**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**fiecFACULTAD DE INGENIERIA EN**

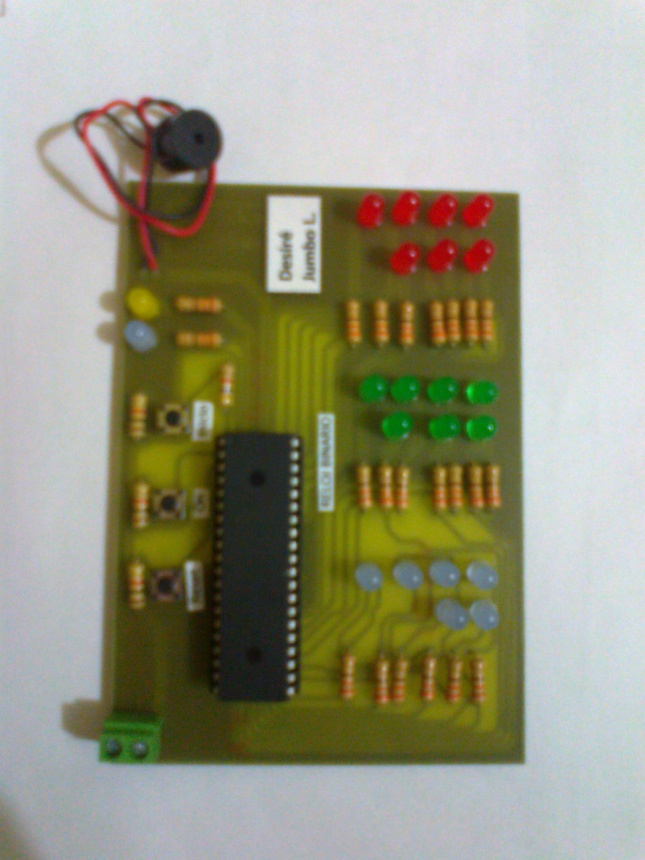
**EN ELECTRICIDAD Y COMPUTACIÓN**

LABORATORIO DE MICROCONTROLADORES

**Proyecto 7**

**Reloj capaz de generar 5 tonos audibles cada 5 segundos**

**Hardware y Software**





Estudiante:

Desiré Jumbo Lucas

Profesor:

Carlos Valdiviezo Armendariz

Término:

II Término 2011

1. **DESCRIPCIÓN DEL PROYECTO**

El proyecto que se llevará a cabo consiste en la implementación de un reloj binario que muestra los segundos, minutos y horas mediante el encendido y apagado de leds siguiendo la secuencia de los números binarios (distribuidos en seis filas), éste proyecto tiene la característica de que cada 5 segundos contados del reloj, emite 5 tonos audibles, además consta de tres botones uno para encender el circuito, otro para dar marcha al reloj y uno de reset para volver a iniciar el PIC.

1. **DIAGRAMA DE BLOQUES**



1. **DIAGRAMA DE FLUJO**



1. **Descripción**

* Inicialización del programa
* Elegir PIC16F887 y configurar los parámetros y las directivas correspondientes.
* Declaración de Variables e inicialización de las mismas.
* Configuración de puertos B(para segundos) , C( para minutos) y D (para horas)
* Configuración de PORTA,0, PORTA,1 y PORTA,2 como entradas.
* Preguntar si se presionó el botón en RAO y si se soltó.
* Llamar al retardo de 1 segundo
* Preguntar por start.
* Se muestran los segundos. Se usan las funciones BCD\_a\_DEC para poder mostrar en los puertos, esto de da en cada uno de los casos.
* Se vuelve a preguntar por el botón el botón en RA1 en cada segundo que se muestra en los leds.
* Se generan los sonidos luego de cada 5 segundos.
* Se muestran los minutos.
* Se vuelve a preguntar por el botón en RAO y el botón en RA1 (esto se lo vuelve a realizar para no provocar el efecto anti rebote).
* Se incrementan las unidades de segundos, decenas de segundos, unidades de minutos y decenas de minutos.

1. **PROGRAMA FUENTE**

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Proyecto # 7

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; NOMBRE: prueba1.asm

; FECHA PRESENTACION: 21/12/2011

; VERSION: 3.00

; PROGRAMADOR: Desiré Jumbo Lucas

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; DESCRIPCION:

;Implementar un reloj binario que contenga un botón que encienda el reloj

;se usará los puertos B,C,D para mostrar los leds encendidos

;representando las segundos, minutos y horas. Cuando llega a contar 5 segundos

;comienza a sonar 5 tonos de igual magnitud. Después de eso sigue contando.

