2003A$, el cual ya incluye un conector para el motor de acuerdo con las especificaciones de este último Fig. 1.3.
El driver $ULNN2003A$ permite controlar el motor $28BYJ-48$ mediante los pines de entrada $In1$, $In2$, $In3$ y $In4$, los cuales permitirán suministrar una corriente más alta y a las salidas A, B, C y D un control lógico del motor; este driver tiene además 4 LEDS para indicar los estados lógicos aplicados. Cabe destacar que este driver puede suministrar una corriente de salida máxima de $500mA$ lo cual es suficiente para el motor.

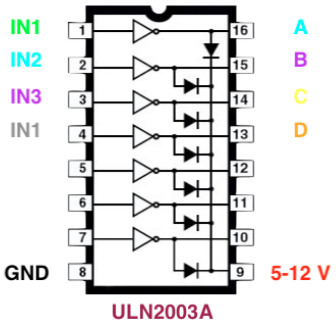

Figura 1.4: Driver $ULN2003A$.

Para lograr que el eje del motor gire se debe aplicar una secuencia especifica para hacer girar el rotor encendiendo las bobinas Fig. 1.5.

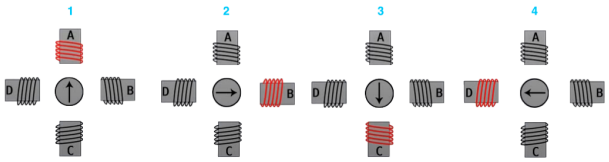

Figura 1.5: Secuencia en bobinas.

Es decir, se tiene cuatro cables con las denominaciones A, B, C y D, que deberán corresponder a las cuatro fases del motor por lo cual para que se logre un paso se debe enviar un 1 a la bobina A y a las demas un 0 , para un segundo paso se debe enviar un 1 a la bobina B y a las demas un 0, ara un tercer paso se debe enviar un 1 a la bobina C a las demas un 0 y para el cuarto paso se debe enviar un 1 a la bobina D y a las demas un 0, Fig.1.6.

Paso	A	B	C	D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Paso Completo Simple (Wave Drive)

Figura 1.6: Secuencia en bits.

1.1. Objetivo General

Haciendo uso de retardos realizar el control de giro de un motor a pasos bipolar, con una frecuencia establecida de 110ms de retardo entre cada excitación en las bobinas.

1.2. Objetivos Específicos

1. Identificar las características del PIC18F4550.
2. Manejar el software MPLAB X para crear un proyecto y simular.
3. Analizar el proceso para generar retardos de tiempo, haciendo que el procesador ejecute instrucciones.
4. Analizar el funcionamiento de los Motores a pasos.
5. Realizar un programa en lenguaje ensamblador para controlar el movimiento de giro de un motor a pasos, utilizando el PIC18F4550.
6. Determinar la velocidad máxima a la que responde el motor a pasos.
7. Realizar la simulación en el software Proteus del circuito completo.
8. Programar el microcontrolador (Pickit2, ICD2, Master-Pro, etc.) y probar su correcto funcionamiento.

1.3. Conocimientos Previos

- Conocimientos de la arquitectura del PIC18F4550 y de su conjunto de instrucciones.

- Conocimientos del funcionamiento de los Motores a pasos.

1.4. Material y Equipo Requerido

- 1 PIC 18F4550.
- 1 Puente H (L293, L298 o ULN2003).
- 1 Resistencia de 10KΩ
- 1 Pulsador (Push button).
- 1 Tablilla de experimentos.
- 1 Fuente de 5 V.
- 1 Programador para PIC.
- 1 Computadora.
- Software MPLAB, Proteus y software para el programador.

Capítulo 2

Desarrollo

2.1. Diagrama de Flujo

Una vez que se identificó de manera general las características del PIC 18F4550 se procede a realizar un diagrama de flujo, Figura 2.1, para desglosar los requerimientos necesarios y facilitar la implementación en código. La primera sección del diagrama corresponde a la configuración del , en ella se establecen características como el uso del oscilador interno de $4MHz$, configuración de los puertos B y D como entrada y salida correspondientemente y la activación de las resistencias de Pull up, entre otras.

