# LABORATORIO 4. MEMORIAS EEPROM-FLASH (Junio 2017)

Ingrid Catherin Morales, Alejandro Antonio Nuñez, *U.P.T.C*

***Resumen*— En esta guía se describe de forma detallada el diseño, simulación e implementación de un generador de señales senoidal, cuadrada y diente de sierra, cada una a 255 frecuencias diferentes posibles. Se hizo uso de LCD, donde se aprecia la señal generada y su frecuencia; y un teclado alfanumérico, a través del cual se ingresa la frecuencia deseada y se hace el cambio entre una señal y otra. Para la generación de las señales se hizo uso de tablas y del Timer2 en un modo auto-configurable de acuerdo a la frecuencia. En el documento se puede apreciar la simulación en Proteus y el funcionamiento correcto del montaje.**

***Índice de Términos*— Generador de Señales, LCD, señal cuadrada, señal diente de sierra, señal senoidal, teclado alfanumérico, Timer2.**

## I. INTRODUCCIÓN

UN generador de señales es un aparato que permite sintetizar señales eléctricas. Existen muchas maneras de hacer uno de estos, y una de las más fáciles es usar un circuito digital que entregue los valores correspondientes a la cuantización de las señales, luego pasarlos por un conversor DAC y finalmente por un filtro.

Dado que el uso de tablas permite mejor procesado de la señal (simplicidad en cálculos, pero consumo de memoria del programa), es uno de los métodos más comunes en la generación de para microcontroladores de bajas especificaciones, sin embargo, en la medida que el procesador sea más potente, será más eficiente hacer la operación.

En la mayoría de sistemas digitales después de una pérdida del fluido eléctrico, se suele restablecer la información de la mayoría de los procesos que, realizados antes del incidente, y para ellos se usan memorias de guardado «permanente».

En esta guía se desarrolla un generador de señales de forma digital, con un microcontrolador PIC 18F4550. También se hace uso de las memorias mencionadas anteriormente para el guardado de la frecuencia y el tipo de señal generada.

## II. MATERIALES Y EQUIPOS

- Ordenador con las aplicaciones PROTEUS y MPLAB X.
- Microcontrolador de la serie PIC18F4550.
- Programador de microcontroladores PIC.
- Protoboard
- Resistencias, trimmer, pantalla LCD y teclado matricial.
- Fuente de alimentación 5V.
- Alambre.
- Compilador de microchip XC8.

## III. PROCEDIMIENTO

Implementar un generador de señales. Con las siguientes señales:

- Seno
- Cuadrada
- Diente de sierra

Se deben tener 255 frecuencias diferentes. Se debe visualizar en la LCD el nombre y la frecuencia de la señal generada. El cambio y modificación de la frecuencia de las señales debe ser por medio de un teclado matricial. El funcionamiento del programa no se deberá ver afectado por fallos de energía (uso de memorias). La programación se debe realizar en lenguaje C, para esto haga uso del compilador XC8 y un pic18f4550.

## IV. DESARROLLO DE LA PRÁCTICA

*Diagramas de Flujo*

Se hace uso del microcontrolador para obtener un generador de señales de las siguientes formas: senoidal, cuadrada (TTL) y diente de sierra. Se utiliza la siguiente configuración para el microcontrolador:

- Salida a DAC, esta salida es de un byte, y los pines del microcontrolador usados en este fin son en orden ascendente: 4 bits más bajos del puerto A, 2 bits más bajos del puerto C y 2 bits más altos del mismo puerto.
- Conexión al teclado matricial, el teclado se conecta en el puerto B
- Conexión a la pantalla LCD, Se conecta con 8 hilos (corresponden al puerto D) y los bits de control se conectan en el puerto E

Para el correcto funcionamiento del programa se cuenta con un par de datos que almacenan las frecuencias: el primero tiene la frecuencia lineal (un solo registro y tiene un binario natural), el segundo es un vector de 2 posiciones, que corresponde a el

código BCD de la frecuencia. Se usa este par de datos, porque el primero es útil para la configuración del Timer, pero es algo complicado para visualizar los datos en la pantalla LCD.

