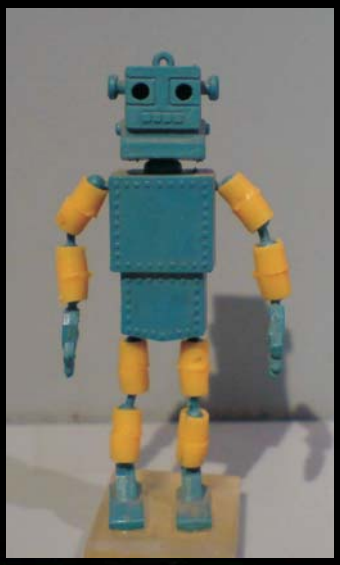
爆風ロボット

WIKI

NUM. 3

Toca la melodía "Pretty woman" con PIC.

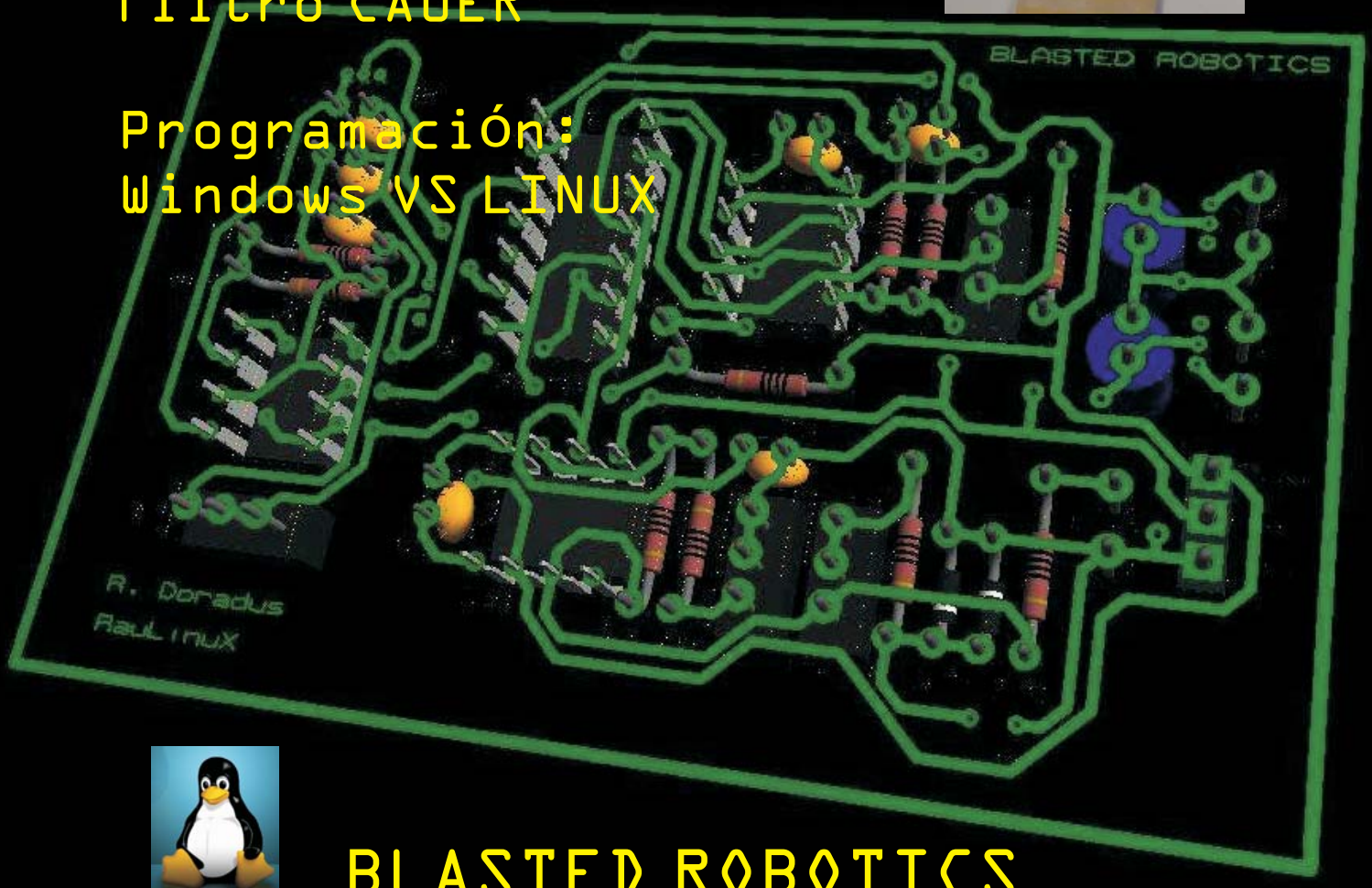
Diseño de un amplificador diferencial con BJT



Interfaz DAC-ADC

Síntesis de un filtro CAUER

Programación: Windows VS LINUX



BLASTED ROBOTICS

Año 1 - Marzo 2009 - Número 3

爆風ロボット

Índice.

Publicación del grupo estudiantil

Blasted Robotics, de la FCE, BUAP, Puebla, Pue. México.

Revista WIKI Número 3.

Marzo de 2009.

e-mail:

blastedrobotics@yahoo.com.mx

La revista tiene como finalidad publicar artículos hechos por la comunidad estudiantil de la Facultad de Ciencias de la Electrónica de la Benemérita Universidad Autónoma de Puebla, para apoyar a nuestros compañeros difundiendo un poco de la creatividad desarrollada y aplicada en el área de la electrónica, el lector podrá construir y probar los diferentes proyectos y prácticas descritas en esta publicación teniendo la certeza de que funcionarán al cien por ciento, todos los artículos han sido diseñados, innovados y probados en los laboratorios de nuestra universidad y otros varios en casa o lugares específicos de diseño como pasatiempo de algunos autores.

En este número 3 de nuestra revista aportamos un poco de los circuitos para armar más útiles en electrónica, comenzando con la topología de un amplificador par diferencial para armar, uno de los que comúnmente nos hacen sufrir un poco siempre que los diseñamos; incluimos una parte de caracterización de diodos de potencia, la síntesis de un filtro elíptico, un circuito divertido para armar y programar en un PIC que se trata de una melodía que es reproducida en un buzzer; una interfaz completa ADC-DAC, una cerradura por código y una comparación de programación entre sistemas operativos.

Esperamos disfruten de los artículos y esperamos que les sea de utilidad, de antemano gracias.

Diseño de un Amplificador Diferencial Acoplado con BJT (El macabro Par Diferencial).

Introducción.

En un amplificador diferencial, por lo común se utiliza un par acoplado por emisor, como el que se muestra en la figura. La corriente de polarización debe ser tal que los transistores funcionen en las regiones activas. El circuito de polarización de cd, que se muestra como una fuente de corriente constante, puede ser o un resistor simple (en cuyo caso el generador equivalente de corriente será cero) o una fuente de corriente con transistor, que en general es lo que se utiliza en circuitos integrados.

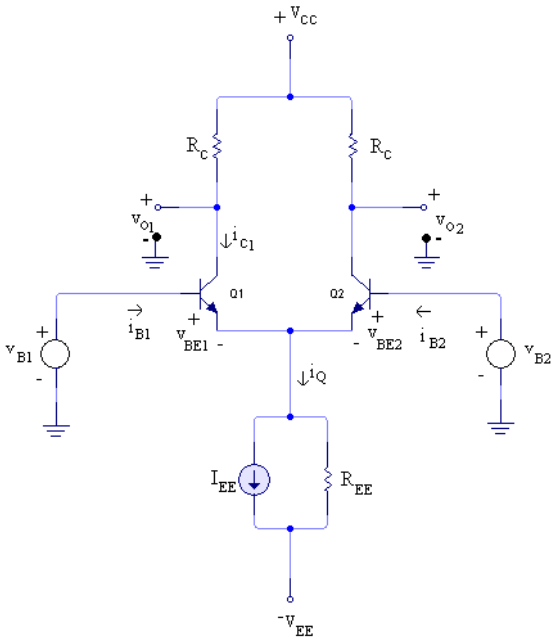


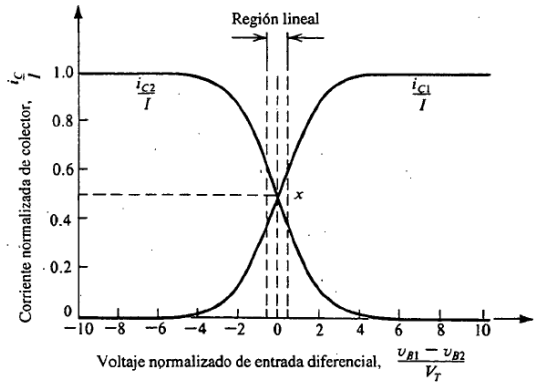
Figura. Par Diferencial acoplado por emisor.

Las características de transferencia de cd, que proporciona la relación entre los voltajes de entrada y salida, se puede obtener a partir del análisis de señal grande, y debe ser lineal en un rango amplio. Se puede simplificar el análisis bajo los supuestos:

1. Las resistencias de salida de los transistores son infinitas: $r_0=\infty$

2. La resistencia de salida de la fuente de corriente con transistor es infinita: $R_E=\infty$

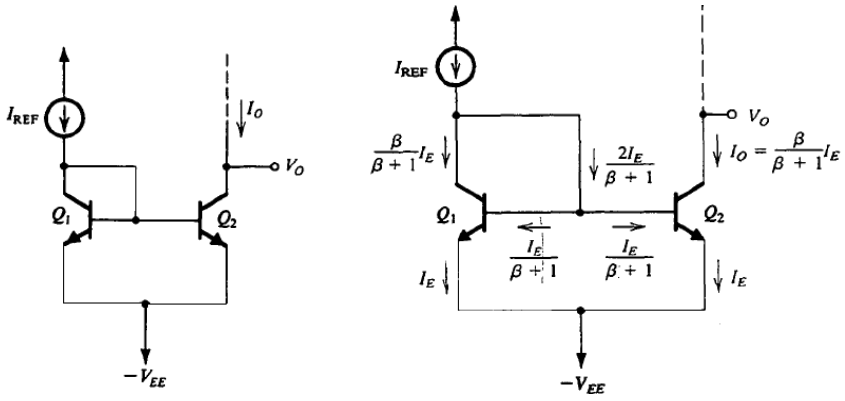
Realizando análisis de Kirchoff a este circuito se llega a la representación en gran señal del par diferencial, pudiendo representar las características de transferencia del par acoplado.