La siguiente sección corresponde al algoritmo de funcionamiento como tal, se tiene como condición inicial al motor apagado, posteriormente se pregunta por el estado del puerto B para conocer las entradas y a raíz del estado de B decidir el sentido de giro del motor, este proceso se repite de manera cíclica hasta que el sistema sea apagado.

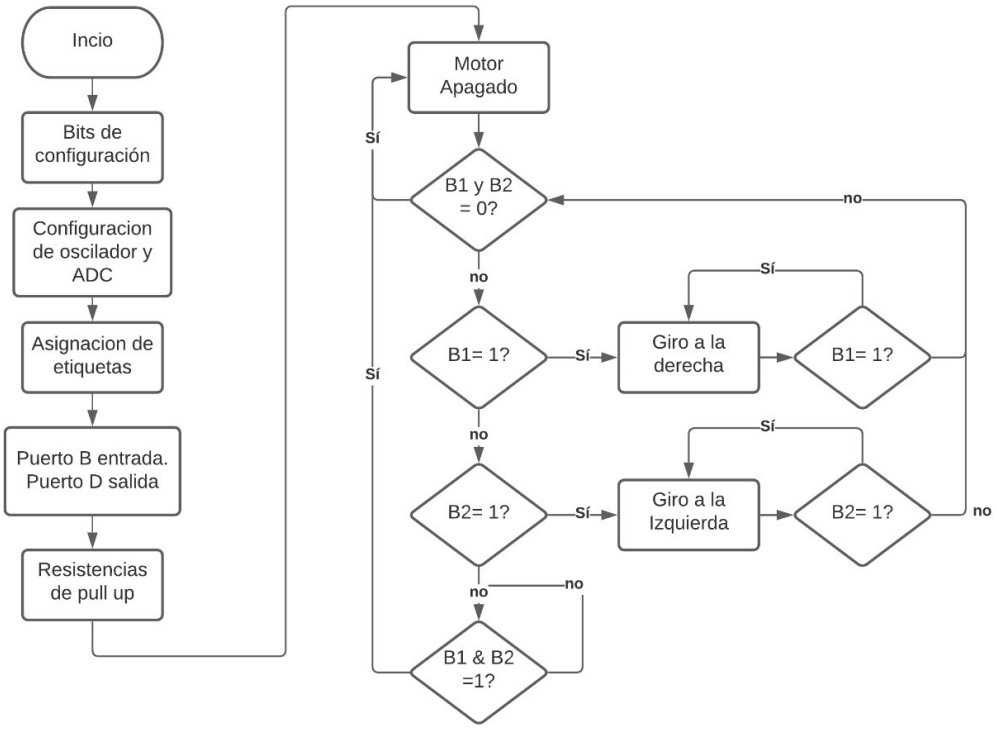

Figura 2.1: Diagrama de Flujo Secuencia de Giro.

Para generar el giro del motor como se presentó en la introducción se debe inyectar una secuencia en las bobinas del motor, en la figura 2.2 se muestran

las secuencias de salida para los giros tanto en sentido horario como en sentido contrario y los valores del puerto D que corresponden a dicha secuencia.