; Use un reloj de 4MHz.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;DIRECTIVAS:

LIST p=16F887 ;Tipo de microcontrolador

INCLUDE P16F887.INC ;Define los SFRs y bits del pic16F887

\_\_CONFIG \_CONFIG1, \_LVP\_OFF & \_FCMEN\_OFF & \_IESO\_OFF & \_BOR\_OFF & \_CPD\_OFF & \_CP\_OFF & \_MCLRE\_ON & \_PWRTE\_ON & \_WDT\_OFF & \_INTRC\_OSC\_NOCLKOUT

\_\_CONFIG \_CONFIG2, \_WRT\_OFF & \_BOR21V ;Setea parámetros de configuración

errorlevel -302, -306 ;Deshabilita mensajes de

;advertencia por cambio

;bancos

;variables que se utilizan

CBLOCK 0x20

uni\_seg

dec\_seg

tmp\_s

tmp\_min

uni\_min

dec\_min

tmp\_hora

uni\_hora

dec\_hora

tmp\_tonos

registro1

tmp\_t

inc\_frecuencia ; CAMBIAR LA VARIABLE FREC DE TONOS

ENDC

CBLOCK ; En las subrutinas no se debe fijar la dirección

BCD\_Centenas ; de la RAM de usuario. Se toma a continuación de

BCD\_Decenas ; la última asignada.

BCD\_Unidades

ENDC

CBLOCK

R\_ContA ; Contadores para los retardos.

R\_ContB

R\_ContC

ENDC

;SALIDAS Y ENTRADAS

#define led\_go\_on PORTE,1 ;Led amarillo

#define led\_reset PORTE,0 ;Led azul

#define parlante PORTE,2 ;nombre de la salida al parlante

#define boton\_inicio PORTA,0 ;En el puerto A se encuentran

#define boton\_reset PORTA,1 ;seteadas las entradas.

#define boton\_start PORTA,2

#define puerto\_seg PORTB

#define puerto\_min PORTC

#define puerto\_hora PORTD

org 0 ;inicio del programa

goto principal

principal

call seteos\_pic

clrf puerto\_seg ;salidas de leds al puerto B

clrf puerto\_min ;salidas de leds al puerto C

clrf puerto\_hora ;salidas de leds al puerto D

clrf tmp\_s ;limpia la variable de entrada de segundos

clrf tmp\_min ;limpia la variable de entrada de minutos

clrf tmp\_hora ;limpia la variable de entrada de horas

clrf tmp\_tonos ;limpia la variable de entrada de la cantidad de tonos

bcf led\_go\_on ;led amarillo off

bcf led\_reset ;led azul off

bcf parlante ;parlante apagado

menu\_inicio

btfsc boton\_inicio

goto menu\_inicio ;antirebote del boton inicio/arranque

call Retardo\_10ms

bsf led\_go\_on

soltar1

btfss boton\_inicio

goto soltar1

menu\_start

btfsc boton\_start

goto menu\_start ;antirebote del boton Start

call Retardo\_10ms

soltar2

btfss boton\_start

goto soltar2

contando

clrwdt

btfsc boton\_reset ; Pregunta si botón de reset presionado.

goto cnt\_reloj

bsf led\_reset ;Enciende el led de reset

bcf led\_go\_on ;Apaga el led de encendido

call Retardo\_1s

goto principal ;se reinicia el programa

cnt\_reloj

movfw tmp\_s

call BIN\_a\_BCD

movfw BCD\_Unidades

call bin\_hex

movwf uni\_seg

movfw BCD\_Decenas

call bin\_hex

movwf dec\_seg

swapf dec\_seg,f

movf dec\_seg,w

addwf uni\_seg,w

movwf puerto\_seg

call Retardo\_1s

incf tmp\_s,f

movfw tmp\_tonos ;tiempo de los tonos

sublw d'5' ; para que realice cada 5 segundos

btfss STATUS,Z

goto check

call generar\_sonido

check

incf tmp\_tonos,f

movfw tmp\_s

sublw d'60' ; Valor para que se vaya restando hasta al fin obtener 0 en la respuesta

btfss STATUS,Z

goto contando

incf tmp\_min

clrf tmp\_s

movfw tmp\_min

call BIN\_a\_BCD

movfw BCD\_Unidades

call bin\_hex

movwf uni\_min

movfw BCD\_Decenas

call bin\_hex

movwf dec\_min

swapf dec\_min,f

movf dec\_min,w

addwf uni\_min,w

movwf puerto\_min

movfw tmp\_min

sublw d'60' ; Valor para que se vaya restando hasta al fin obtener 0 en la respuesta