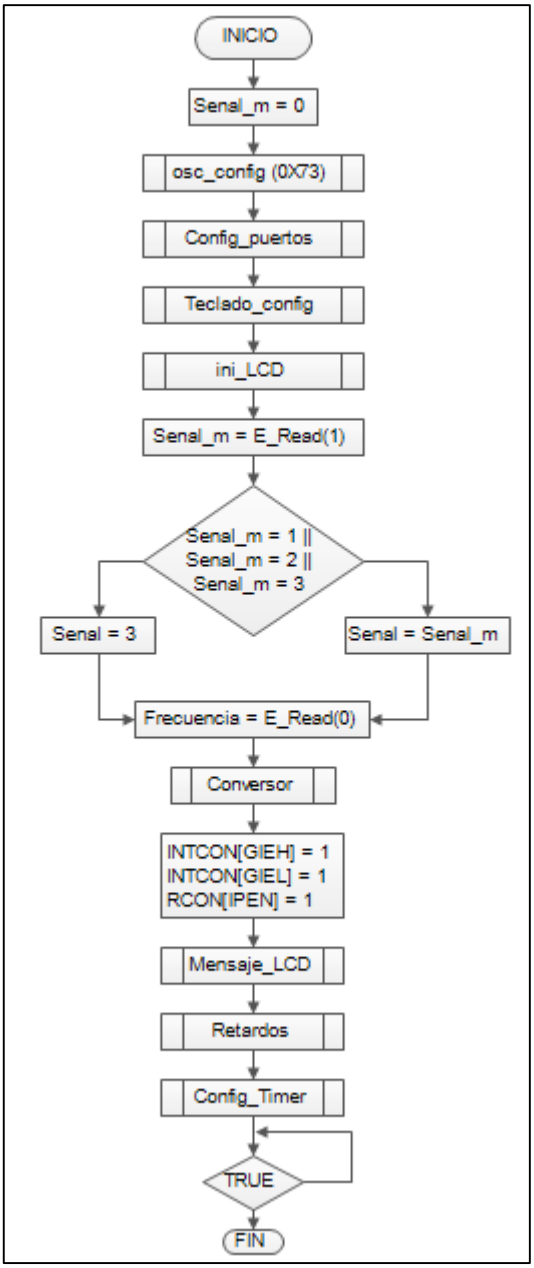


Fig 1. Diagrama de flujo del Main del programa.

En la Figura 1 se observa el diagrama de flujo que corresponde al Main o programa principal, el cual contiene secuencias de inicio de registros, periféricos, y puertos. En este se configura la pantalla LCD y las interrupciones (globales, por Timer y por cambio de estado en puerto B), es decir, su habilitación y sus distintos niveles de prioridad. También se leen los datos de la memoria para inicializar los registros que contienen la información de la adecuada ejecución del programa.

Se utiliza las interrupciones para hacer el cambio periódico de las muestras entregadas al DAC (baja prioridad) y atender el teclado (alta prioridad). Se considera una interrupción de más alto peso el teclado, porque no se quiere hacer esperar a usuario al momento de modificar el tipo de señal o la frecuencia.

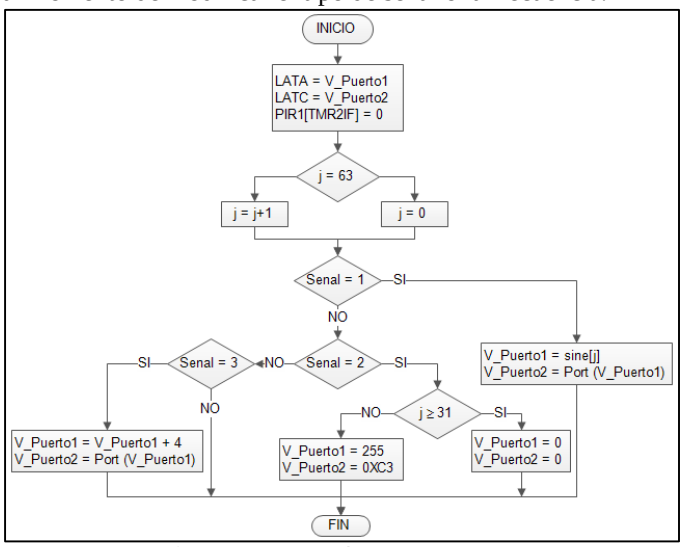


Fig 2. Diagrama de fujo de la interrupción de baja prioridad.

En la Figura donde se aprecia que la interrupción de baja prioridad ISR_L, ocurre periódicamente de acuerdo con la configuración del Timer 2, en ella se asigna directamente los valores pre-calculados para el bus del DAC. Luego de indica que ya se atendió la interrupción (0 lógico para la bandera del Timer 2) y se procesa el siguiente dato que se quiere asignar al bus que va conectado al DAC.