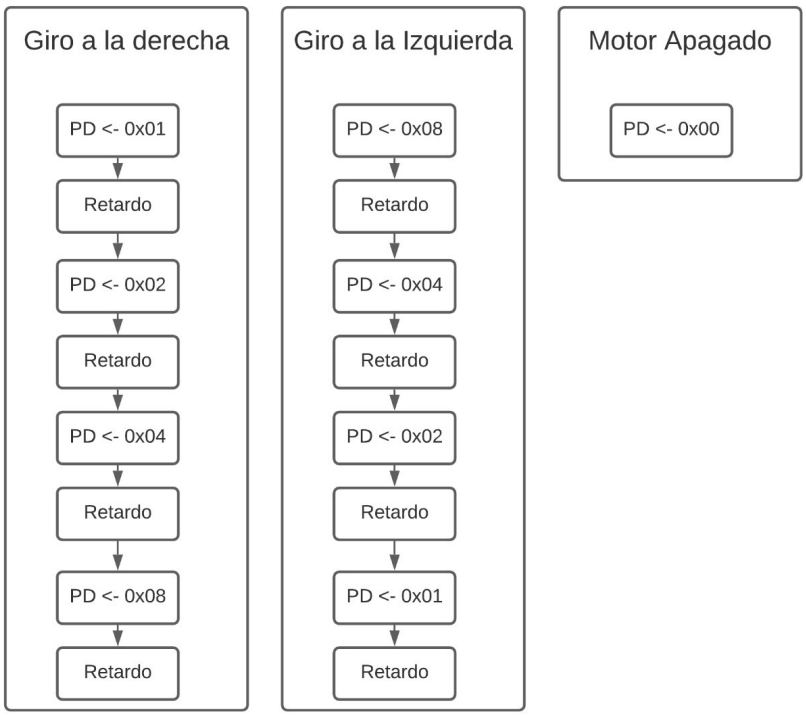

Figura 2.2: Diagrama de Flujo.

En las secuencias de giro se utilizan retardos entre cada salida del puerto D, en la figura 2.3 se muestra el diagrama de flujo correspondiente a estos retardos, en esta práctica el tiempo de retardo solicitado fue de $110ms$, para lograr este tiempo fue necesario anidar los ciclos resultando en un retardo de exactamente 109,987ms.

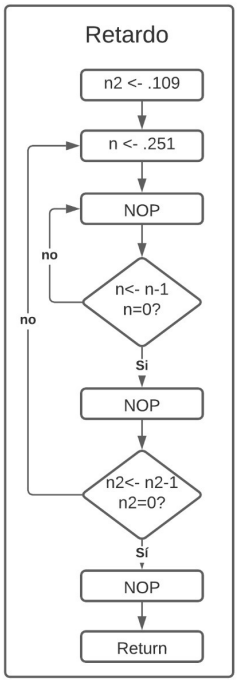

Figura 2.3: Diagrama de Flujo Retardo.

2.2. Código

Para implementar el diagrama de flujo se utilizó MPLAB X V5.20, el cual es un IDE gratuito que proporciona Microchip Technology Inc. Figura 2.4.

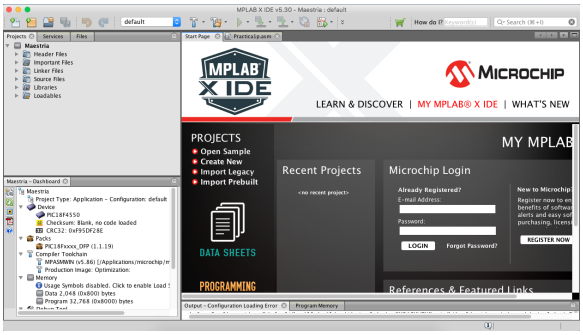

Figura 2.4: MPLABX IDE.

En este IDE se creó un proyecto en File ≫ New Project, en esta sección emergente se escogió Microchip Embedded ≫ Standalone Project donde se seleccionó Family y Device. Posterior a este paso en el apartado Select

