PROGRAMACIÓN DE NOTAS MUSICALES EN MICROCONTROLADORES

Programación de notas musicales en microcontroladores

José Juan Torres Duarte,
josejuan_td@hotmail.com
Marco Antonio Quintero Mercado,
maquinterom@gmail.com
Francisco Alejandro Chávez Estrada,
falexchavez@hotmail.es
Juan Carlos Herrera Lozada,
jlozada@ipn.mx
Mauricio Olguín Carbajal,
molguinc@ipn.mx
Jesús Antonio Álvarez Cedillo,
jaalvarez@ipn.mx

IPN CIDETEC
Instituto Politécnico Nacional
Centro de Innovación y Desarrollo Tecnológico en Cómputo

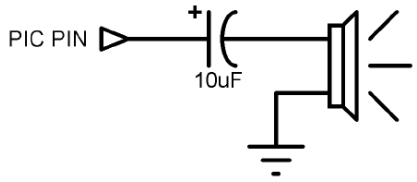
Abstract

En este trabajo se presenta un análisis dirigido a la capacidad del microcontrolador PIC16F628A para generar frecuencias que deriven en tonos musicales. Se diseñó una metodología con subdivisiones (bemol y sostenido) para diversificar más la generación de frecuencias utilizando el lenguaje de alto nivel PicBasic Pro.

Introducción

PicBasic es un lenguaje de alto nivel para programar microcontroladores PIC. Desarrollado por la empresa Melabs, PicBasic Pro consta de un conjunto de instrucciones diseñadas a la medida para ayudar al diseñador principiante o al profesional.
La instrucción SOUND en PicBasic Pro genera un tono y/o ruido blanco en uno de los pines del microcontrolador, el cual ha sido configurado como salida, por ejemplo PORTA.0, que se refiere a que el pin 0 del puerto A será por donde se obtenga el tono deseado [1]. La nomenclatura *SOUND Pin,[Note,Duration{,Note,Duration…}]* se refiere a la manera de utilizar la instrucción. *Note* es un valor entre 0 y 255, considerando *Note* 0 para el silencio, *Note* con valores de 1 y hasta 127 son tonos y *Note* con valores de 128 y hasta 255 son ruido blanco .Los tonos y el ruido blanco están en una escala ascendente (p.ej. 1 y 128 son las frecuencias menores, 129 y 266 las mayores). *Note* 1 es aproximadamente 78,74 Hz, así como *Note* 127 es aproximadamente 10000 Hz, en ambos casos considerando el oscilador interno del microcontrolador a 4 MHz. La variable *Duration* puede tomar valores de 0 y hasta 255, determinando el largo de la nota, en

incrementos de 12 ms. SOUND entrega como salida ondas cuadradas con nivel TTL, por lo que el circuito básico recomendado para trabajar con la instrucción SOUND se aprecia en la Figura 1. Obsérvese que el valor del capacitor está calculado acorde a la frecuencia de 4 MHz del oscilador interno del dispositivo PIC16F628A.




```
SOUND PORTB.7,[100,10,50,10]
' Send 2 sounds consecutively toPin7
```

Figura 1: Circuito a construir para utilizar la instrucción SOUND de PicBasic Pro. Imagen extraía de [1].

Como ejemplo práctico, es posible consultar el código implementado por Dr. Herrera Lozada para la Unidad de Aprendizaje de Aplicaciones de Microcontroladores en el IPN CIDETEC, utilizando la interfaz MicroCode Studio en conjunto con el compilador de PicBasic Pro, en el cual se describe la simulación de los sonidos en frecuencia de una escala diatónica la cual se compone de las notas *Do, Re, Mi, Fa, Sol, La* y *Si* [2]. La escala diatónica está compuesta de 5 tonos y 2 semitonos distribuidos como se aprecia en la Figura 2. En este código sencillo se anexó un valor de silencio con la tarea de darle dinámica a las piezas musicales, todo esto implementado en un PIC16F628A.

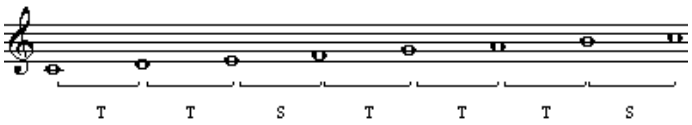


Figura 2: Composición de la escala diatónica.

Otro trabajo interesante en el que se diseñó e implementó una alarma para protección civil utilizando este mismo formato se presenta en [3].

Análisis

Toda secuencia musical está descrita por un comportamiento exponencial, sin embargo existen errores en los cuales el oído humano, que tiene un rango de audición entre los 20 y 20Khz, se deja llevar creyendo escuchar sonidos lineales, es decir, sonidos consecutivos como se muestra en la Figura 3.