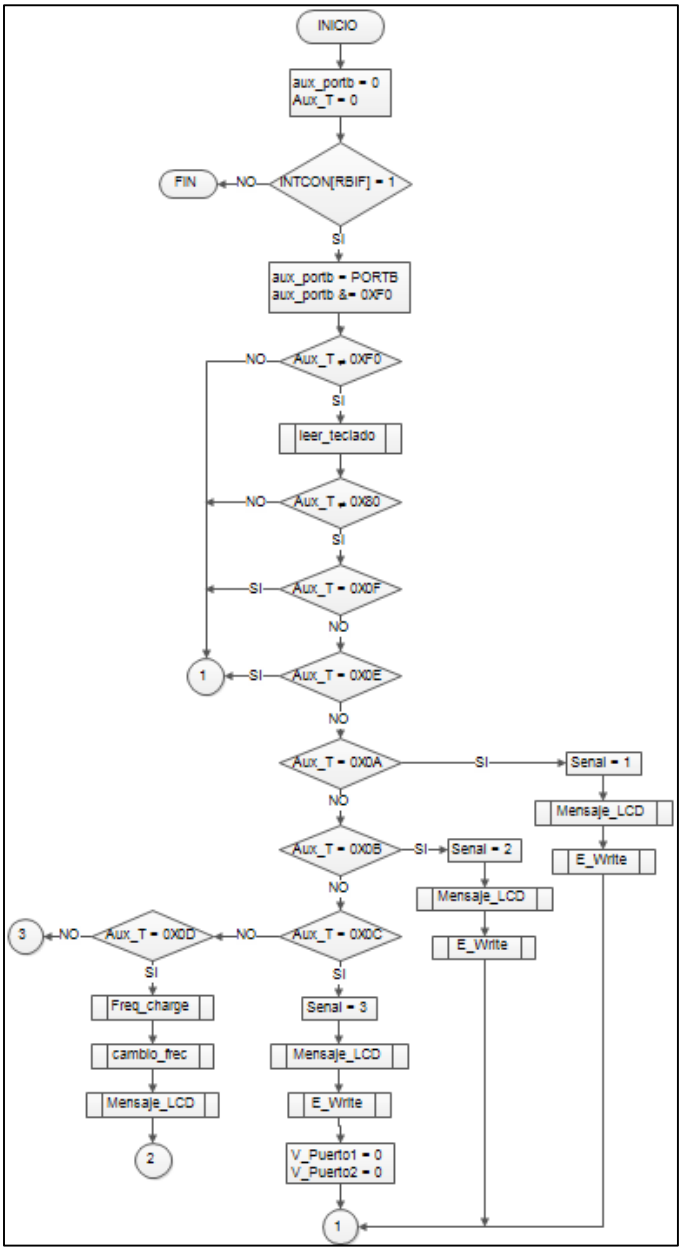


Fig 3. Diagrama de flujo de la interrupción de alta prioridad. Parte 1.

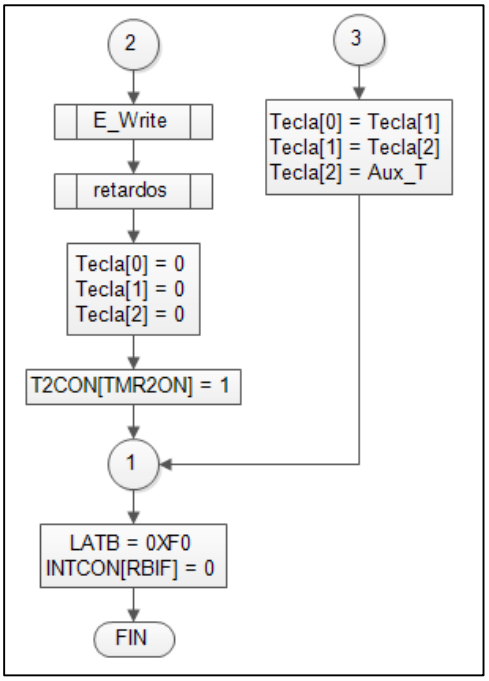


Fig 4. Diagrama de flujo de la interrupción de alta prioridad. Parte 2.

En las Figuras 3 y 4, se observa el diagrama de flujo para la interrupción de más alta prioridad, donde se atiende al teclado y se procesan los cambios correspondientes a los comandos recibidos. Los comandos usados son:

- Números: La asignación de los datos se hace forma tal que se hace un intercambio entre los registros adyacentes, entonces un dato se deshecha y otro ingresa. Además, es importante resaltar que solo los 3 últimos números oprimidos son válidos para un cambio de frecuencia y cada que se ejecute un cambio de señal o frecuencia el banco de registros asignado es reiniciado.
- Los correspondientes las letras A, B, C del teclado, hacen cambios en el tipo de señal generada y corresponde a la señal senoidales, cuadrada y diente de sierra respectivamente.
- La letra D corresponde a un ENTER y hace efectivo el cambio en la frecuencia.

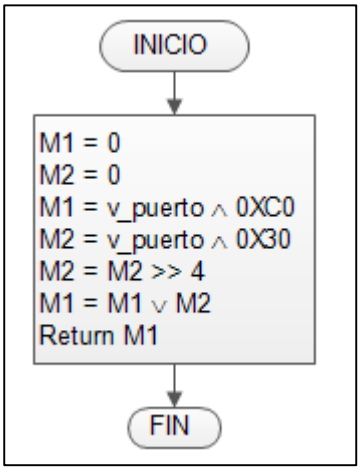


Fig 5. Diagrama de flujo de port.

En la Figura 5 se parecía la función utilizada para calcular el dato que se debe asignar al puerto C (parte alta del bus para el DAC).

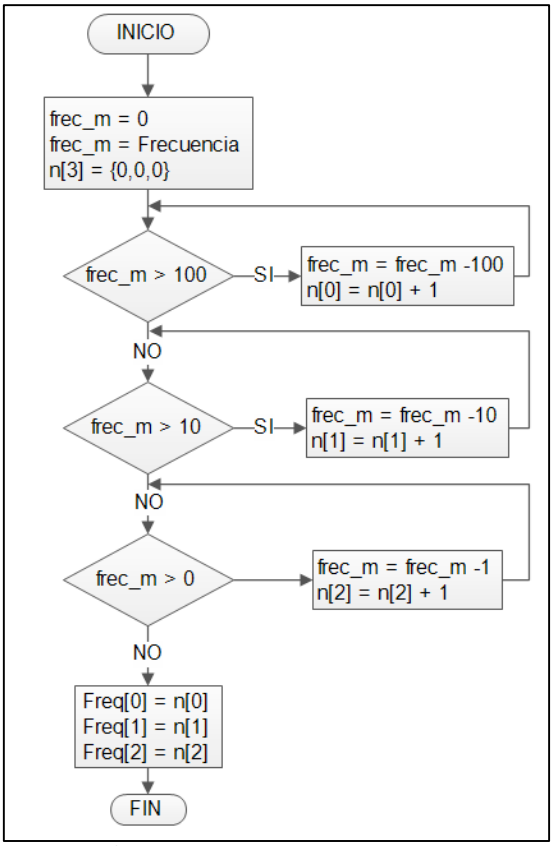


Fig 6. Diagrama de flujo del conversor.