Tool (Optional) no se seleccionó ninguna opción. Paso siguiente se seleccionó mpasm en el apartado Select Compiler y por último paso se escribió el nombre del proyecto y ubicación del mismo. En esta implementación se debe seleccionar el PIC a programar y la librería a utilizar, lo cual se realizó mediante directivas. Posteriormente se escribió los bits de configuración, Figura 2.5, lo cual se realizó de acuerdo al diagrama de flujo.

```
; TODO INSERT INCLUDE CODE HERE
    List    P=18F4550           ;Microcontrolador a utilizar
    #include <P18F4550.inc>     ;Definiciones de constantes
;*******************************************************************************
        ; TODO INSERT CONFIG CODE HERE USING CONFIG BITS GENERATOR
;**********   Palabra de conficuración  ***************************************
        CONFIG FOSC = INTOSCIO_EC       ;INTOSC_XT              ;Internal oscillator
        ;CONFIG CPUDIV = OSC1_PLL2      ;[Primary Oscillator Src: /1][96 MHz PLL Src: /2]
        CONFIG PWRT= ON                 ; Power-up Timer Enable bit
        CONFIG BOR= OFF                 ; Brown-out Reset disabled in hardware and software
        CONFIG WDT= OFF                 ; WDT disabled
        CONFIG MCLRE= ON                ; MCLR pin enabled
        CONFIG PBADEN= OFF              ; PORTB<4:0> pins are configured as digital I/O
        CONFIG LVP= OFF                 ; Single-Supply ICSP disabled
        CONFIG DEBUG = OFF              ; Background debugger disabled
        CONFIG XINST = OFF              ; Extended Instruction disabled
```

Figura 2.5: Código de configuración.

Después de escribir los bits de configuración se creó las variables n y $n2$ mediante etiqueta, lo cual corresponde a asignar un espacio de memoria RAM, Figura 2.6.

```
;*******************************************************************************
; TODO PLACE VARIABLE DEFINITIONS GO HERE
n       equ 0x00        ; --> RAM[0]
n2      equ 0x01        ; --> RAM[1]
;*******************************************************************************
```

Figura 2.6: Código de definición de variables.

De acuerdo con el diagrama de flujo se tiene que configurar el oscilador mediante File OSCCON, de igual manera se configuro el PB mediante File TRISB como entrada, de manera análoga el PD mediante File TRISD como salida. Como último paso en la configuración se habilitó PU mediante File INTCON2, Figura 2.7.

```
    org     0x0000      ;la inst. que sigue
    movlw   0x62        ;W <- 0x62
    movwf   OSCCON,A    ;Osc Interno, Fosc=4 MHz, A es el banco de acceso
    movlw   0x0F        ;
    movwf   ADCON1      ; Todos los pines digitales
    setf  TRISB,0       ;PB entrada
    clrf  PORTD,A       ;PD Inicializado en 0's/Limpiar PORTD
    clrf  TRISD         ;PD salida
    bcf   INTCON2,RBPU  ;Habilitar resistencias de Pull up
```

Figura 2.7: Código de configuración puertos.

Continuando con el diagrama de flujo, se establece el primer estado del motor que corresponde al estado de apagado, Figura 2.8.

```
33
34  APAGADO
35      movlw   0x00
36      movwf   PORTD,A
37      movlw   0xFC
38      cpfseq  PORTB,A
39      bra     EVALUAR_A
40      bra     APAGADO
```

Figura 2.8: Código de ciclo.

Después se inicia con la siguiente parte de los estados del moto de acuerdo con el diagrama de flujo, para esto se usan las etiquetas EVALUAR, que consta de cuatro partes. En la primera de estas se pregunta si los dos están en bajo, en las siguientes dos se pregunta si alguna está en alto para que se realice el giro a la derecha o a la izquierda y en la ultima se vuelve a preguntar si alguna está en alto para regresar al estado apagado, Figura 2.9.

```
42  EVALUAR_A
43      movlw   0xFC
44      cpfseq  PORTB,A
45      bra     EVALUAR_B
46      bra     APAGADO
47
48  EVALUAR_B
49      movlw   0xFE
50      cpfseq  PORTB,A
51      bra     EVALUAR_C
52      bra     DERECHA
53
54  EVALUAR_C
55      movlw   0xFD
56      cpfseq  PORTB,A
57      bra     EVALUAR_D
58      bra     IZQUIERDA
59
60  EVALUAR_D
61      movlw   0xFF
62      cpfseq  PORTB,A
63      bra     EVALUAR_D
64      bra     APAGADO
65
```

Figura 2.9: Código estado del motor.