btfss STATUS,Z

goto contando

clrf tmp\_min

incf tmp\_hora

movfw tmp\_hora

call BIN\_a\_BCD

movfw BCD\_Unidades

call bin\_hex

movwf uni\_hora

movfw BCD\_Decenas

call bin\_hex

movwf dec\_hora

swapf dec\_hora,f

movf dec\_hora,w

addwf uni\_hora,w

movwf puerto\_hora

movfw tmp\_hora

sublw d'24' ;Valor para que se vaya restando hasta al fin obtener 0 en la respuesta

btfss STATUS,Z

goto contando

clrf tmp\_hora

goto contando

;FUNCIONES

seteos\_pic

banksel ANSELH

clrf ANSELH

clrf ANSEL

banksel TRISB

bsf boton\_inicio ;entrada de pines

bsf boton\_reset ;entrada de pines

bcf led\_go\_on ;salida de pines

bcf led\_reset ;salida de pines

bcf parlante ;salida de pines

clrf puerto\_seg ;salidas de leds al puerto D

clrf puerto\_min

clrf puerto\_hora

banksel PORTA

return

generar\_sonido

movlw d'1'

movwf tmp\_t

tono1

call Retardo\_200ms

btfsc parlante

goto alto

bajo

bsf parlante

goto on1

alto

bcf parlante

on1

decfsz tmp\_t,f

goto tono1

movlw d'1'

movwf tmp\_t

tono2

call Retardo\_200ms

btfsc parlante

goto alto2

bajo2

bsf parlante

goto on2

alto2

bcf parlante

on2

decfsz tmp\_t,f

goto tono2

movlw d'1'

movwf tmp\_t

tono3

call Retardo\_200ms

btfsc parlante

goto alto3

bajo3

bsf parlante

goto on3

alto3

bcf parlante

on3

decfsz tmp\_t,f

goto tono3

movlw d'1'

movwf tmp\_t

tono4

call Retardo\_200ms

btfsc parlante

goto alto4

bajo4

bsf parlante

goto on4

alto4

bcf parlante

on4

decfsz tmp\_t,f

goto tono4

movlw d'4'

movwf tmp\_t

tono5

call Retardo\_200ms

btfsc parlante

goto alto5

bajo5

bsf parlante

goto on5

alto5

bcf parlante

on5

decfsz tmp\_t,f

goto tono5

clrf tmp\_tonos

bcf parlante

RETURN

bin\_hex ; Tabla para display hex

addwf PCL,F

Tabla

;binarios invertidos para adaptarse a la salida indicada por la tarjeta del pic

RETLW b'00000000' ; Retorna con el código del 0

RETLW b'10000000' ; Retorna con el código del 1

RETLW b'01000000' ; Retorna con el código del 2

RETLW b'11000000' ; Retorna con el código del 3

RETLW b'00100000' ; Retorna con el código del 4

RETLW b'10100000' ; Retorna con el código del 5

RETLW b'01100000' ; Retorna con el código del 6

RETLW b'11100000' ; Retorna con el código del 7

RETLW b'00010000' ; Retorna con el código del 8

RETLW b'10010000' ; Retorna con el código del 9

;librerías adicionales proporcionadas en el libro tilizadas para los retardos y la conversion

BIN\_a\_BCD

clrf BCD\_Centenas ; Carga los registros con el resultado inicial.

clrf BCD\_Decenas ; En principio las centenas y decenas a cero.

movwf BCD\_Unidades ; Se carga el número binario a convertir.

BCD\_Resta10

movlw .10 ; A las unidades se les va restando 10 en cada

subwf BCD\_Unidades,W ; pasada. (W)=(BCD\_Unidades) -10.

btfss STATUS,C ; ¿C = 1?, ¿(W) positivo?, ¿(BCD\_Unidades)>=10?

goto BIN\_BCD\_Fin ; No, es menor de 10. Se acabó.

BCD\_IncrementaDecenas

movwf BCD\_Unidades ; Recupera lo que queda por restar.

incf BCD\_Decenas,F ; Incrementa las decenas y comprueba si ha llegado

movlw .10 ; a 10. Lo hace mediante una resta.

subwf BCD\_Decenas,W ; (W)=(BCD\_Decenas)-10).

btfss STATUS,C ; ¿C = 1?, ¿(W) positivo?, ¿(BCD\_Decenas)>=10?

goto BCD\_Resta10 ; No. Vuelve a dar otra pasada, restándole 10 a

BCD\_IncrementaCentenas ; las unidades.

clrf BCD\_Decenas ; Pone a cero las decenas

incf BCD\_Centenas,F ; e incrementa las centenas.

goto BCD\_Resta10 ; Otra pasada: Resta 10 al número a convertir.