En la Figura 6 se aprecia la función encargada de convertir una frecuencia de binario natural (un solo registro) a código BCD (Vector de 3 posiciones Freq). Esta función es de especial utilidad al iniciar el sistema, porque en la memoria E2PROM se guarda un dato binario natural y para el correcto funcionamiento del ejercicio se requiere el par de datos Binario natural y código BCD.

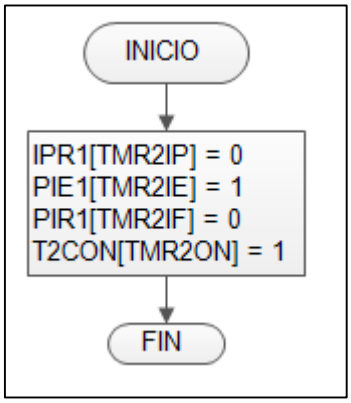


Fig 7. Diagrama de flujo donde se configure el Timer 2.

En la Figura 7 se observa el caso en que se configura las interrupciones causadas por Timer 2 (Nivel de prioridad, habilitador de interrupción, Timer encendido).

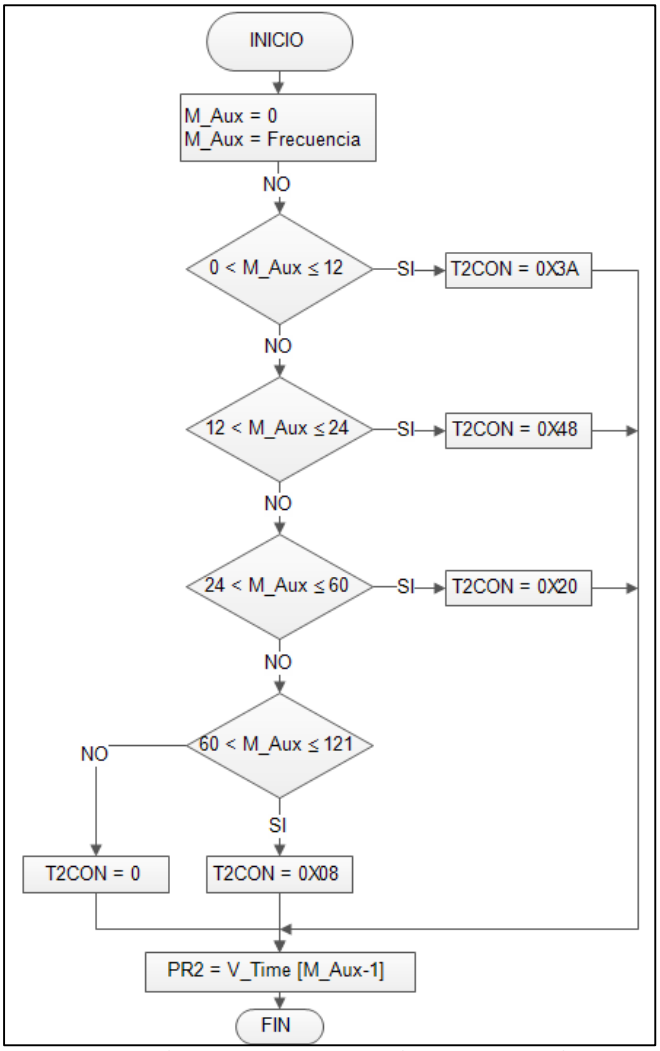


Fig 8. Diagrama de flujo de retardos para los diferentes rangos de frecuencias.

En la Figura 8 se encuentra el diagrama de flujo de los retardos, función que se encarga de configurar el tiempo en que el Timer 2 produce periódicamente las interrupciones (Pre-escala, Post-escala y Valor de comparación, es decir, el registro PR2). El valor del PR2 se obtiene de una tabla de 255 posiciones (correspondientes con las frecuencias).

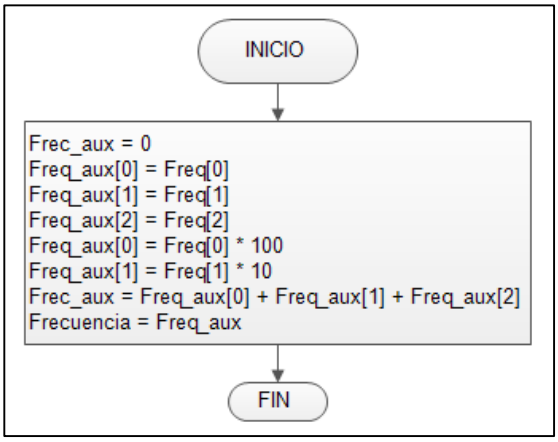


Fig 9. Diagrama de flujo para cargar el dato de la frecuencia.

La subrutina mostrada en la Figura 9, cambia el registro directamente relacionado (Registro con el valor binario natural de la frecuencia) con la rutina de cambio de la configuración de las interrupciones periódicas del Timer 2.

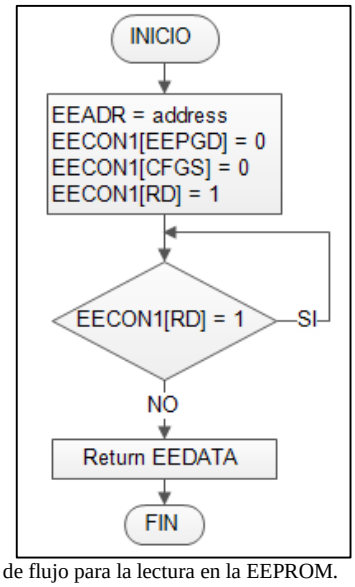


Fig 10. Diagrama de flujo para la lectura en la EEPROM.