Características de transferencia.

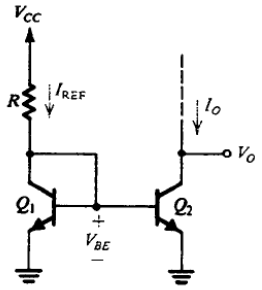
Dando a entender que si i_{c1} aumenta , i_{c2} disminuye, de modo que si $i_{c1}+i_{c2}=\alpha I_Q=\alpha I_E$ se mantiene constante.

Para la construcción de una fuente de corriente podemos usar un Espejo de Corriente hecho con transistores:



Espejo de corriente con BJT y su análisis.

Para la realización de una fuente de corriente podemos tomar el espejo de corriente y proponer el espejo siguiente:



Fuente de corriente sencilla.

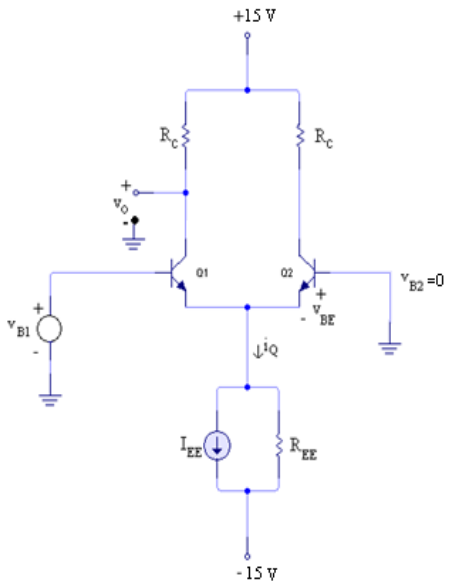
Que responde a la relación siguiente:

$$I_{REF} = \frac{V_{CC} - V_{BE}}{R}$$

Y con esto tendremos una fuente de corriente para nuestro circuito.

Desarrollo.

Para el diseño del amplificador diferencial acoplado por emisor se proponen el siguiente diagrama y los siguientes valores:



$$I_{EE} = 1mA$$
$$V_{CC} = -V_{EE} = 15V$$
$$V_{BE} \approx 0.7V$$
$$V_T = 26mV$$
$$V_A = \infty$$
$$\beta \approx 280$$
$$A_1 = -250$$

Los valores propuestos son para su realización con transistores BC548B.

Entonces analizando el circuito:
Sabemos que:

$$v_{B2} = 0$$

El voltaje de alimentación es simétrico a 15 y -15Volts:

$$R_{EE}I_{EE} + V_{BE} = -V_{EE} = 15V$$

Despejando

$$R_{EE} = \frac{-V_{EE}-V_{BE}}{-I_{EE}} = \frac{15V-0.7V}{1mA} = 14.3K\Omega$$

Luego las corrientes en los colectores son iguales, el análisis puede hacerse separando a la mitad el par y haciendo el equivalente de resistencias en el emisor.

$$i_{C1} = i_{C2} = \frac{I_{EE}}{2} = \frac{1mA}{2} = 0.5mA$$

Obtenemos el valor de la transconductancia:

$$g_m = \frac{i_{C1}}{V_T} = \frac{0.5mA}{26mV} = 19.23\frac{mA}{V}$$

Por la configuración de una terminal aterrizada:

$$v_{B2} = V_{B2} + v_{b2} = 0$$

Entonces

$$v_{b2} = 0$$

Tomando en cuenta las definiciones para los voltajes de entrada:

$$v_{id} = v_{b1} - v_{b2} = v_{b1}$$
$$v_{ic} = \frac{v_{b1}+v_{b2}}{2} = \frac{v_{b1}}{2}$$

Sustituyendo los valores en la definición de voltaje en la salida del par diferencial:

$$v_o = A_d(v_1 - v_2) + A_{cm}(\frac{v_1+v_2}{2})$$
$$v_{o1} = A_d(v_{b1} - v_{b2}) + A_{cm}(\frac{v_{b1}+v_{b2}}{2})$$
$$v_{o1} = A_d v_{b1} + A_c \frac{v_{b1}}{2}$$
$$v_{o1} = \frac{v_{b1}}{2}(2A_d + A_c)$$

Despejando el voltaje de entrada de la ecuación anterior se transforma en la forma de ganancia para este circuito:

$$A_1 = \frac{v_{o1}}{v_{b1}} = \frac{1}{2}(2A_d + A_c)$$

Por definición:

$$A_d = \frac{v_{od}}{v_{id}} = -g_m R_C$$

$$A_c = \frac{v_{oc}}{v_{ic}} = \frac{-\beta R_C}{r_\pi + 2(\beta+1)R_{EE}} = \frac{-g_m R_C}{1+2g_m R_{EE}(1+\frac{1}{\beta})}$$

Y sustituyendo estos valores en la representación de A_1

$$A_1 = \frac{1}{2}\left[2(-g_m R_C) + \left(\frac{-g_m R_C}{1+2g_m R_{EE}(1+\frac{1}{\beta})}\right)\right]$$

$$= -\frac{1}{2}\left[2g_m R_C + \frac{g_m R_C}{1+2g_m R_{EE}(1+\frac{1}{\beta})}\right]$$

$$= -\frac{1}{2}\left[2g_m + \frac{g_m}{1+2g_m R_{EE}(1+\frac{1}{\beta})}\right]R_C$$

Entonces, podemos despejar R_c ya que los demás valores los conocemos.

$$R_C = \frac{-2A_1}{2g_m + \frac{g_m}{1+2g_m R_{EE}(1+\frac{1}{\beta})}}$$

Sustituimos y desarrollamos los cálculos correspondientes:

$$R_C = \frac{-2(-250)}{2(19.23\frac{mA}{V}) + \frac{19.23\frac{mA}{V}}{1+2(19.23\frac{mA}{V})(14.3k\Omega)(1+\frac{1}{280})}}$$

$$= \frac{500}{38.46\frac{mA}{V} + \frac{19.23\frac{mA}{V}}{1+38.46\frac{mA}{V}(14.3k\Omega)(1.003)}} = \frac{500}{38.46\frac{mA}{V} + \frac{19.23\frac{mA}{V}}{552.942\frac{mA}{V}k\Omega}}$$

$$= \frac{500}{38.46+0.03477}k\Omega$$

Finalmente tenemos el valor para R_C:

$$R_C = 12.989k\Omega$$

Para confirmar el diseño se evalúan las definiciones de ganancia en modo diferencial y ganancia en modo común.

$$A_d = -g_m R_C = -19.23\frac{mA}{V}(12.989k\Omega) = -249.78$$

$$A_c = \frac{-g_m R_C}{1+2g_m R_{EE}(1+\frac{1}{\beta})} = \frac{-249.78}{1+2(0.01923)(14300)(1+\frac{1}{280})} = -0.45173$$

Adicionalmente podemos calcular la Razón de Rechazo en Modo Común:

$$CMRR = \left|\frac{A_d}{A_c}\right| = \frac{249.78}{0.45173} = 552.94 = 54.86dB$$

Que en efecto son muy buenos y se aproximan a la ganancia requerida en el diseño.

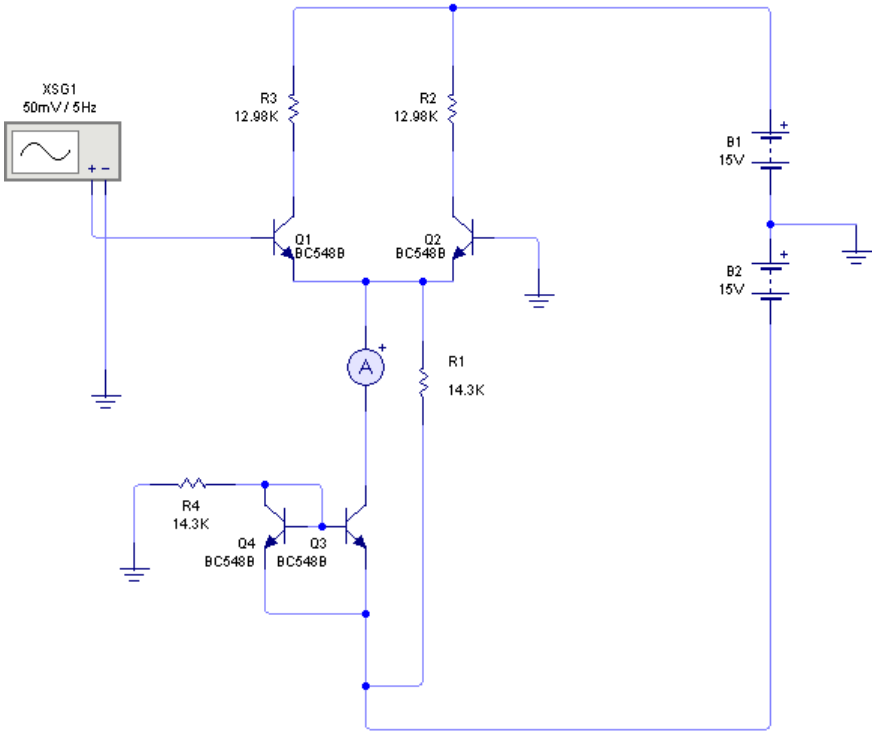
El circuito final para armar requiere de una fuente de corriente de 1mA además de los componentes que calculamos, así que se diseña una fuente de 1mA con un circuito Espejo de corriente con transistores BC548B también.