BIN\_BCD\_Fin

swapf BCD\_Decenas,W ; En el nibble alto de (W) también las decenas.

addwf BCD\_Unidades,W ; En el nibble bajo de (W) las unidades.

return ; Vuelve al programa principal.

Retardo\_200ms ; La llamada "call" aporta 2 ciclos máquina.

movlw d'200' ; Aporta 1 ciclo máquina. Este es el valor de "M".

goto Retardos\_ms ; Aporta 2 ciclos máquina.

Retardo\_100ms ; La llamada "call" aporta 2 ciclos máquina.

movlw d'100' ; Aporta 1 ciclo máquina. Este es el valor de "M".

goto Retardos\_ms ; Aporta 2 ciclos máquina.

Retardo\_10ms ; La llamada "call" aporta 2 ciclos máquina.

movlw d'10' ; Aporta 1 ciclo máquina. Este es el valor de "M".

goto Retardos\_ms ; Aporta 2 ciclos máquina.

; Aporta 1 ciclo máquina. Este es el valor de "M".

;

; El próximo bloque "Retardos\_ms" tarda:

; 1 + M + M + KxM + (K-1)xM + Mx2 + (K-1)Mx2 + (M-1) + 2 + (M-1)x2 + 2 =

; = (2 + 4M + 4KM) ciclos máquina. Para K=249 y M=1 supone 1002 ciclos máquina

; que a 4 MHz son 1002 µs = 1 ms.

;

Retardos\_ms

movwf R\_ContB ; Aporta 1 ciclo máquina.

R1ms\_BucleExterno

movlw d'249' ; Aporta Mx1 ciclos máquina. Este es el valor de "K".

movwf R\_ContA ; Aporta Mx1 ciclos máquina.

R1ms\_BucleInterno

nop ; Aporta KxMx1 ciclos máquina.

decfsz R\_ContA,F ; (K-1)xMx1 cm (cuando no salta) + Mx2 cm (al saltar).

goto R1ms\_BucleInterno ; Aporta (K-1)xMx2 ciclos máquina.

decfsz R\_ContB,F ; (M-1)x1 cm (cuando no salta) + 2 cm (al saltar).

goto R1ms\_BucleExterno ; Aporta (M-1)x2 ciclos máquina.

return ; El salto del retorno aporta 2 ciclos máquina.

Retardo\_1s ; La llamada "call" aporta 2 ciclos máquina.

movlw d'10' ; Aporta 1 ciclo máquina. Este es el valor de "N".

goto Retardo\_1Decima ; Aporta 2 ciclos máquina.

Retardo\_1Decima

movwf R\_ContC ; Aporta 1 ciclo máquina.

R1Decima\_BucleExterno2

movlw d'100' ; Aporta Nx1 ciclos máquina. Este es el valor de "M".

movwf R\_ContB ; Aporta Nx1 ciclos máquina.

R1Decima\_BucleExterno

movlw d'249' ; Aporta MxNx1 ciclos máquina. Este es el valor de "K".

movwf R\_ContA ; Aporta MxNx1 ciclos máquina.

R1Decima\_BucleInterno

nop ; Aporta KxMxNx1 ciclos máquina.

decfsz R\_ContA,F ; (K-1)xMxNx1 cm (si no salta) + MxNx2 cm (al saltar).

goto R1Decima\_BucleInterno ; Aporta (K-1)xMxNx2 ciclos máquina.

decfsz R\_ContB,F ; (M-1)xNx1 cm (cuando no salta) + Nx2 cm (al saltar).

goto R1Decima\_BucleExterno ; Aporta (M-1)xNx2 ciclos máquina.