En la Figura 10 se muestra la función que regresa la lectura de la memoria E2PROM en una posición deseada. Utilizada en la inicialización del sistema.

El diagrama de flujo de la figura 11 y 12 describe la rutina Freq_charge, que se encarga de guardar los datos del banco de registros utilizado en la interrupción (Vector Tecla) al banco de registros (Banco de registro con el valor de la frecuencia binaria) relacionado con la frecuencia (Vector Freq) de las señales generadas. Además de limitar los valores de frecuencia entre 1 y 255Hz.

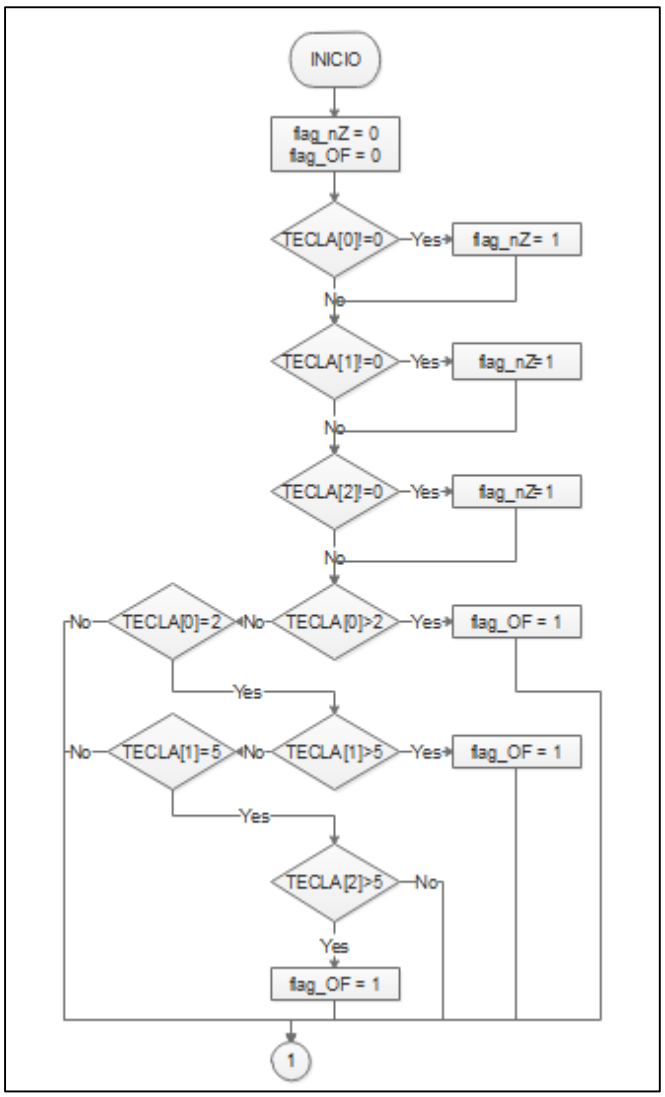


Fig 11. Diagrama de flujo de la rutina Freq_Charge. Parte 1

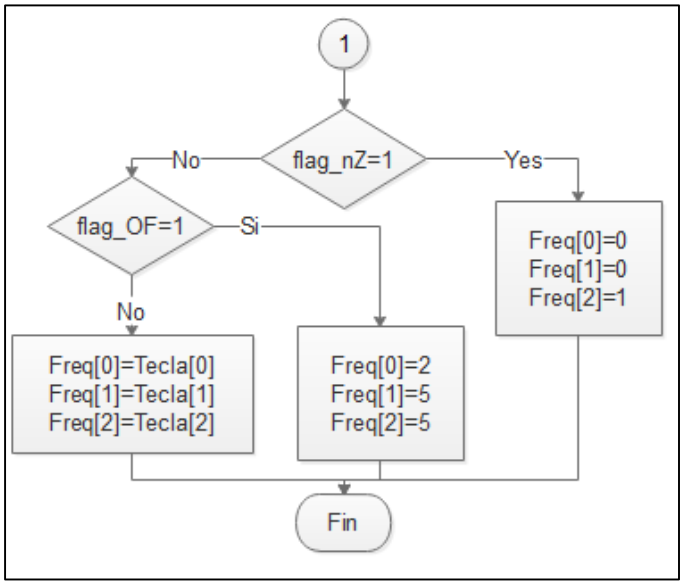


Fig 12. Diagrama de flujo de la rutina Freq_Charge. Parte 2

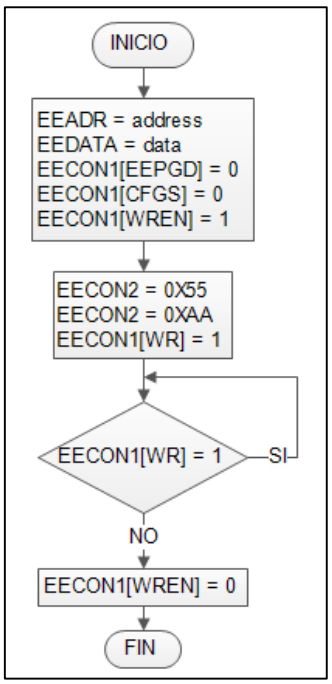


Fig 13.Diagrama de flujo para la escritura en la EEPROM.