Para ello utilizamos la fórmula:

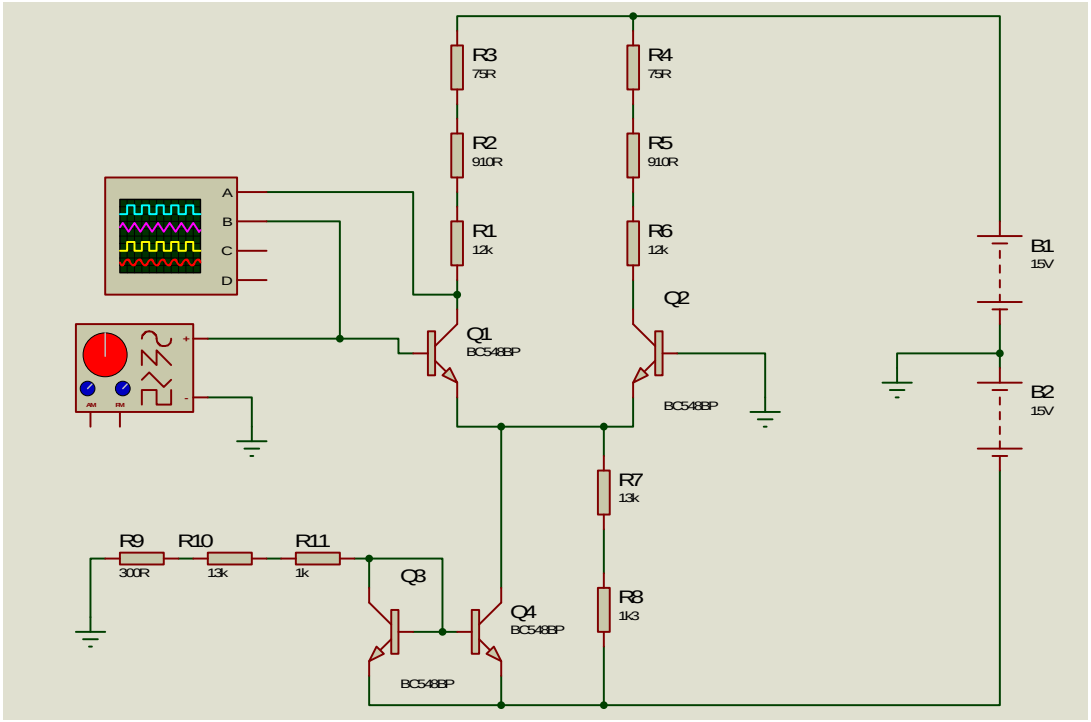
$$I_{REF} = \frac{V_{CC}-V_{BE}}{R}$$

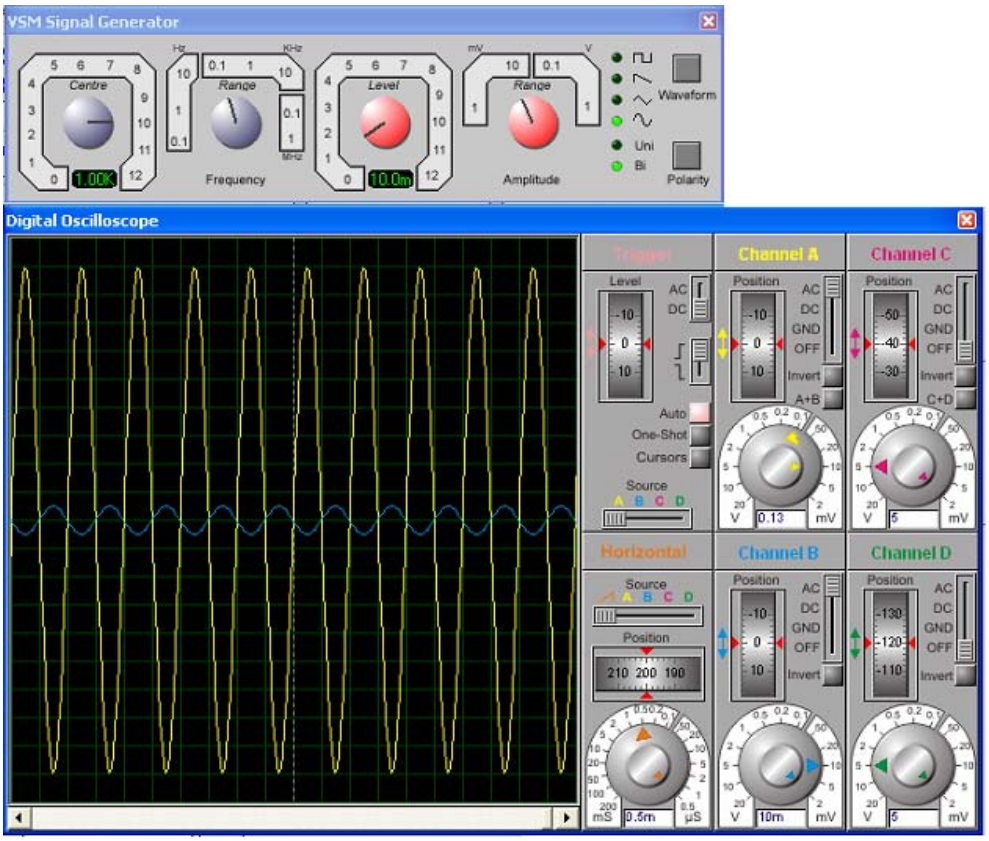
$$I_{REF} = \frac{V_{CC}-V_{BE}}{R}$$
$$1mA = \frac{15V-0.7V}{R}$$
$$R = \frac{15V-0.7V}{1mA} = \frac{14.3V}{1mA} = 14.3k\Omega$$

El circuito final para armar es el siguiente:

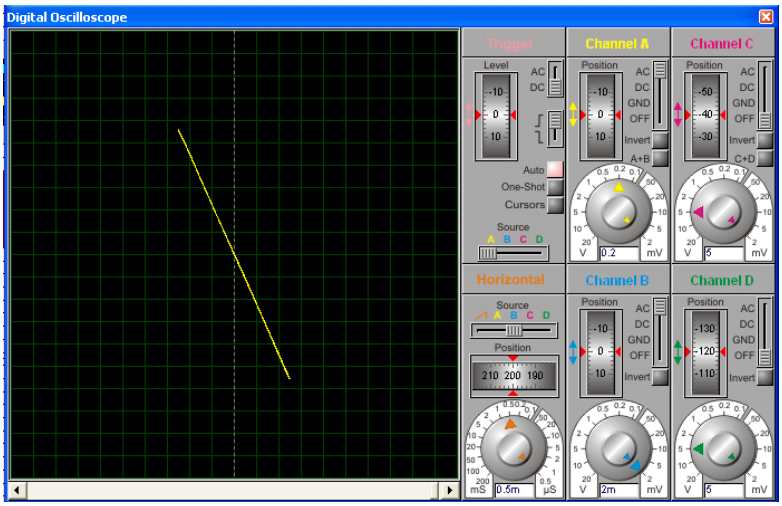


Y la simulación fue hecha en Livewire y Proteus 7.





En la cual se ve claramente la amplificación. O la salida en el formato x-y del osciloscopio:



Donde la señal azul es la entrada de voltaje en la base del transistor Q_1, y la señal amarilla es la salida en el colector del mismo transistor.

Conclusión.

Para realizar diseños de amplificadores diferenciales es necesario conocer muy bien los análisis de los transistores BJT, así como sus modelos internos, las leyes de circuitos, etcétera, ya que sin estos conocimientos se nos complicará mucho el diseño, en cambio haciendo los análisis, simplemente obtenemos las fórmulas o representaciones generales y después solamente sustituimos y calculamos valores de acuerdo a los que necesitemos.

Cabe destacar que para hacer estos análisis hay varios métodos que son muy útiles, no se restringen a uno sólo, en este caso usamos los más convenientes.

El armado del circuito se llevó a cabo con transistores de propósito general BC548B pero es recomendable hacerlo con transistores de propósito de amplificación y que mejor que con transistores encapsulados pareados, ya que con ellos el amplificador funcionará más acorde al diseño. Las hojas de especificaciones en estos casos son muy útiles para tomar los parámetros como V_T o la β de amplificación de acuerdo al transistor que tengamos.

Autor: R.Doradus.

sanpablojuarez@yahoo.com.mx

Bibliografía:

Adel S. Sedra-Kenneth C. Smith. Circuitos Microelectrónicos. 4a Ed. Oxford.

Rashid. Circuitos Microelectrónicos.

Boylestad, Nashelsky. Análisis de Circuitos y Dispositivos Electrónicos.

www.datasheetcatalog.com & www.nteinc.com

Caracterización de Diodos de Potencia.

Objetivo:

En este artículo se caracterizará el comportamiento estático y dinámico de diferentes Diodos de Potencia, comparando los resultados, a su vez, con un Diodo de Señal.

Motivación:

El presente artículo lo realizamos con el fin de comprender el funcionamiento de los diodos de potencia desde el punto de vista experimental y despejar nuestras dudas al hacer la caracterización de un diodo con el osciloscopio.

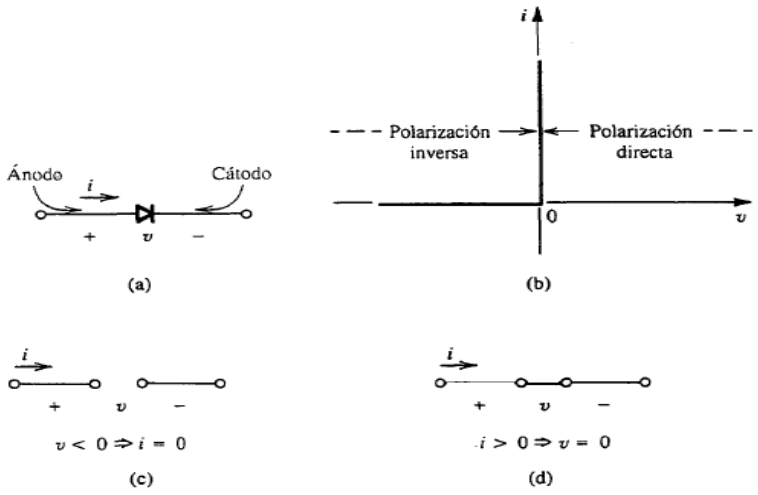
Introducción:

1.-En esta práctica se pretende, en primer lugar, medir la característica estática de los diodos de potencia, obtenida punto a punto en un circuito rectificador simple (R-D). Por su utilidad y sencillez, se comparan los resultados con medidas realizadas en un diodo de señal con diodos de red, potencia y Schottky respectivamente.

2.-En segundo lugar, se mide el comportamiento dinámico de los diferentes diodos ensayados, sometiéndolos a conmutaciones rápidas, resaltando los problemas en el corte asociados a la recuperación inversa.