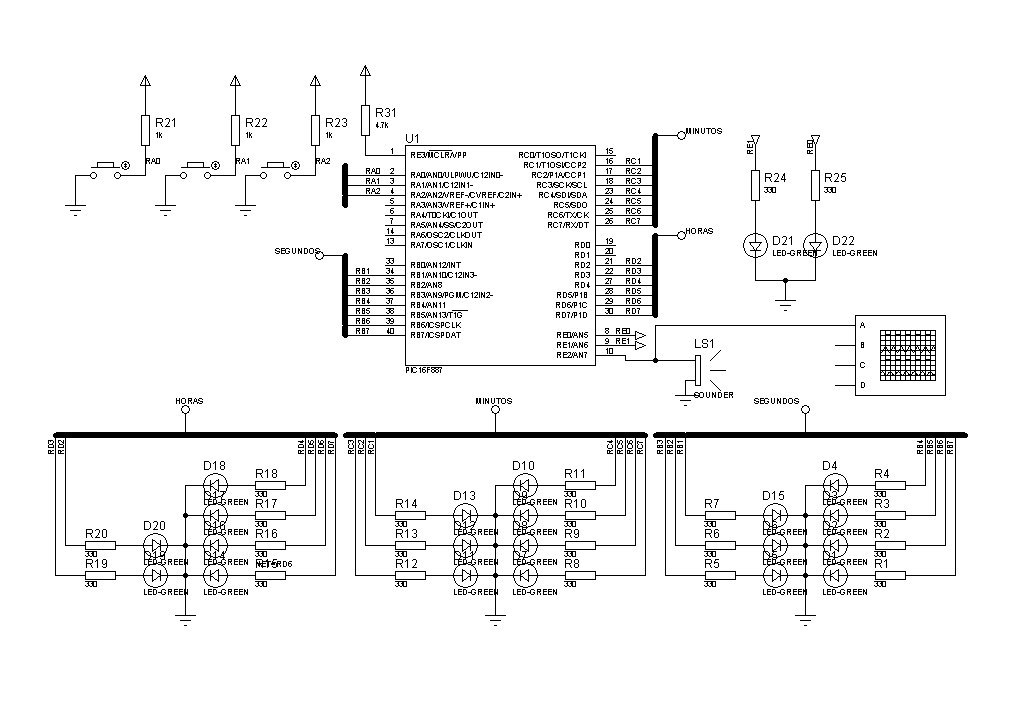
decfsz R\_ContC,F ; (N-1)x1 cm (cuando no salta) + 2 cm (al saltar).

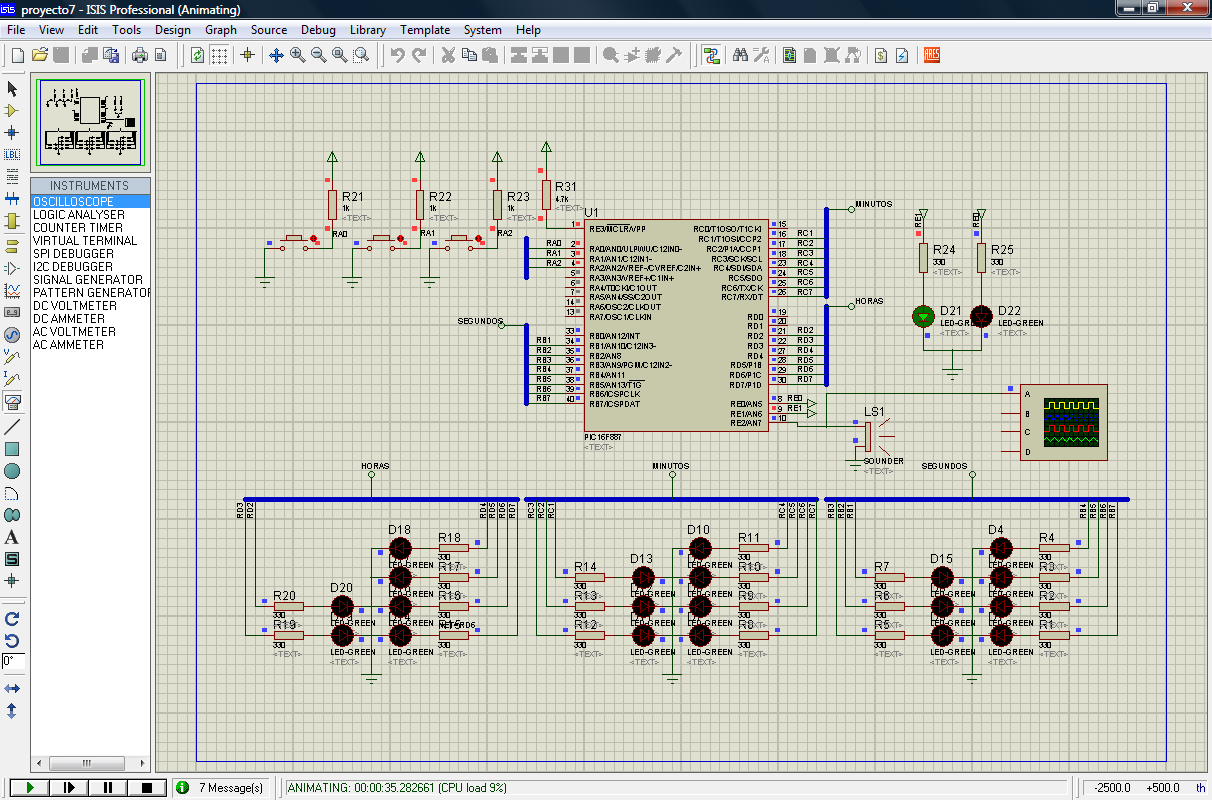
goto R1Decima\_BucleExterno2 ; Aporta (N-1)x2 ciclos máquina.