En la Figura 13, se detalla la función que guarda datos en la memoria E2PROM en una posición deseada. Es utilizada en la interrupción por teclado, cuando se cambia el tipo de función o la frecuencia, es decir, que solo se guardan dos datos en la memoria E2PROM.

NOTA: Se usaron 64 muestras para la reconstrucción de las señales cumpliendo con el teorema del muestreo. La reconstrucción de la señal senoidal se hizo con una tabla, pero las otras se calculaban a partir del dato anterior (diente de sierra y señal cuadrada)

*Simulación*

Una vez terminado el programa se procede a simular con la ayuda de Proteus, como se observa a continuación:

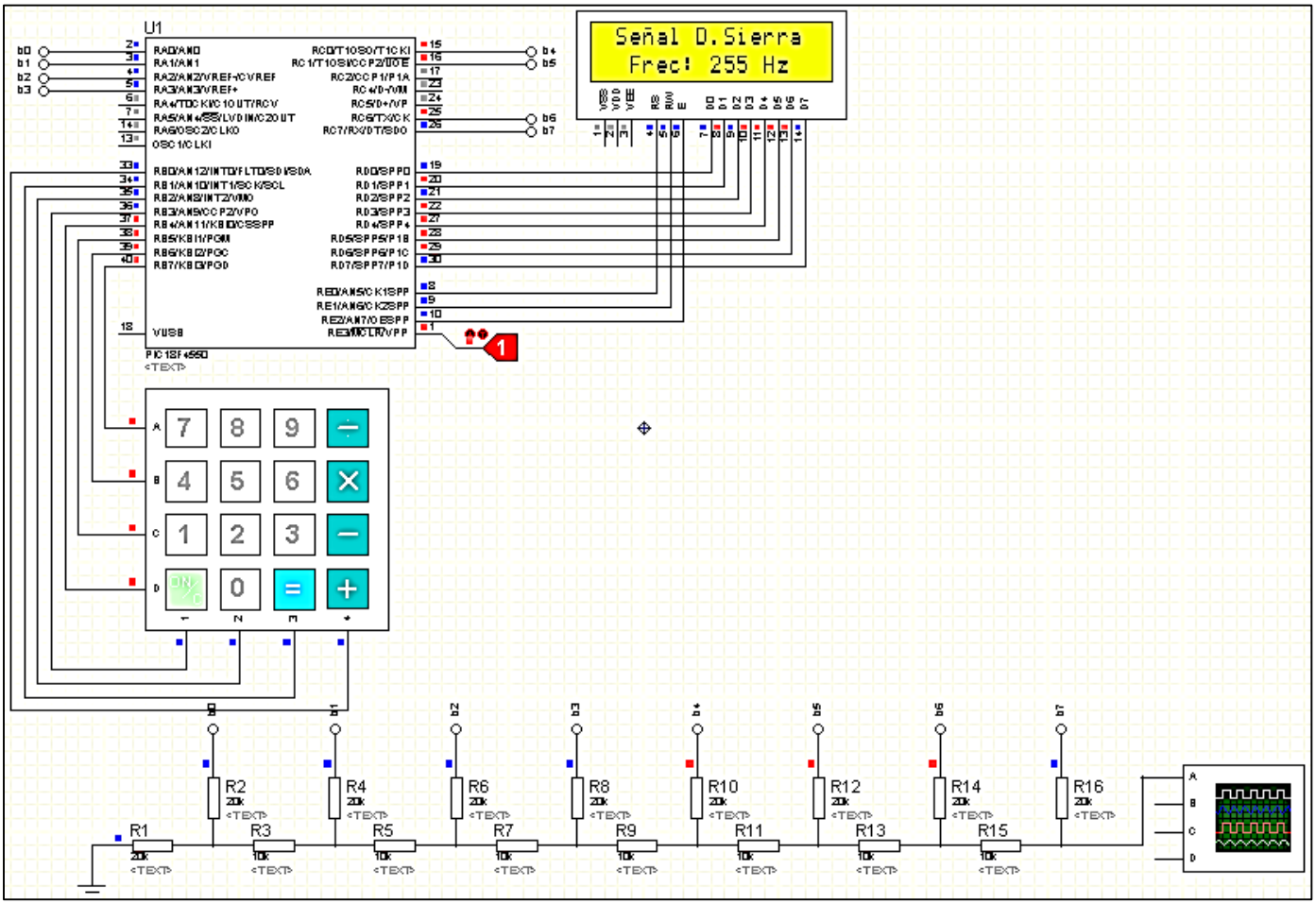


Fig 14. Simulación en Proteus del generador de señales.

En la Figura 14 se puede observar de forma más detenida la conexión de los puertos del PIC en uso. Se aprecia que el teclado alfanumérico ocupa la totalidad del PORTB, la LCD ocupa el PORTD y el PORTE, y la red R2R ocupa los cuatro

bits de más bajo peso del PORTA y los dos de más alto y más bajo peso en el PORTC.

NOTA: Cabe resaltar que la red R2R está conformada por resistencias de 10KΩ y 20KΩ debido a su gran sencillez de hallazgo en mercado, a la vez que se asegura la protección de los puertos.

A continuación, se muestran las tres señales a diferentes frecuencias, pero a una misma escala tanto en el eje vertical como horizontal, corroborando el correcto funcionamiento del programa.