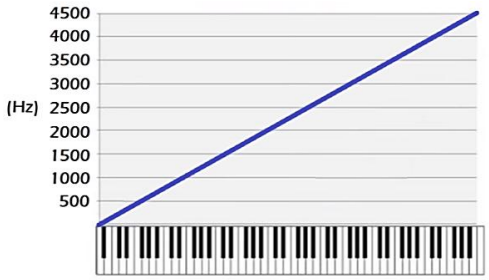


Figura 3: Representación lineal de sonidos. "Error"

Se logra apreciar en la Figura 4, el comportamiento correcto de las frecuencias en aumento. Cabe destacar que los sonidos son apreciados dependiendo de su altura o tono; cuanto mayor sea su frecuencia, más aguda o "alta" será la nota musical, es decir, un sonido agudo o grave "alto o bajo", donde todos estos términos son subjetivos.

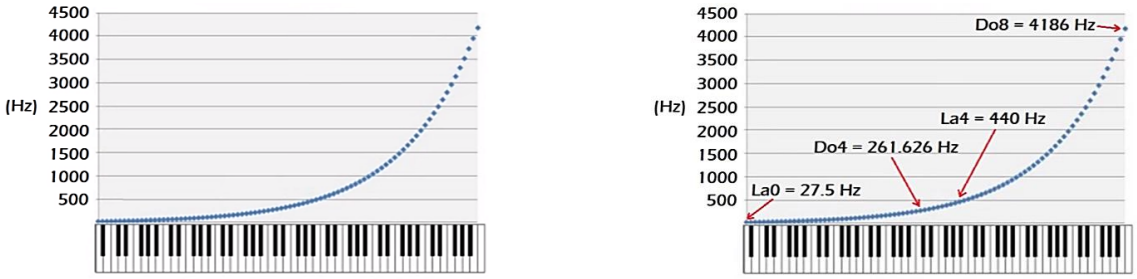


Figura 4: Representación exponencial de sonidos.

Octava

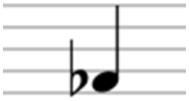
La palabra octava viene de la teoría musical; la octava es el intervalo entre dos sonidos que tienen una relación de frecuencias igual a 1:2 y que corresponde a ocho notas de una escala musical diatónica; o trece en una escala cromática.

Propuesta e implementación en un Microcontrolador PIC16F628A

Con la teoría antes mencionada se propone subdividir las frecuencias, basándose en el código desarrollado en [2], logrando abrir un abanico de posibilidades para trabajar con escalas como la *escala cromática, escala menor, escala pentatónica*, etc. Y así mismo interpretar melodías con estas nuevas frecuencias, tonos o sonidos, pero con una restricción; estas escalas se logran solo en una octava ya que las frecuencias que puede aportar el PIC16F628A son limitadas.

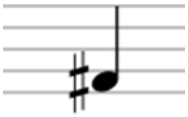
Las subdivisiones nos obligan a tener nuevos conceptos como lo son:

Bemol



Disminuye el tono de una nota por un semitono.

Sostenido



Aumenta el tono de la nota por un semitono.

Se realizaron 3 códigos; el primero con la finalidad de verificar la viabilidad de la propuesta, el segundo fue demostrar hasta qué rangos de frecuencias lograba generar el PIC16F628A y por último la implementación de una melodía (se decidió el Himno a la Alegría) haciendo uso de las subdivisiones.

Primer código:

```
TRISB = 0
CMCON = 7
melodia var byte
num var byte

d  CON 2
d1 CON 8
r  CON 15
rm CON 20
m  CON 27
f  CON 36
f1 CON 42
s  CON 48
s1 CON 53
l  CON 59
l1 CON 64
y  CON 70

b  CON 0

dG  CON 72
d1G CON 78
rG  CON 93
rmG CON 113
mG  CON 140
fG  CON 176
f1G CON 218
sG  CON 266
s1G CON 319
lG  CON 378
l1G CON 442
yG  CON 512

inicio:
FOR melodia=0 to 140

lookup melodia,[d,d1,r ,rm,m,f,f1,s,s1,l,l1,y,b,b,b,y,l1,l,s1,s,f1,f,m,rm,r,d1,d,b,b,b],num

sound PORTB.0, [num, 12]
pause 10
next melodia
goto inicio
end
```

Segundo código:

```
TRISB = 0
CMCON = 7
melodia var byte
num var byte

d  CON 2
d1 CON 8
r  CON 15
rm CON 20
m  CON 27
f  CON 36
f1 CON 42
s  CON 48
s1 CON 53
l  CON 59
l1 CON 64
y  CON 70

b  CON 0

dG  CON 72
d1G CON 78
rG  CON 93
rmG CON 113
mG  CON 140
fG  CON 176
f1G CON 218
sG  CON 266
s1G CON 319
lG  CON 378
l1G CON 442
yG  CON 512

inicio:
FOR melodia=0 to 140

lookup melodia,[d,d1,r ,rm,m,f,f1,s,s1,l,l1,y,b,b,dG,d1G,rG,rmG,mG,fG,f1G,sG,s1G,lG ,l1G,yG],num

sound PORTB.0, [num, 12]
pause 10
next melodia
goto inicio
end
```

La máxima tonalidad que se alcanzo fue *m* en 140, a partir de ese rango en adelante todo fue ruido.