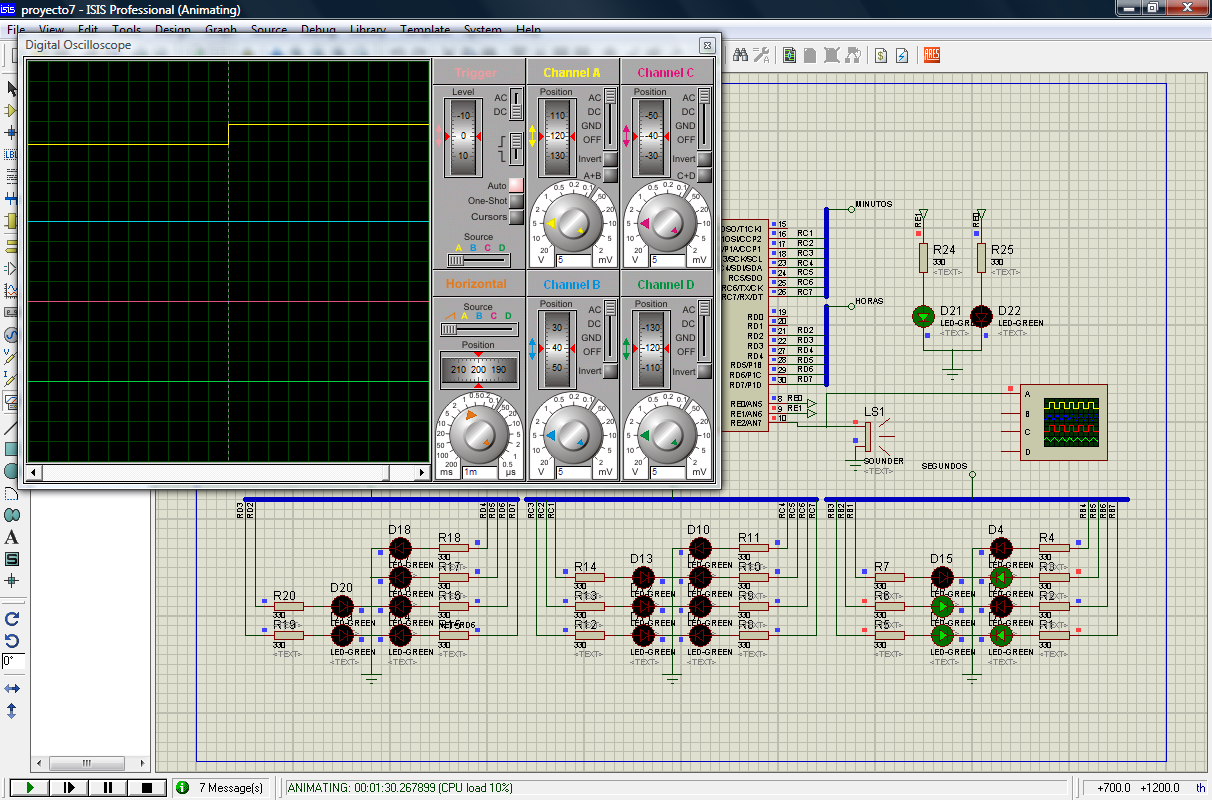
return ; El salto del retorno aporta 2 ciclos máquina.