Siguiendo el diagrama de flujo, al obtener el primer estado del motor en alto se avanza a la parte del código en la cual se dan las instrucciones de giro. Esto se realiza asignando al puerto de diferentes valores de asignación al PD y entre cada una de estas se ejecuta un retardo, Figura 2.10. De manera análoga se asigna la secuencia de asignación al PD junto con los retardos, pero en esta ocasión se realiza de manera inversa, Figura 2.11.

Por último, en la parte final del código se escribe las lineas de código para los retardos, en la cual se hace uso de las variables asignadas en la memoria RAM. De igual manera para los ciclos.

```
66  IZQUIERDA
67      movlw   0x01  ;
68      movwf   PORTD,A
69      call    RETARDO2
70      movlw   0x02  ;
71      movwf   PORTD,A
72      call    RETARDO2
73      movlw   0x04  ;
74      movwf   PORTD,A
75      call    RETARDO2
76      movlw   0x08  ;
77      movwf   PORTD,A
78      call    RETARDO2
79      movlw   0xFD
80      cpfseq  PORTB,A
81      bra     EVALUAR_A
82      bra     IZQUIERDA
83
```

Figura 2.10: Código de giro a la izquierda.

```
84  DERECHA
85      movlw   0x08  ;
86      movwf   PORTD,A
87      call    RETARDO2
88      movlw   0x04  ;
89      movwf   PORTD,A
90      call    RETARDO2
91      movlw   0x02  ;
92      movwf   PORTD,A
93      call    RETARDO2
94      movlw   0x01  ;
95      movwf   PORTD,A
96      call    RETARDO2
97      movlw   0xFE
98      cpfseq  PORTB,A
99      bra     EVALUAR_A
100     bra     DERECHA     ;0x000C
```

Figura 2.11: Código de giro a la derecha.

```
101
102 RETARDO2
103     movlw   .109        ; 100 decimal
104     movwf   n2          ; n <- 100
105
106 RETARDO                 ;Rutina de Retardo de 100 us
107 ;                               (n-1)
108         ;  No CI = 2+1+1 + (1+1+2) * 99 + 1+2+2 = 405
109     movlw   .251        ; 100 decimal
110     movwf   n           ; n <- 100
111 CICLO2
112     nop                 ; consume un CI
113     decfsz  n,1         ;n <- n -1 y si n es cero salta la siguiente inst
114     bra     CICLO2
115 CICLO3
116     nop                 ; consume un CI
117     decfsz  n2,1        ;n <- n -1 y si n es cero salta la siguiente inst
118     bra     RETARDO
119     nop
120
121     return
122     end
```

Figura 2.12: Código de retardo y ciclo.

2.3. Cálculos

En esta practica se utilizaran retardos para generar una velocidad de giro en el motor a pasos. La velocidad asignada es de 110 ms segundo, para ello se tiene una tolerancia de 1ms de error. Ya que se requiere ser lo mas preciso posible es necesario desarrollar una pequeña formula que contemple cada

instrucción utilizada en el anidamiento de retardos para llegar al objetivo. A continuación se muestra la formula desarrollada.

$$Retardo = (2 + 1 + 1) + ((((1 + 1 + 2) * n) + 1 + 2 + 2) * n_2) + 1 + 2 \quad (2.1)$$

Teniendo esta formula se realizo la sustitución de n y n_2 por los valores de 251 y 109 respectivamente. Como se muestra a continuación.

$$(2+1+1)+((((1+1+2)*251)+1+2+2)*109)+1+2 = 109{,}987ms \quad (2.2)$$

Con el resultado obtenido obtenido en la ecuación 2.2 se consigue una aproximación muy buena al valor necesario, es decir se cuenta con una exactitud del 99.9881 % con un error de 0.013 ms con respecto al objetivo. Por utlimo en la figura 2.13