El tercer código que contiene la melodía el *Himno a la Alegría* se presenta de la siguiente forma:

```
TRISB = 0
CMCON = 7
melodia var byte
num var byte

d  CON 2
d1 CON 8
r  CON 15
rm CON 20
m  CON 27
f  CON 36
f1 CON 42
s  CON 48
```

```
s1 CON 53
l  CON 59
l1 CON 64
y  CON 70

b  CON 0

inicio:
FOR melodia=0 to 140

lookup melodia,[f,0,f,0,s,0,l,0,l,0,s,0,f,0,m,0,r,0,r,0,m,0,f,0,f,f,0,m,m,0,0,0,0,_
                f,0,f,0,s,0,l,0,l,0,s,0,f,0,m,0,r,0,r,0,m,0,f,0,m,m,0,r,r,0,0,0,0,_
               m,0,m,0,f,0,r,0,m,0,f,0,s,f,0,r,0,m,0,f,0,s,f,0,m,0,r,0,m,0,l,0],num

sound PORTB.0, [num, 12]
pause 10
next melodia
goto inicio
end
```

Cabe mencionar que en este diseño de implementación de melodía sólo se hizo uso de la escala diatónica.

La construcción física del circuito solo hace uso del PIC16F628A, un altavoz con 4Ω 3W, un capacitor de 10µF, una tabla de prueba o placa fenólica según sea la aplicación, posibilidades o accesibilidad y una fuente de alimentación de 5v.

En las siguientes figuras, Figura 5 y Figura 6, se muestran el circuito finalizado en una tabla de prueba así como la simulación funcional en ISIS de Proteus, respectivamente.

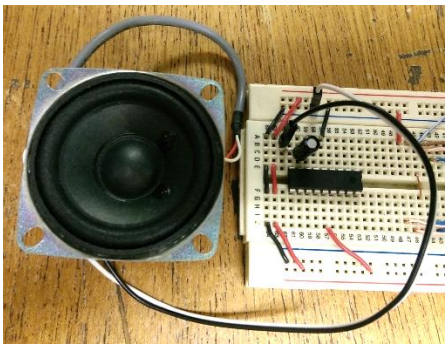


Figura 5: Construcción física.

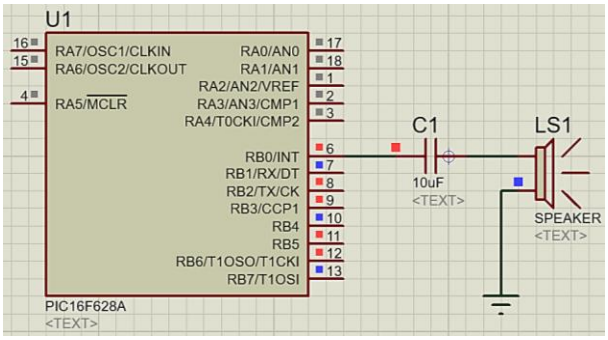


Figura 6: Simulación en ISIS de Proteus.

Conclusión

Se logró tener una gama más alta de tonalidades. Esto nos abre las posibilidades de llevar a cabo la ejecución de una melodía con más juegos tonales, destacando que ahora se conoce más propiedades y limitantes del PIC16F626A. Es posible considerar en diferentes proyectos la teoría aquí expuesta.

Referencias

[1] Micro Engineering Lab, Manual de PicBasic Pro. http://melabs.com/resources/pbpmanual/. Consultado en línea el 29 de abril de 2016.

[2] Herrera – Lozada, J.C., IPN CIDETEC, Aplicaciones de Microcontroladores. http://www.embebidos-cidetec.com.mx/profesores/jcrls/doctos/bocina.zip. Consultado en línea el 29 de abril de 2016.

[3] Márquez-Sánchez C. et Al., Boletín Upiita No. 45, Diseño y Construcción de un Prototipo de una Alarma para Protección Civil con un Microcontrolador. http://www.boletin.upiita.ipn.mx/index.php/ciencia/593-cyt-numero-45/1077-diseno-y-construccion-de-un-prototipo-de-una-alarma-para-proteccion-civil-con-un-microcontrolador. Consultado en línea el 29 de abril de 2016.