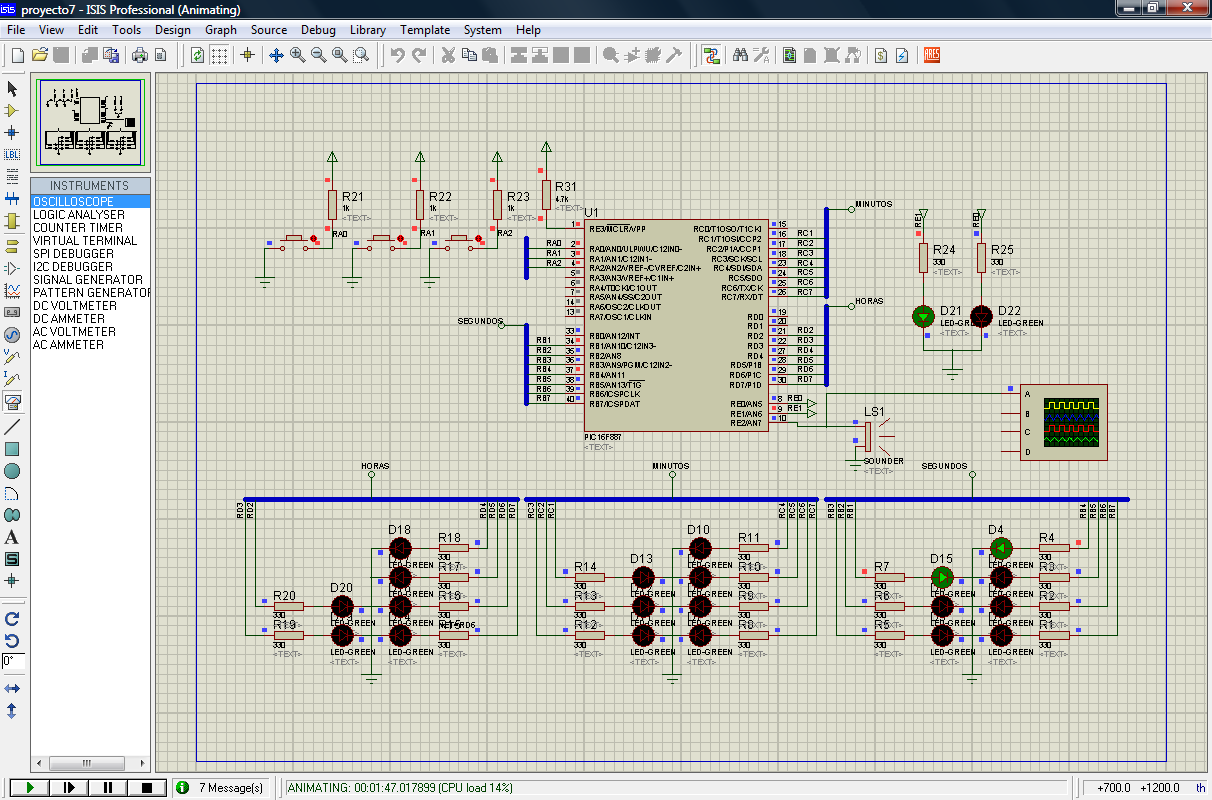
END;finalizacion del programa

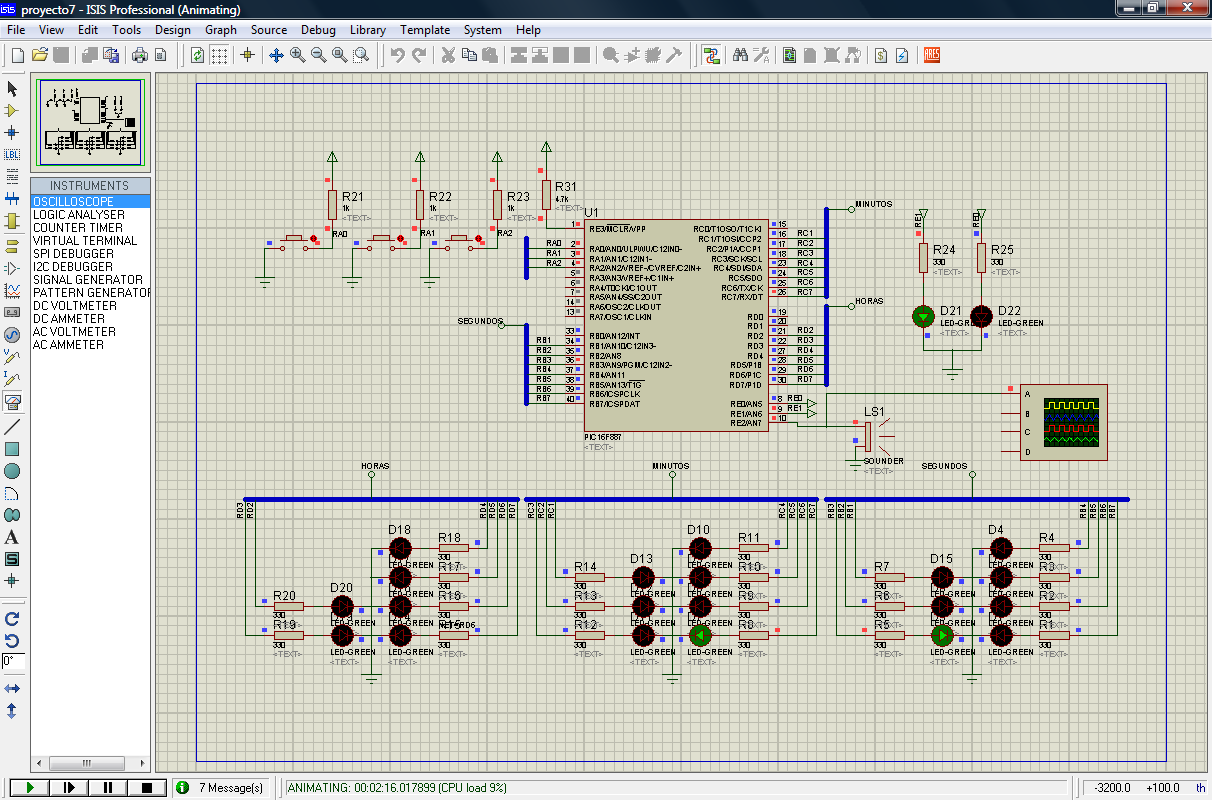
1. **CIRCUITO EN PROTEUS**

******









1. **CONCLUSIONES**

Este proyecto fue implementado por medio de retardos, se pudo comprobar que funcionan tan bien como con los temporizadores., además tenemos la facilidad de uno de los recursos del libro Microcontrolador PIC16F84, en el cual se encuentra el código de la librería de retardos de algunos valores lista para ser utilizada dentro de cualquier proyecto que los requiera, como por ejemplo relojes, alarmas, cronómetros, etc.

La implementación de llamadas a funciones o subrutinas por medio de la instrucción “CALL” proporcionó gran ayuda al momento de preguntar por el arranque del programa, la pausa del reloj, y la continuación del mismo para evitar la pérdida del valor con el cual se generó la pausa del reloj. La utilización de esta instrucción fue tratar de contrarrestar el efecto anti rebote generado al momento de presionar y soltar las botoneras respectivas, efecto que algunas ocasiones producía errores en el funcionamiento del PIC.

La implementación del proyecto en físico, es decir en la placa PCB, fue el paso final del proyecto y esto nos permitió comprobar y dar ese paso más para poder ver cómo funcionaba el reloj en la vida real, más no dejarlo solo en simulación. Para este efecto debimos poner en práctica los conocimientos previos sobre manipulación de elementos, uso del multímetro para probar continuidad, y experiencia en soldar. Como resultado se obtuvo un proyecto que puede ser empleado para otros proyectos, configurando las mismas entradas que tenemos en este momento en el proyecto.

1. **RECOMENDACIONES**