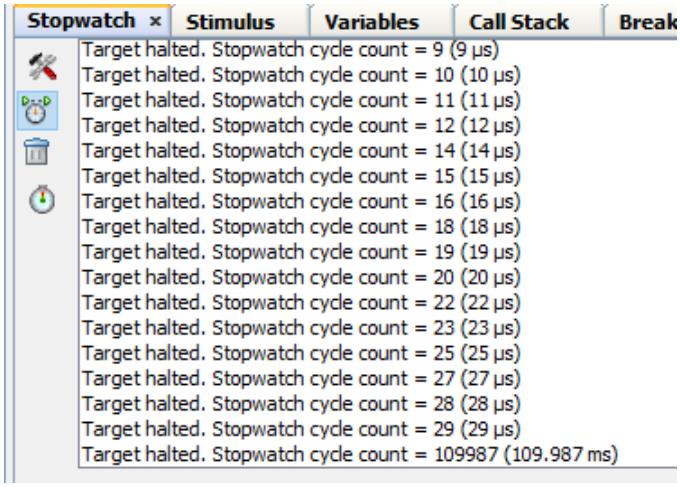

Figura 2.13: Tiempo generado por retardo.

2.4. Simulación

Para realizar una verificación de que el sistema se encuentra funcionando de forma óptima antes de llevarlo a una aplicación física es necesario realizar una simulación, para ello se utiliza el software Proteus, el cual permite implementar un sistema haciendo uso del PIC18F4550 utilizando el archivo ".hex"generado por MPLABX. A continuación en la figura 2.14 se muestra la simulación generada para este caso.

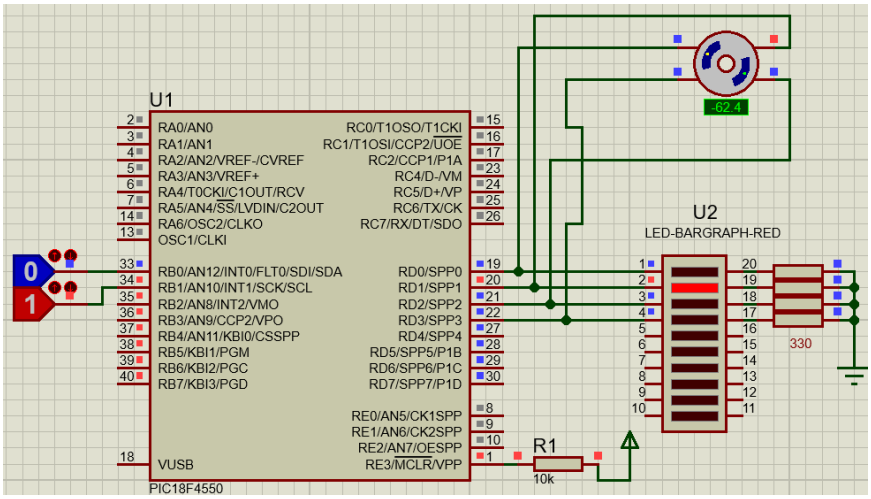

Figura 2.14: Simulación en Proteus.

En esta simulación se puede observar que a modo de entrada se utilizaron estados lógicos para facilitar la simulación. En la parte de la salida se utilizo una led bargraph y un motor a pasos. En esta simulación se puede observar que se encuentra funcionando de forma óptima el sistema diseñado.

2.5. Implementación

Para realizar la implementación física se utilizo un PIC18F4550 con 40 pines tipo DIP, una led bargraph MV57164 de color rojo, cuatro resistencias de 330Ω, un dipswitch de 8 vías, un motor a pasos 28by-48, un driver ULN2003A ,jumpers y alambres para realizar las conexiones dentro del protoboard, para la alimentación del sistema se utilizo una fuente regulable establecida en 5v de salida. En figura 2.15 se muestra el sistema físico ensamblado para esta practica.

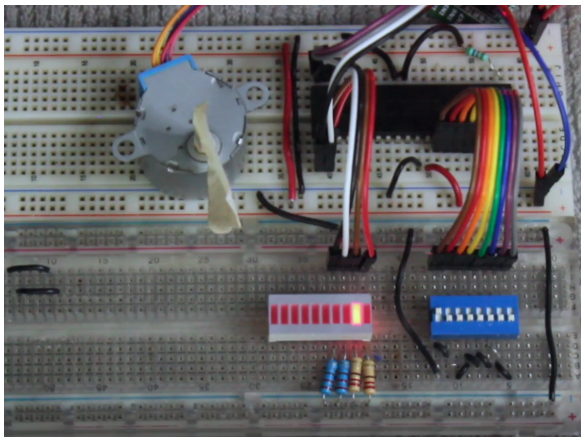

Figura 2.15: Implementación Física.