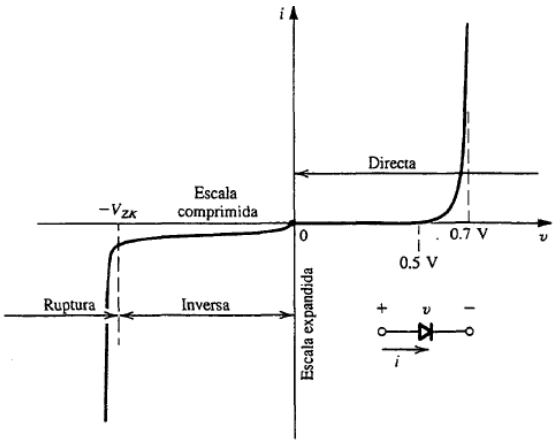
Marco Teórico:

El elemento no lineal mas sencillo y fundamental es el **diodo**. Al igual que el resistor, el diodo tiene dos terminales; pero, a diferencia de aquél que tiene una relación lineal (en línea recta) entre la corriente que circula a través de ese elemento y el voltaje que aparece en sus terminales, el diodo tiene una curva característica *i*-*v* no lineal.



El diodo ideal: (a) símbolo de diodo; (b) curva característica *i*–*v*; (c) circuito equivalente en la dirección inversa; (d) circuito equivalente en la dirección directa.

En la siguiente figura se muestra la curva característica *i-v* de un diodo de unión de Silicio.



La relación *i–v* de un diodo con algunas escalas expandidas y otras comprimidas para dejar ver detalles.

Descripción Práctica:

1. Armar el circuito de la figura 1, con los diodos polarizados directamente.
   Variando la tensión de alimentación Vi de 0 a 10V, en incrementos de 0.5V, anotar en cada caso el punto de trabajo (I_d, V_d) que permitirá trazar la característica estática para cada tipo de diodo. Para el diodo de señal (1N4148) R1 será de 1KΩ, para los demás R1=100Ω.

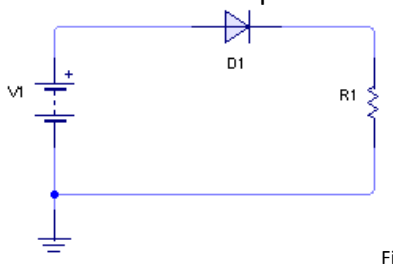


Figura 1.

2. Determinar la resistencia interna equivalente para polarización inversa para cada uno de los diodos midiendo la corriente en inversa para dos tensiones de 10V y 20V. Debido al reducido valor de I_o (corriente inversa de saturación), realizar la medida ensayando con resistencias serie de alto valor (100KΩ, 1MΩ). Anotar también el valor de I_0. Comprobar la variación de I_0 con la temperatura calentando el diodo con un dedo.

3. Utilizando el generador de señal, seleccionar forma de onda cuadrada de amplitud $10V_{pp}$ y montar el circuito de conmutación de la figura 2. Representar las formas de onda de la tensión en la resistencia para los diferentes diodos

para frecuencias de 10 kHz y 100 kHz.

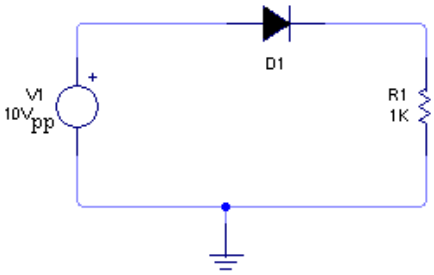


Figura 2.

4. Determinar el tiempo de recuperación inversa de los diferentes diodos en la forma indicada en la figura 3.

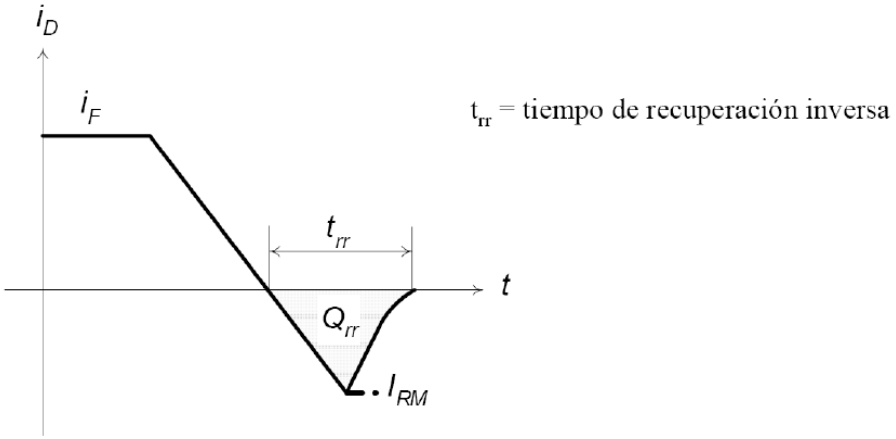


Figura 3. Tiempo de recuperación inversa de los diodos.

5. Reportar los resultados obtenidos en el laboratorio, en concreto, la curva I_d-V_d (característica estática), la resistencia en inversa, la corriente inversa de polarización, las formas de onda de conmutación y el tiempo de recuperación en inversa de cada diodo.

Desarrollo Matemático:

La curva característica del diodo se puede representar mediante modelos matemáticos como lo son la ecuación de Shockley entre otras, tal ecuación liga la intensidad de corriente y la diferencia de potencial:

$$I = I_S\left(e^{\frac{qV_D}{nkT}} - 1\right)$$

Donde:

I es la intensidad de la corriente que atraviesa el diodo

V_D es la diferencia de tensión entre sus extremos.

I_S es la corriente de saturación (aproximadamente $10^{-12} A$)

q es la carga del electrón cuyo valor es $1.6 * 10^{-19}$

T es la temperatura absoluta de la unión

k es la constante de Boltzmann

n es el coeficiente de emisión, dependiente del proceso de fabricación del diodo y que suele adoptar valores entre 1 (para el germanio) y del orden de 2 (para el silicio).

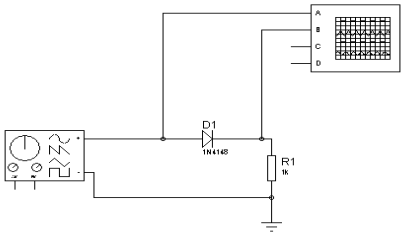
El término $V_T = kT/q = T/11600$ es la tensión debida a la temperatura, del orden de 26 mV a temperatura ambiente (300 K ó 27 ºC).

Con esots datos y esta fórmula se puede graficar la curva característica de un diodo de unión pn, simplemente sustituyendo los valores correspondientes a los que dicta la expresión matemática.

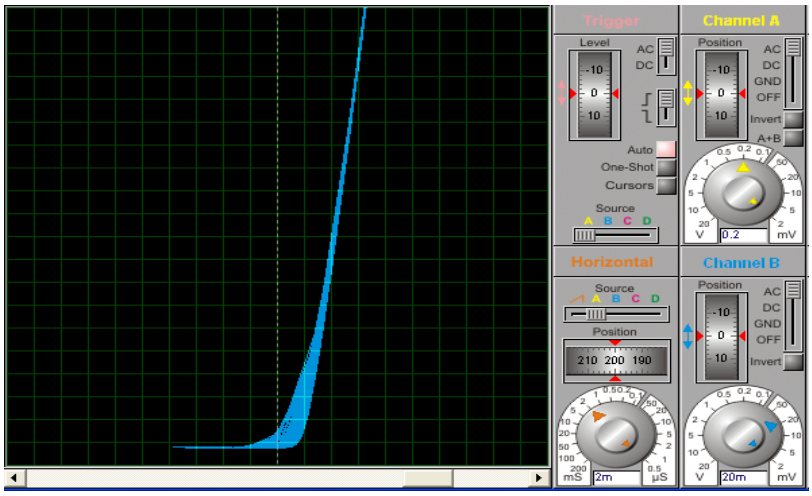
De aquí podemos ver claramente que el comportamiento de la curva característica de corriente versus voltaje o potencial en un diodo será de un crecimiento exponencial dependiente tanto de voltaje entre sus extremos, temperatura en la unión del diodo, corriente de saturación, constantes de termodinámica aplicables al diodo, material del diodo y tipo de fabricación, y el voltaje ocasionado por la temperatura.

Resultados Numéricos y Gráficos:

Simulación.



Esquema Simulado para el 1N4148.



Simulación en Proteus.

Los resultados de la simulación para el diodo 1N4148 con la entrada de señal cuadrada en el modo xy del osciloscopio son una curva característica con voltaje de umbral variable, por ello el grosor de la curva en el simulador, oscilando entre los 0.6 a los 0.8 Volts y eso debido a la frecuencia de conmutación de la señal cuadrada del generador de funciones.

Por tanto en la práctica se esperaban resultados similares a los obtenidos en la simulación, es decir, la caracterización mostrará curvas características de tal forma, pero sin el desvío provocado por la frecuencia de conmutación en el voltaje de umbral.

Práctica.

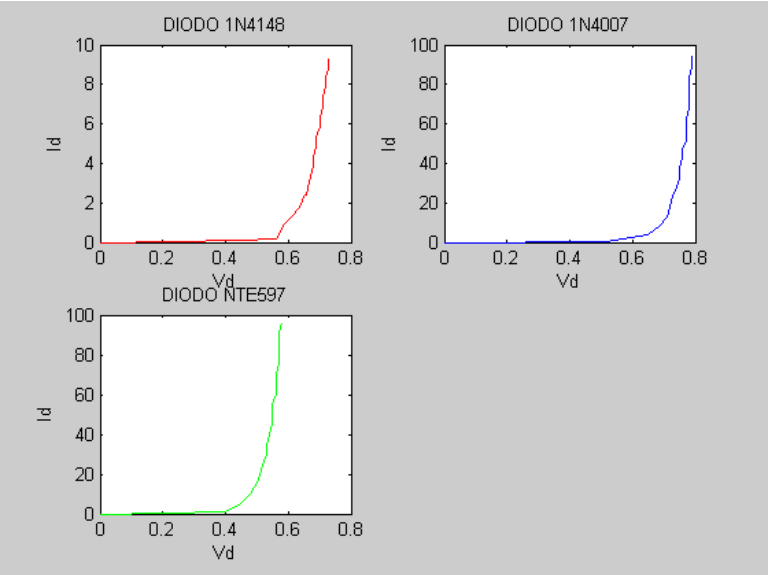
Los resultados medidos se presentan en la siguiente tabla:

	Diodo 1N4148		Diodo 1N4007		Diodo NTE597	
Vcc (V)	Vd (V)	Id (mA)	Vd (V)	Id (mA)	Vd (V)	Id (mA)
0	0	0	0	0	0	0
0.5	0.45	0.09	0.52	0.164	0.4	1.2
1	0.56	0.15	0.65	3.8	0.45	5.4
1.5	0.59	0.94	0.69	8.8	0.48	10.3
2	0.62	1.41	0.71	13.1	0.5	15.5
2.5	0.64	1.92	0.72	17.8	0.51	19.8
3	0.65	2.3	0.73	23.3	0.52	24.7
3.5	0.66	2.42	0.74	27.8	0.53	29.9
4	0.67	3.36	0.75	32.1	0.53	34.8
4.5	0.68	3.83	0.75	37.5	0.54	39.5
5	0.68	4.37	0.76	42.6	0.55	44.9

5.5	0.69	4.83	0.76	47	0.55	50
6	0.69	5.32	0.77	52	0.55	55.1
6.5	0.7	5.88	0.77	58.2	0.56	60.2
7	0.7	6.32	0.77	62.4	0.56	64.8
7.5	0.71	6.82	0.78	68.3	0.56	70.3
8	0.71	7.35	0.78	73.4	0.57	75.5
8.5	0.72	7.95	0.78	78.2	0.57	80.3
9	0.72	8.35	0.78	83.3	0.57	85.7
9.5	0.73	8.81	0.79	88.6	0.57	90.8
10	0.73	9.33	0.79	94.6	0.58	96.4

Los valores obtenidos fueron graficados en Matlab.

```
V=[0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10];
Id1=[0 0.09 0.15 0.94 1.41 1.92 2.30 2.42 3.36 3.83 4.37 4.83 5.32 5.88 6.32 6.82 7.35 7.95 8.35
8.81 9.33];
Vd1=[0 0.45 0.56 0.59 0.62 0.64 0.65 0.66 0.67 0.68 0.68 0.69 0.69 0.70 0.70 0.71 0.71 0.72 0.72
0.73 0.73];
Id2=[0 0.164 3.8 8.8 13.1 17.8 23.3 27.8 32.1 37.5 42.6 47 52 58.2 62.4 68.3 73.4 78.2 83.3 88.6
94.6];
Vd2=[0 .52 .65 .69 .71 .72 .73 .74 .75 .75 .76 .76 .77 .77 .77 .78 .78 .78 .78 .79 .79];
Id3=[0 1.2 5.4 10.3 15.5 19.8 24.7 29.9 34.8 39.5 44.9 50 55.1 60.2 64.8 70.3 75.5 80.3 85.7 90.8
96.4];
Vd3=[0 .4 .45 .48 .5 .51 .52 .53 .53 .54 .55 .55 .55 .56 .56 .56 .57 .57 .57 .57 .58];
subplot(2,2,1)
plot(Vd1,Id1,'r')
title('DIODO 1N4148')
xlabel('Vd')
ylabel('Id')
subplot(2,2,2)
plot(Vd2,Id2)
title('DIODO 1N4007')
xlabel('Vd')
ylabel('Id')
subplot(2,2,3)
plot(Vd3,Id3,'g')
title('DIODO NTE597')
xlabel('Vd')
ylabel('Id')
```



Las graficas de nuestros resultados muestran claramente la curva característica de cada uno de los diodos, mostrando que el comportamiento del voltaje contra la corriente es exponencial.

A comparación con la simulación los resultados se parecen bastante, con la pequeña diferencia de que el voltaje de umbral es más claro entre los 0.6 y 0.7 Volts, para el primer diodo y sin la variación de respuesta a la conmutación en el diodo.

Comparando resultados medidos entre los 3 diodos, tenemos que e diodo de potencia NTE597 presenta un crecimiento más suave y más rápido en comparación a los otros, el diodo 1N4007 presentó el comportamiento más lento al estabilizarse cercano a los 0.8 Volts pero un cambio igualmente suave al aumento de corriente en función del voltaje y el 1N4148 presentando un comportamiento con más pendiente en corriente en la región de voltaje de umbral con más lentitud al cambiar del estado de no conducción a su opuesto, pero éste presenta un menor gasto de corriente en comparación a los otros 2, dando a entender que los otros funcionan un poco más rápido y un poco más idealmente pero con un consumo mayor de energía en comparación con los de señal.

Conclusiones:

La caracterización de los diodos de potencia nos ayuda mucho en el diseño de nuestros sistemas electrónicos, ya que nos da un panorama de qué tan lejos de la realidad nos encontramos en cuanto a

la aplicación y diseño de estos diodos.

Los resultados y valores obtenidos en esta caracterización los podemos relacionar directamente tomando los valores que arroja nuestro dispositivo y comparándolos con los que proporciona el fabricante, se pueden utilizar sin embargo para tener una visión más clara del que será du comportamiento en la práctica, trabajo o aplicación que vayamos a realizar.

El diodo de señal no es para la aplicación en potencia, sin embargo, es de gran ayuda para relacionar directamente su funcionamiento con los diodos de potencia, haciendo la analogía con ellos vemos que son bastante parecidos en cuanto a su funcionamiento. La variación en cuanto a la temperatura es muy mínima, ya que el cambio a los resultados difiere en milivolts.

En la práctica podemos decir que afortunadamente caracterizamos con éxito los diodos a excepción del Schottky en cuanto a su comportamiento estático y dinámico, logrando obtener valores muy parecidos a las de las especificaciones y graficar las curvas características de cada uno de ellos.

Apéndices:

Artículo tomado de un reporte de Potencia 1.

El similar del diodo de potencia BYW29-200 es el NTE597.

La hoja de datos y características se anexa a continuación

(The datasheet of NTE597 is in the next page.)

(Fact Sheet oder Datenblatt ist auf der nächsten Seite)

Bibliografía:

Adel S. Sedra-Kenneth C. Smith. Circuitos Microelectrónicos. 4a Ed. Oxford.

http://woody.us.es/~leopoldo/Ficheros/practica_diodos.pdf

www.datasheetcatalog.com & www.nteinc.com

Síntesis de un filtro Cauer.

R.Doradus.
sanpablojuarez@yahoo.com.mx

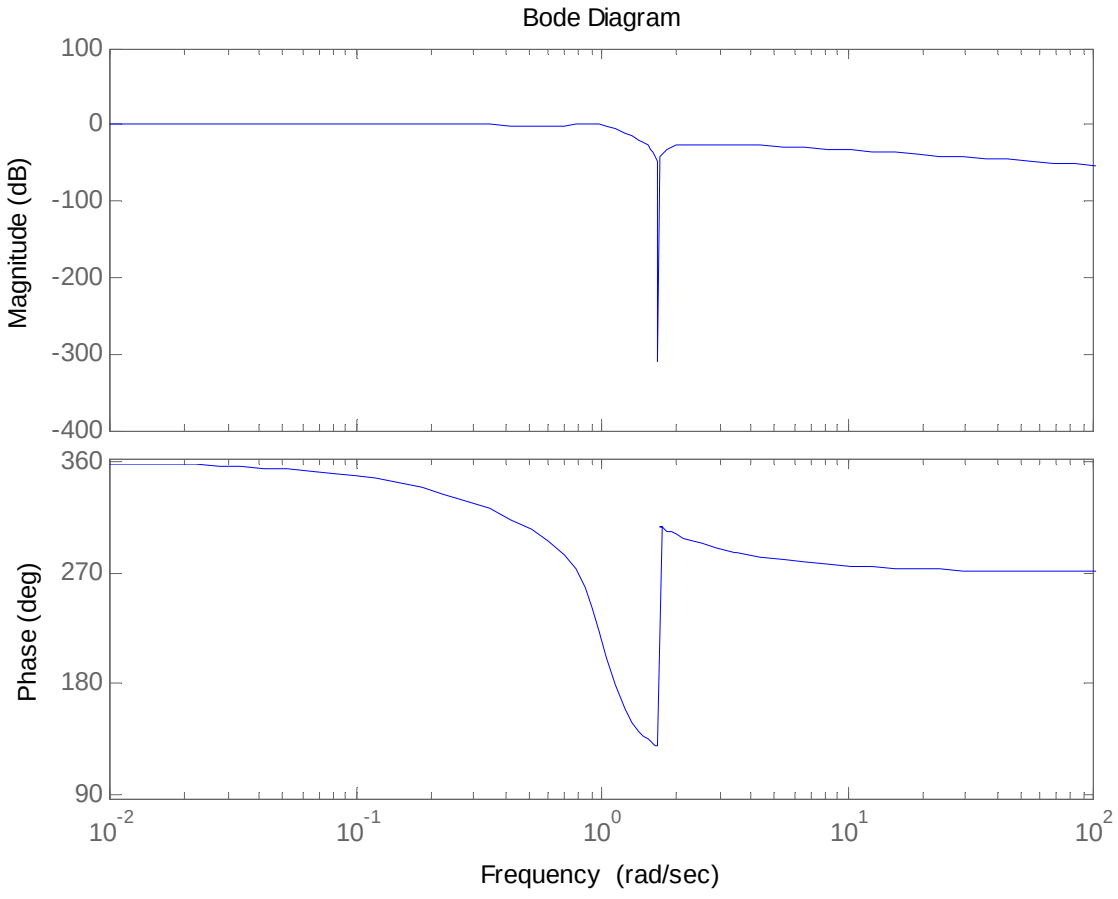
Para el filtro Cauer: ωp=1000rad/s; αmax=1dB; ωs/ωp=1.5; n=3

⇨ αmin=25dB; H=1/4.6378=0.2156.

$$T(s) = \frac{0.2156(s^2+2.806014)}{(s+0.591015)(s^2+0.375396s+1.023714)}$$

Pasándola a matlab:

```
Transfer function:
      0.2156 s^2 + 0.605
-------------------------------
s^3 + 0.965 s^2 + 1.244 s + 0.6
```



Para obtener el circuito se descompone la función de transferencia en dos términos, para luego aplicar el circuito Sallen-Key en configuraciones de pasa altas y pasa bajas, con un filtro de polo simple y todo unido con un sumador.

Para ello:

$$T(s) = \frac{0.2156S^2}{(S+0.591015)(S^2+0.375396S+1.02371)} + \frac{0.605}{(S+0.591015)(S^2+0.375396S+1.02371)}$$

$$T(S) = \frac{1}{(S+0.591015)}\left[\frac{0.2156S^2}{(S^2+0.375396S+1.02371)} + \frac{0.605}{(S^2+0.375396S+1.02371)}\right]$$

$T(S)=T_1(S)\,[T_2(S)+T_3(S)]$

Para $T_1(S)$

$$T_1(S) = \frac{1}{(S+0.591015)}$$

$T_1(S) = -\frac{SC_1+G_1}{SC_2+G_2}$ para el circuito con Opamps de impedancias general, pero $C_1=0$ ya que no hay S en el denominador.

$$T_1(S) = -\frac{G_1/C_2}{S+\frac{G_2}{C_2}} = -\frac{(1/R_1C_1)}{S+(1/R_2C_2)}$$ sin normalizar.