Cada persona posee una manera distinta de programar, así como también cada quien tiene su fuerte. En este caso, el reloj pudo haber sido implementado tanto con retardos, como con temporizadores, sin embargo e uso de cada uno depende de la persona y su dominio en el tema. Es muy recomendable revisar los materiales del curso, ya que en este caso, el libro aportó una parte muy importante del reloj como lo fue la librería de retardo previamente cargada en lugar de comenzar a programar y quizás por falta de experiencia cometer algún error en la creación de los propios retardos.

En algunas ocasionas recomiendo cambiar las directivas directamente en el código y no en el configuration bits del MPLAB, esto puede acarrear problemas al momento de comprobar el funcionamiento de nuestro PIC en una placa porque al momento de grabarse lo haría de una manera incorrecta y no funcionaría como se espera que trabaje.

Soldar con mucha precaución la placa ya que un pequeño error de soldadura puede hacer que el proyecto no funcione, revisar que las piezas soldadas estén en su lugar posicionadas y polarizadas correctamente, verificar cortos generados por mala soldadura, etc.

Revisar la placa antes de ser soldada, muchas veces existen cortos, porque las pistas están muy pegadas unas a otras, otras es la existencia de pequeñas partes de suelda que se quedan luego de ser soldada, así como que a veces no hace contacto la suelda.

En mi caso usé un programador propio, tener cuidado de conectar los pines correctamente, ya que el microcontrolador es sensible a esos cambios bruscos de voltaje que se puede dar cuando no está conectado de forma adecuada. Preguntar a alguien con experiencia si no se sabe cómo grabar el PIC.