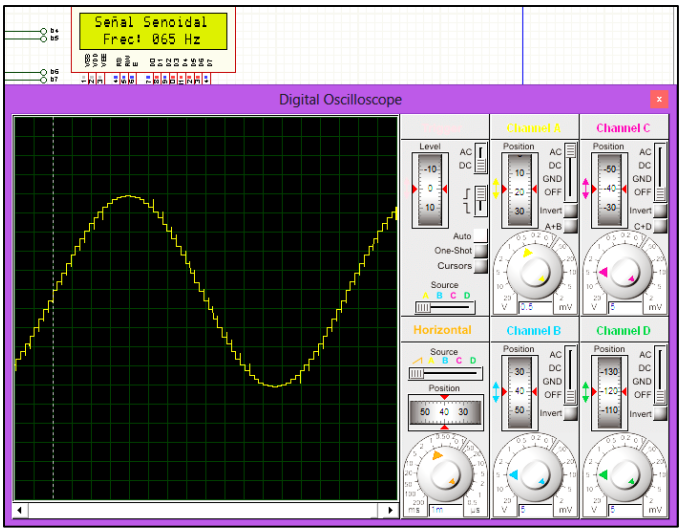


Fig 15. Señal senoidal a 65Hz.

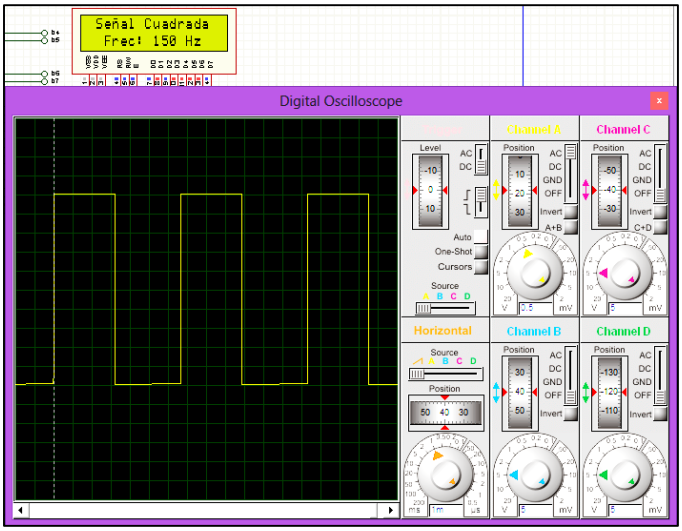


Fig 16. Señal cuadrada a 150Hz

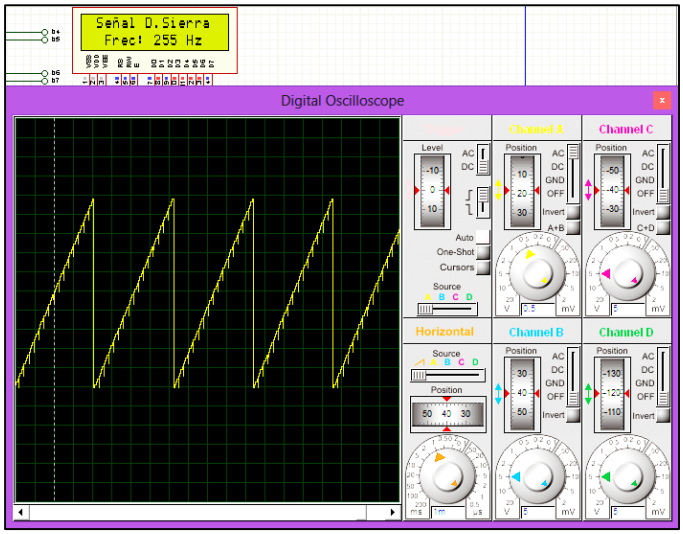


Fig 17. Señal diente de sierra a 255Hz.

*Implementación*

Finalmente se procede a la realización del montaje del circuito y a realizar las respectivas pruebas de funcionamiento.

Se tuvo en cuenta el conexionado de cada una de las terminales de la LCD siguiendo en cuenta el siguiente esquema:

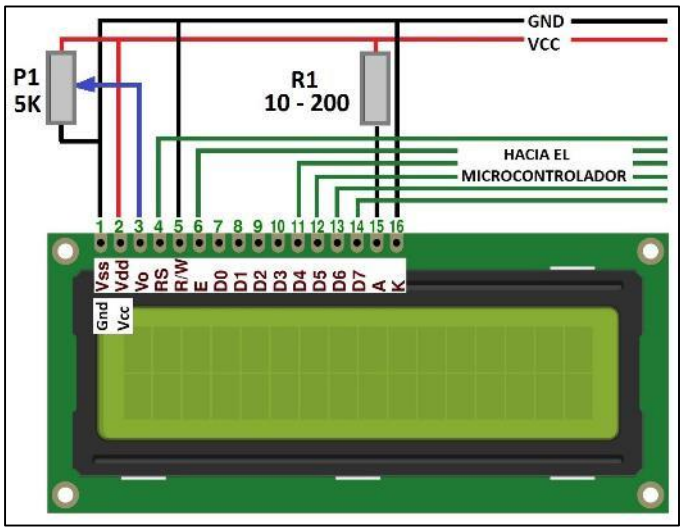


Fig 18. Conexión de pines de la LCD [1].