Para realizar la programación del PIC18F4550 se utilizo un programador PIC-600 de la marca Steren, el cual cuenta con dos tipos de conexiones, utilizando un zócalo zif de 40 pines o utilizando la conexión ICSP. En la figura 2.16 se puede observar el programador utilizado.

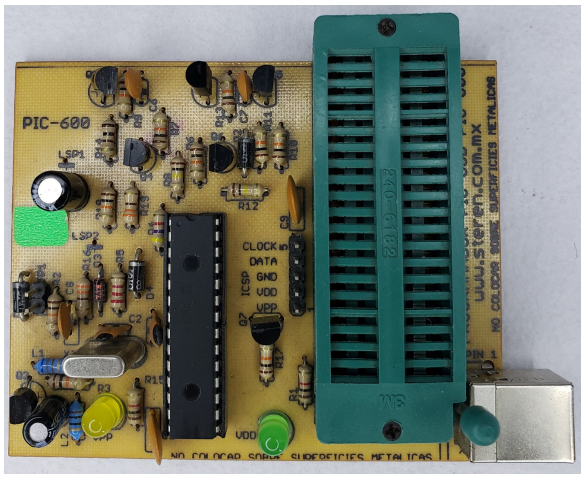

Figura 2.16: Programador PIC-600.

Capítulo 3

Resultados y Conclusiones

3.1. Resultados

En esta practica se buscaba controlar el funcionamiento de un motor a pasos, es por ello que para verificar que el sistema funciona de forma correcta se realizo un vídeo en donde se puede ver de forma clara el funcionamiento del sistema utilizando las convenciones vistas en el cuadro 3.1. El video puede ser encontrado en el siguiente link: https://youtu.be/iAcDcGMRCqI

P1	p2	Salida
OFF	OFF	Parado
OFF	ON	Giro Horario
ON	OFF	Giro Antihorario
ON	ON	Parado

Cuadro 3.1: Movimiento del motor utilizando dos interruptores

3.2. Conclusiones

En el presente trabajo se mostró el proceso de desarrollo de una práctica haciendo uso de un microcontrolador PIC18F4550 tomando en cuenta procesos de conceptualización como lo es el desarrollo de un diagrama de flujo y de los cálculos para determinar la forma de generar los retardos para el control de velocidad del motor, también se realizaron simulaciones para verificar el funcionamiento del diseño previo a su implementación.

Un punto muy importante es entender la forma en la que se pueden generar retardos dentro de este microcontrolador de 8-bits, ya que depende de la frecuencia utilizada en el oscilador y la forma del algoritmo relacionado con estos retardos, ya que dentro de ellos cada instrucción cuenta en el tiempo final.

Una vez implementado el circuito diseñado y verificado su funcionamiento es posible decir que el resultado es el esperado ya que lo obtenido durante las pruebas cumple con lo arrojado por la simulación, es importante resaltar lo valiosa que es la etapa de diseño previa a la implementación de cualquier sistema por más sencillo que este parezca, dado que durante este proceso es posible encontrar potenciales fallas y evitar un desperdicio de recursos como tiempo o dinero.