Normalizando:

$$T_1(S) = -\frac{G_1/C_2}{S+\frac{G_2}{C_2}} = -\frac{(1/1000R_1C_1)}{S+(1/1000R_2C_2)}$$

Proponemos C1=C2=100nF.

$$1 = \frac{1}{R_1C_1} = \frac{1}{1000R_1 100*10^{-9}}$$

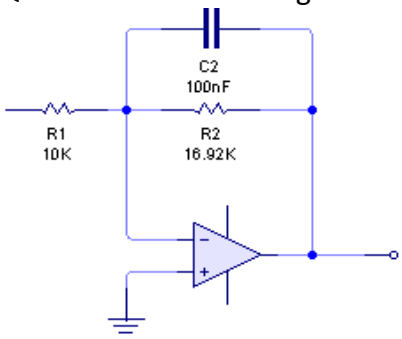
$$R_1 = \frac{1}{1000*100*10^{-9}} = 10\,000 = 10K\Omega$$

$$\frac{1}{1000R_2C_2} = 0.591015$$

$$\frac{1}{1000R_2 100*10^{-9}} = 0.591015$$

$$R_2 = \frac{1}{1000*0.591015*100*10^{-9}} = 16920.0 = 16.9K\Omega$$

Quedando el circuito siguiente:



Para $T_2(S)$

$$T_2(S) = \frac{0.2156S^2}{(S^2+0.375396S+1.02371)}$$

Tiene la forma de un pasa altas, y en este caso es hecho con el circuito Sallen-Key en la configuración de pasa altas.

$$T_2(S) = \frac{kS^2C_1C_2}{G1G2+s[G2(C1+C2)+G1C2(1-k)]+S^2C1C2}$$ sin normalizar.

Elegimos $C_1=C_2$; $R_1=R_2=R_A$:

$$T_2(S) = \frac{kS^2C^2}{G^2+s[G(2C)+GC(1-k)]+S^2C^2} = \frac{kS^2}{S^2+s(\frac{G}{C})(3-k)+(\frac{G}{C})^2}$$

Normalizamos:

$$T_2(S) = \frac{kS^2}{S^2+s(\frac{G}{1000C})(3-k)+(\frac{G}{1000C})^2}$$

$$(\frac{G}{1000C})^2 = 1.023714$$

=> $$\frac{G}{1000C} = \sqrt{1.023714} = 1.0118$$

C=100nF

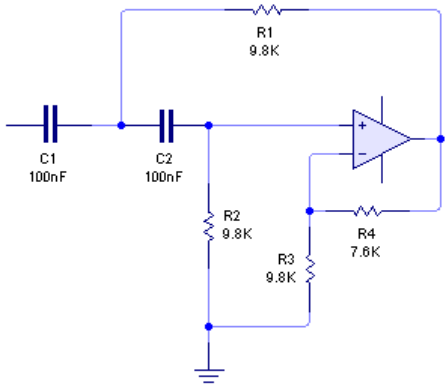
$$\frac{1}{R_1 1000*100*10^{-9}} = 1.0118$$

$$R_1 = \frac{1}{1.0118*1000*100*10^{-9}} = 9883.4 = 9.8K\Omega$$

K=0.2156;

$$R_B = R_A(k-1) = 9.8*10^3*(0.2156-1) = -7687.1$$

Por lo que tomamos el valor de 7.6K



Para el circuito que falta $T_3(S)$

También tomamos el Sallen Key, pero en configuración pasa bajas.

$$T_3(s) = \frac{0.605}{(S^2+0.375396S+1.02371)} = \frac{(\frac{kG1G2}{1000^2C^2})}{S^2+S[\frac{G1+G2(2-k)}{1000C}]+\frac{G1G2}{1000^2C^2}}$$

Tomamos $R_1=R_2=R_A$ y C=100nF:

$$\frac{1}{R^2*1000^2(100*10^{-9})^2} = 1.023$$

$$R^2 = \frac{1}{1.023*1000^2(100*10^{-9})^2} = 9.7752\times10^7$$

$$R = \sqrt{9.7752\times10^7} = 9887.0 = 9.8K\Omega$$

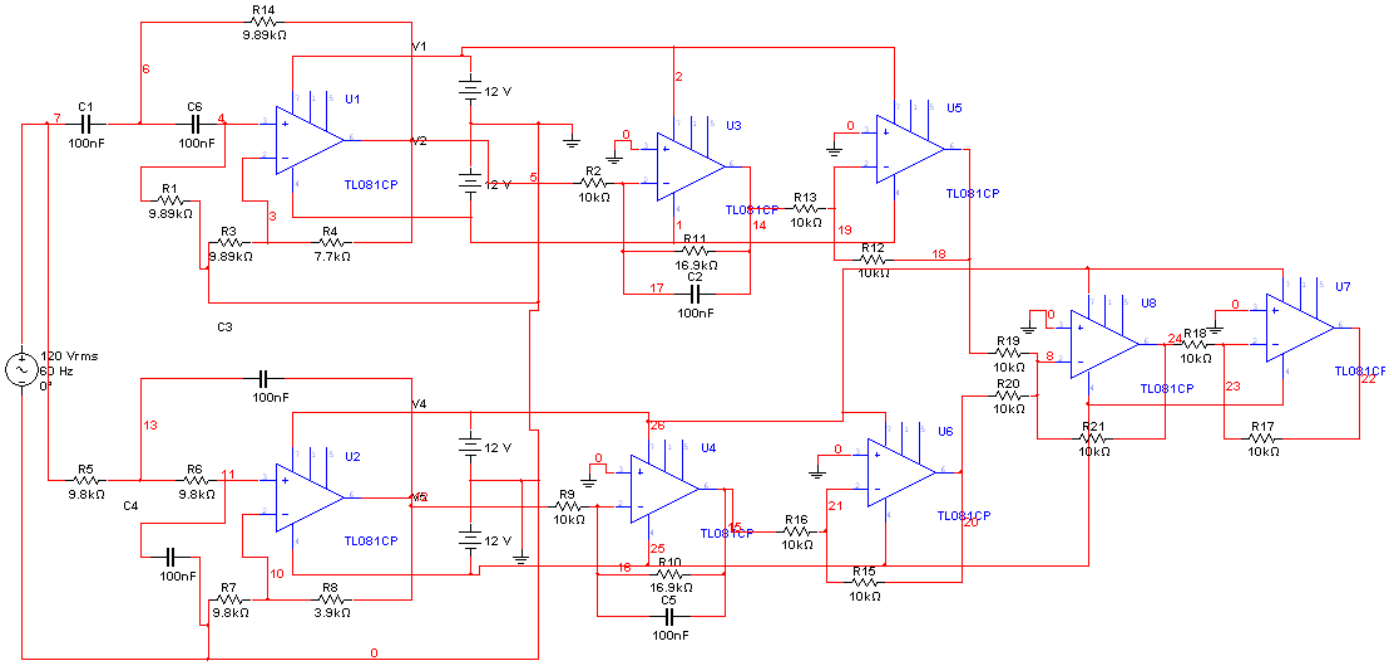
K=0.6049;

$$R_B = R_A(k-1) = 9.8*10^3*(0.6049-1) = -3872.0$$

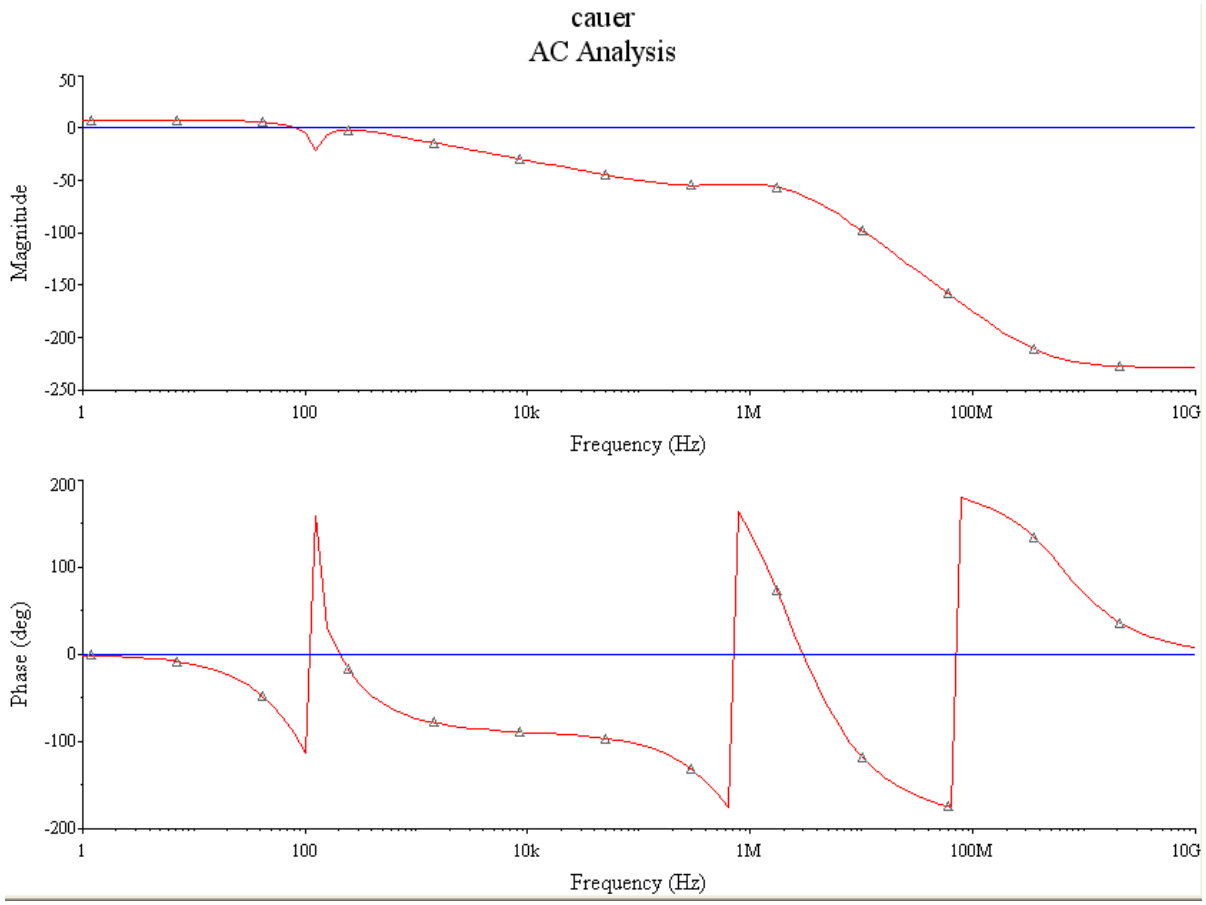
Entonces tomamos el valor de R_B=3.9K

Ahora solo se le agrega un sumador de ganancia unitaria y sus respectivos inversores de ganancia unitaria para invertir los signos.