El montaje final del generador de señales fue:

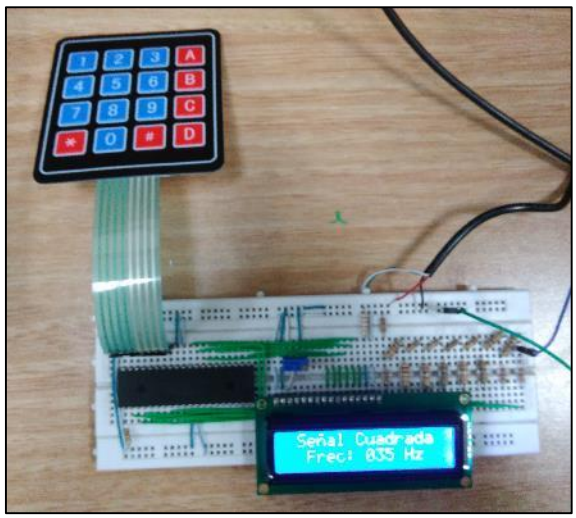


Fig 19. Implementación Generador de señales con PIC18F4550, LCD y teclado.

Circuito a partir del cual se muestran las señales generadas con la ayuda del osciloscopio. En esta ocasión las tres señales se muestran a la misma frecuencia.

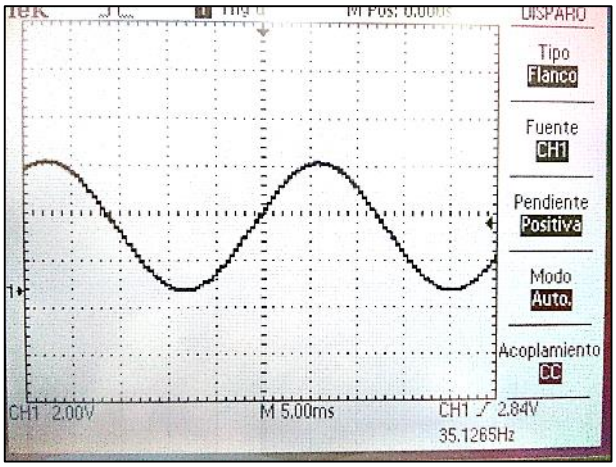


Fig 20. Señal senoidal a 35Hz.

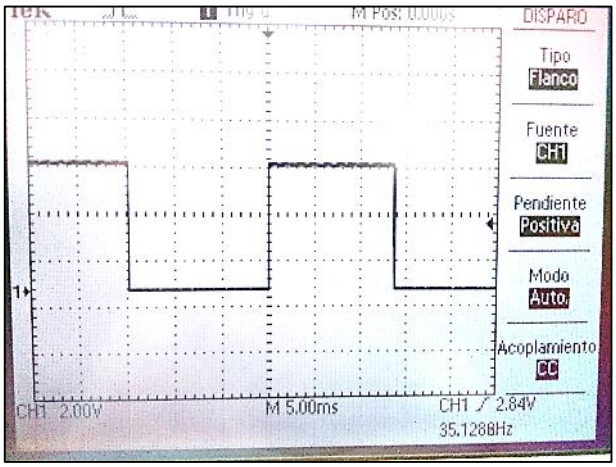


Fig 21. Señal cuadrada a 35Hz.

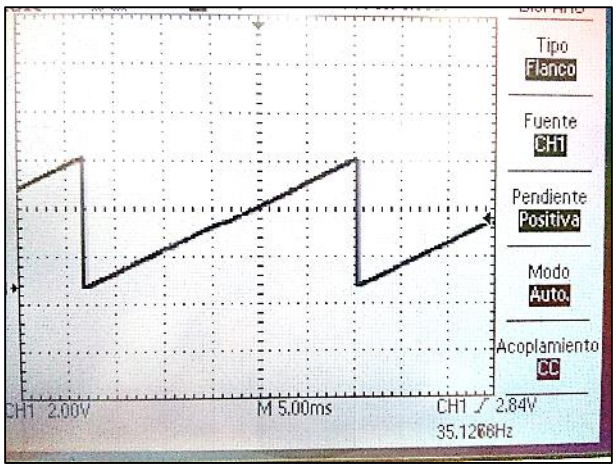


Fig 22. Señal diente de sierra a 35Hz.

## V. CONCLUSIONES

La utilización del lenguaje C permite mayor flexibilidad y facilidad en la implementación de circuitos que cumpla con ciertas secuencias lógicas, pero se pierde eficiencia y control sobre la ejecución del programa, por esta razón si se requiere tener un control de tiempos, para lo cual se debe configurar las interrupciones por Timer.

Se debe evaluar anticipadamente y de manera correcta los momentos en que las interrupciones tanto de alta como de baja prioridad son ejecutadas, debido a que pueden existir colisiones en las ordenes sobre los mismos, causando un mal funcionamiento del programa.

Es de suma importancia realizar un buen montaje, debido a que, a pesar de tener un programa perfecto en diseño y simulación, diversos factores físicos pueden llevar a distorsionar la señal e incluso deteriorarla por completo. Se recomienda prestar atención a la conexión de los dispositivos ajenos al PIC, como es el caso de la LCD, debido a que cuenta con pines que no existen en simulación y no son usados en el código, factor que pude conllevar a un juicio erróneo de mal funcionamiento del programa.

## REFERENCIAS

[1] http://www.mundoscratch.website.uy/category/arduino-2/ejercicios-de-arduino/
[2] http://www.microchip.com