Anexo A

Código en Ensamblador

```
; TODO INSERT INCLUDE CODE HERE
    List      P=18F4550           ;Microcontrolador a utilizar
    #include <P18F4550.inc>      ;Definiciones de constantes
;*******************************************************************************
        ; TODO INSERT CONFIG CODE HERE USING CONFIG BITS GENERATOR
;**********   Palabra de conficuraci n          ****************************************
        CONFIG FOSC = INTOSCIO_EC ;INTOSC_XT            ;Internal oscillator
        ;CONFIG CPUDIV = OSC1_PLL2       ;[Primary Oscillator Src: /1][96 MHz PLL Src: /2]
        CONFIG PWRT= ON                  ; Power-up Timer Enable bit
        CONFIG BOR= OFF                  ; Brown-out Reset disabled in hardware and software
        CONFIG WDT= OFF                  ; WDT disabled
        CONFIG MCLRE= ON                 ; MCLR pin enabled
        CONFIG PBADEN= OFF               ; PORTB<4:0> pins are configured as digital I/O
        CONFIG LVP= OFF                  ; Single-Supply ICSP disabled
        CONFIG DEBUG = OFF               ; Background debugger disabled
        CONFIG XINST = OFF               ; Extended Instruction disabled
;*******************************************************************************
; TODO PLACE VARIABLE DEFINITIONS GO HERE
n       equ 0x00          ; --> RAM[0]
n2      equ 0x01          ; --> RAM[1]
;*******************************************************************************

    org     0x0000  ;la inst. que sigue
    movlw   0x62    ;W <- 0x62
    movwf   OSCCON,A;Osc Interno, Fosc=4 MHz, A es el banco de acceso
    movlw   0x0F    ;
    movwf   ADCON1  ; Todos los pines digitales
    setf  TRISB,0       ;PB entrada
    clrf  PORTD,A   ;PD Inicializado en 0's/Limpiar PORTD
    clrf  TRISD     ;PD salida
    bcf   INTCON2,RBPU  ;Habilitar resistencias de Pull up


APAGADO
    movlw   0x00
    movwf   PORTD,A
    movlw   0xFC
    cpfseq  PORTB,A
    bra     EVALUAR_A
    bra     APAGADO
```

```
EVALUAR_A
    movlw   0xFC
    cpfseq  PORTB,A
    bra     EVALUAR_B
    bra     APAGADO

EVALUAR_B
    movlw   0xFE
    cpfseq  PORTB,A
    bra     EVALUAR_C
    bra     DERECHA

EVALUAR_C
    movlw   0xFD
    cpfseq  PORTB,A
    bra     EVALUAR_D
    bra     IZQUIERDA

EVALUAR_D
    movlw   0xFF
    cpfseq  PORTB,A
    bra     EVALUAR_D
    bra     APAGADO

IZQUIERDA
    movlw   0x01    ;
    movwf   PORTD,A
    call    RETARDO2
    movlw   0x02    ;
    movwf   PORTD,A
    call    RETARDO2
    movlw   0x04    ;
    movwf   PORTD,A
    call    RETARDO2
    movlw   0x08    ;
    movwf   PORTD,A
    call    RETARDO2
    movlw   0xFD
    cpfseq  PORTB,A
    bra     EVALUAR_A
    bra     IZQUIERDA

DERECHA
    movlw   0x08    ;
    movwf   PORTD,A
    call    RETARDO2
    movlw   0x04    ;
    movwf   PORTD,A
    call    RETARDO2
    movlw   0x02    ;
    movwf   PORTD,A
    call    RETARDO2
    movlw   0x01    ;
    movwf   PORTD,A
    call    RETARDO2
    movlw   0xFE
    cpfseq  PORTB,A
    bra     EVALUAR_A
    bra     DERECHA     ;0x000C

RETARDO2
    movlw   .109      ; 100 decimal
```

```
    movwf   n2          ; n <- 100

RETARDO     ;Rutina de Retardo de 100 us
    ;                           (n-1)
    ;  No CI = 2+1+1 + (1+1+2) * 99 + 1+2+2 = 405
    movlw   .251        ; 100 decimal
    movwf   n           ; n <- 100
CICLO2
    nop                 ; consume un CI
    decfsz  n,1         ;n <- n -1 y si n es cero salta la siguiente inst
    bra     CICLO2
CICLO3
    nop                 ; consume un CI
    decfsz  n2,1                ;n <- n -1 y si n es cero salta la siguiente inst
    bra     RETARDO
    nop

    return
    end
```