El circuito resultante es el siguiente:



Y su respectiva simulación:



Toca Pretty Woman con PIC.

La música es uno de los puntos más olvidados en el avance de la tecnología, actualmente los sistemas comerciales de procesado de audio son muy caros, hace años cuando comenzaba la era digital, y los primeros celulares incluían música, veíamos como con una pequeña parte de nuestra canción favorita nos ponía de buen humor, y tomemos en cuenta que se trataba de sonidos MIDI generados por el microcontrolador interno del celular. En este artículo mostramos un concepto básico pero muy bueno para iniciarnos en esta aventura de generación de tonos con microcontroladores, para ello nos apoyamos en los compiladores nuevos para la programación de microcontorladores, en este caso la familia de los PIC, pero este mismo ejercicio lo podemos desarrollar con cualquier otro lenguaje o cualquier otra familia de micros.

Para reproducir un sonido es necesario saber cómo es el procedimiento para hacerlo, por ello primero vamos a ver como hacer esto y luego lo programaremos y armaremos. Si disponemos de una melodía que queremos reproducir, nosotros por experiencia o por ayuda de la red sabremos cómo se toca la melodía analógicamente, es decir, sabemos cómo hacerlo con instrumentos reales, aquí, tomemos de ejemplo la canción de Roy Orbison, Pretty Woman, que la mayoría de nosotros nos sabemos la tonadita. Esta tonada se toca con la guitarra en el intro, y su respectiva partitura tablatura la vemos en la figura PW.1.

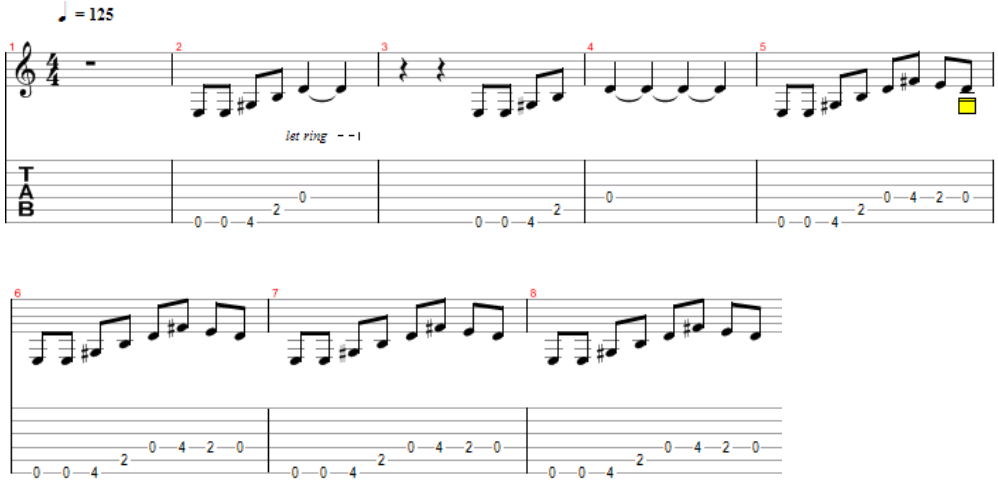


Figura PW.1. Tablatura de la melodía que vamos a reproducir.[i]

Una vez que tenemos ya esta información solo tenemos que hacer que nuestro PIC (o cualquier otro micro) reproduzca el sonido de las frecuencias que corresponden a cada nota musical.

Como usaremos el lenguaje C para programar esta melodía, nos será muy fácil generar notas, ya que existen compiladores que ya traen integradas librerías para melodías MIDI, para este trabajo usamos en útil PICC de la firma CCS, el cual incluye una librería llamada “tones.c” que genera

automáticamente la nota en su frecuencia estándar y un demo llamado EX_TONES.C, que podemos usar para basarnos y generar nuestra melodía. Si usted quisiera ver cómo se hace esto manualmente en c, basta con abrir el código fuente de la librería y podrá ver cómo es el proceso para generar una nota y poder incluso añadir más.

Programando el uC.

Para este diseño utilizamos un PIC de la firma Microchip, de los ya más utilizados, el PIC18F452, ya que es muy rápido y generalmente barato de usos múltiples.

En PICC, hacemos clic en File > New > Source File y creamos un nuevo archivo de código fuente, para después nombrarlo como nos plazca, por ejemplo "rola1.c".

Ahora bien, antes de que se nos olvide, hacemos clic en Project > Create y en Select main source file, buscamos y seleccionamos el archivo que acabamos de crear rola1.c para luego hacer clic en abrir.

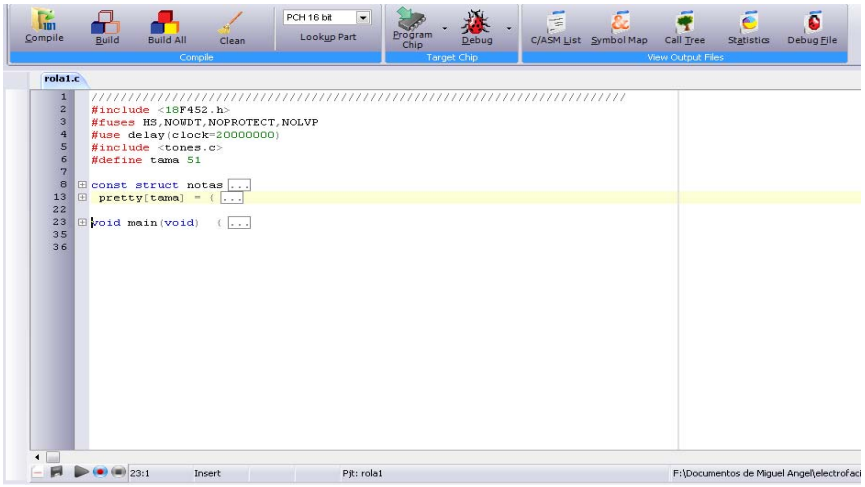


Figura PW.2. Código completo

Etiquetas.

Una vez creado el proyecto ya podemos escribir código y compilarlo.

Primero colocamos las librerías clásicas:

```
#include <18F452.h>
```

```
#include <tones.c>
```

Tener cuidado en colocar esta segunda etiqueta después de la definición del oscilador.

Luego añadimos fusibles de automáticos para grabado y la frecuencia de nuestro oscilador en Mega Hertz:

```
#fuses HS,NOWDT,NOPROTECT,NOLVP
```

```
#use delay(clock=20000000)
```

Incluimos la definición de un identificador que utilizaremos, este es el que indica cuantas notas tocaremos, para este caso tocaremos 51 notas, hasta donde la canción dice: "…on the street.":

```
#define tama 51
```

Luego agregamos una estructura constante la cual crea una colección de variables, en este caso del mismo tipo, agrupadas todas juntas como una sola, con las variables que la librería tones.c necesita para generar las notas:

```
const struct notas
{
   long tono;
   long longitud;
}
```

Para después agregar la tabla que acomodará los valores de dos en dos a la estructura anterior:

```
pretty[tama] = {
E_note[0],125, E_note[0],125, Ab_note[0],125, B_note[0],125, D_note[1],1000,
E_note[0],125, E_note[0],125, Ab_note[0],125, B_note[0],125, D_note[1],1000,
E_note[0],125, E_note[0],125, Ab_note[0],125, B_note[0],125, D_note[1],125, Gb_note[1],125, E_note[1],125, D_note[1],125,
E_note[0],125, E_note[0],125, Ab_note[0],125, B_note[0],125, D_note[1],125, Gb_note[1],125, E_note[1],125, D_note[1],125,
E_note[0],125, E_note[0],125, Ab_note[0],125, B_note[0],125, D_note[1],125, Gb_note[1],125, E_note[1],125, D_note[1],125,
E_note[0],125, E_note[0],125, Ab_note[0],125, B_note[0],125, D_note[1],125, Gb_note[1],125, E_note[1],125, D_note[1],125
Db_note[1],180, Db_note[1],180, B_note[1],180, A_note[1],360,
Db_note[1],180, Db_note[1],180, B_note[1],360, A_note[1],360, Db_note[2],720};
```

Que no son mas que notas que están definidas en la librería de tonos, las cuales van a ser llamadas de dos en dos en la estructura clásica main:

```
void main(void)  {
   int i;
   while(TRUE)
   {
      for(i=0; i<tama; ++i)
      {
         generate_tone(pretty[i].tono,pretty[i].longitud);
         delay_ms(80);  //75 o los que quieras
      }
   }
}
```

Donde la instrucción generate_tone(frequency, duration) genera una onda de salida con frecuencia y duración especificada, aquí esto es el i-ésimo valor de la estructura, que es tomado de dos en dos de la tabla.

El programa escrito completamente lo podemos ver en la figura PW.2

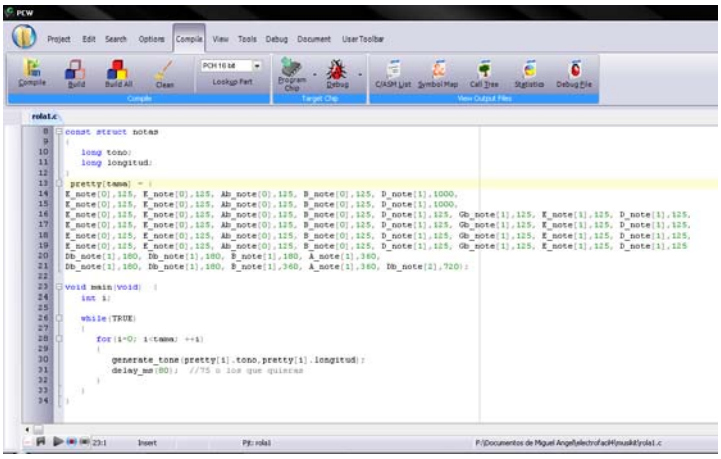


Figura PW.3. Código completo desglosado.

Una vez hecho nuestro código procedemos a compilarlo, hacemos clic en compilar y listo, aparecerán algunos warnings explicándonos que no hemos usado todas las notas que incluye la librería pero ningún error.

Se genera el código hexadecimal y ya podemos grabar el PIC y probarlo conectando un Buzzer o un Sounder barato con la segunda terminal directa a tierra como se muestra en la figura PW.4.

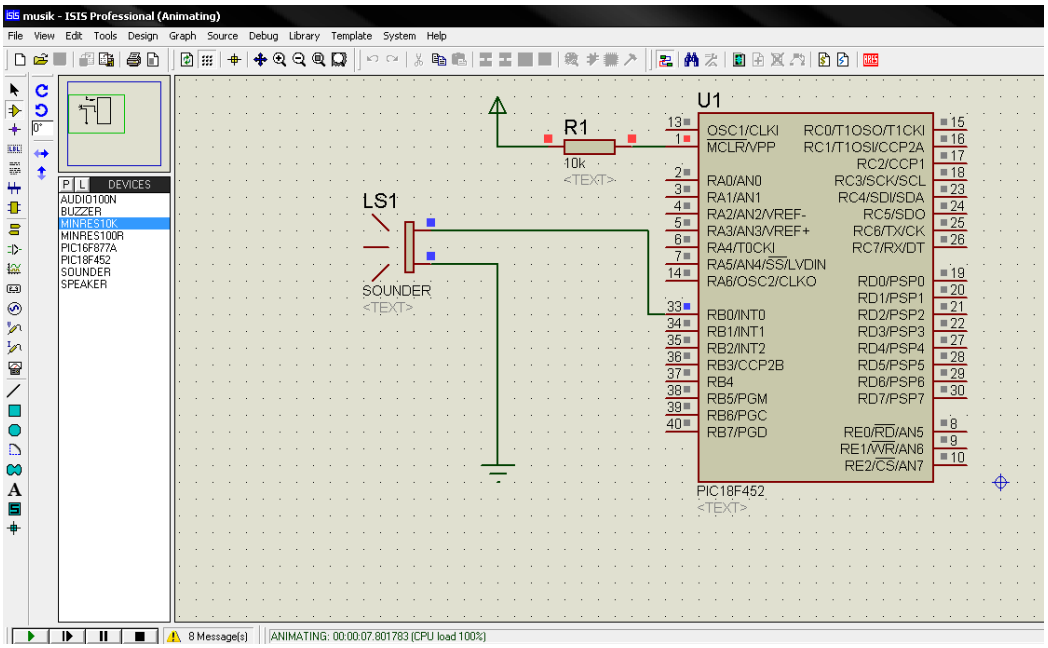


Figura PW.4. Circuito Simulado en Proteus y Armado físicamente.

Nota: La salida es el PIN_B0 del PIC, ya que es como está definido en la librería, pero si tenemos un poquito de experiencia podemos modificar esta librería sin problemas y si no tenemos experiencia pues es necesario hacer pruebas de cambios con su respectivo respaldo de información para no perder librerías.

Autor: R.DORADUS.

Dudas: sanpablojuarez@yahoo.com.mx

[i] Tomado de http://cifraclub.terra.com.br/cifras/roy-orbison/pretty-woman-zhmk.html

Convertidor Analógico Digital (ADC) a partir de un Digital Analógico (DAC)

Gutiérrez Gutiérrez Arizbeth

Objetivo

Construir un convertidor analógico digital (ADC) de 8 bits, a partir de implementar un DAC (convertidor digital analógico).

Introducción

Esta práctica trata de los circuitos de conversión entre valores digitales y valores analógicos. Primero, se presenta el circuito de conversión digital a analógico con la implementación física del mismo. Segundo, se trata el convertidor ADC con un enfoque parecido al DAC.

Muchas variables físicas son de naturaleza analógica y pueden tomar cualquier valor dentro de un rango continúo de estos. Por ejemplo la temperatura, la presión, las señales de audio y la intensidad luminosa entre otras. La tensión de salida de un amplificador de audio hacia los altavoces, esta tensión es una cantidad analógica porque cada uno de sus posibles valores produce una respuesta diferente en el altavoz. Por otro lado, una cantidad digital tiene un valor que se especifica por una de dos posibilidades, como un 0 o 1, un alto o un bajo.

Por otro lado, con los circuitos analógicos es difícil procesar o realizar operaciones numéricas con las tensiones que representan las variables físicas. En cambio, con los circuitos digitales es más fácil procesar o realizar operaciones con los valores representativos de variables físicas o señales, ya que se tienen únicamente dos posibles valores, 0 y 1.

Es por esto, que se crearon los ADC conjuntamente con los DAC, para poder manejar el procesamiento de señales digitalmente. Ambos convertidores sirven para ajustar los sistemas analógicos con los sistemas digitales y viceversa. Los ADCs y los DACs funcionan como interface entre un sistema totalmente digital como lo es una computadora y el mundo analógico.

Marco Teórico

Convertidor Digital Analógico

El DAC acepta una palabra digital de n bits y produce una muestra analógica. En la siguiente figura se presenta el símbolo de un convertidor Digital - Analógico, cada entrada digital puede ser sólo un "0" o un "1". 1 es el bit menos significativo (LSB) y N es el más significativo (MSB). El voltaje de salida analógica tendrá uno de valores posibles dados por una de las combinaciones de la entrada digital.

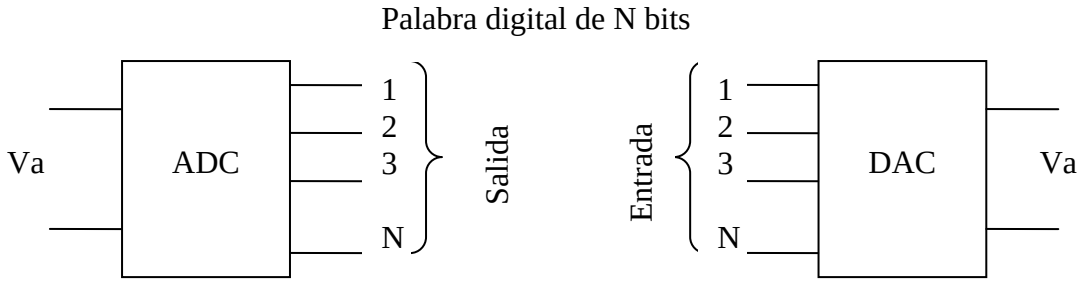


Figura 1.1. Los convertidores A/D y D/A como bloques de circuito**.**

La resolución se define de dos maneras:

Primero se define el número máximo de bits de salida (la salida digital). Este dato permite determinar el número máximo de combinaciones en la salida digital. Este número máximo está dado por: 2^n donde n es el número de bits. También la resolución se entiende como el voltaje necesario (señal analógica) para lograr que en la salida (señal digital) haya un cambio del bit menos significativo (LSB).

$$\Delta V = \frac{V_{MAX} - V_{MIN}}{2^n - 1}$$

Donde: $V_{MAX} - V_{MIN}$ = voltaje de entrada del DAC para la conversión máxima (todas las entradas son "1")

La conversión digital analógico se utiliza como base para muchos métodos de conversión analógico digital. El DAC es el proceso de tomar un valor representado por un código digital ya sea binario, código BCD, y convertirlo en una tensión o corriente que sea proporcional a ese valor digital.

Las entradas digitales se derivan generalmente del registro de salida de un sistema digital. Mediante el incremento del número de bits de entrada se puede reducir la diferencia entre dos valores consecutivos permitiendo aumentar el número de salidas diferentes. Esto nos permitirá producir una salida con saltos más pequeños y cada vez más similar a una cantidad analógica que varía de manera continúa sobre un rango de valores.

El ADC acepta una muestra analógica Va y produce una palabra de n bits como se pudo ver en la figura 1.1. Existen varios tipos de convertidores Analógicos Digitales.

Realimentados:

1. Escalera
2. Seguimiento
3. Aproximaciones Sucesivas

Integradores:

1. Una Rampa
2. Doble Rampa
3. Tensión-Frecuencia

Paralelo:

1. Flash o de Destello

Planteamiento

La práctica tiene dos partes:

1. Tras montar el convertidor para introducir el código binario, en este caso se trabajara con 8 bits. Corregir el error máximo que presenta el voltaje de salida.

2. En la segunda parte se procederá a implementar el ADC a partir de la primera parte y de otros circuitos.

Convertidor Analógico Digital (DAC).

Factores de ponderación de entrada. El DAC, cada entrada digital contribuye con una cantidad diferente a la salida analógica: esto se puede apreciar si se examina los casos donde la entada es alta.

A las contribuciones de cada entrada digital se les asignan factores de ponderación según su posición en el número binario. Para el b_1, que es el bit menos significativo (LBS), tiene un factor de ponderación de 0.019 V. Para b_8, el valor de ponderación es de 5 V para el bit más significativo (MSB).

Los factores de ponderación se duplican así sucesivamente para cada bit comenzando con el LSB. El V_{SAL} para cualquier valor digital de entrada como la suma de los factores de ponderación de las entradas digitales.

En la práctica se probo de forma diferente, comenzando con el MSB, como se muestra en la tabla 1 una vez encendidos todos nuestros bits tomamos el voltaje de salida sucesivamente conforme se apagaba cada bit.

$$V_{sal} = -\frac{5VR'}{R}\left(\frac{b_n}{2} + \frac{b_{n-1}}{4} + \frac{b_{n-2}}{8} + \ldots + \frac{b_2}{2^{n-2}} + \frac{b_1}{2^{n-1}}\right)$$

Donde $R' = -\frac{V_{SMIN}R}{5V}\frac{2^n}{2^{n-1}}$; $R_f = \frac{R'}{2}$

2^7R	2^6R	2^5R	2^4R	2^3R	2^2R	2^1R	2^0R	V_{SAL}
1	1	1	1	1	1	1	1	5V
0	1	1	1	1	1	1	1	2.5V
0	0	1	1	1	1	1	1	1.25V
0	0	0	1	1	1	1	1	0.625V
0	0	0	1	1	1	1	1	0.312V
0	0	0	0	1	1	1	1	0.156V
0	0	0	0	0	1	1	1	0.078V
0	0	0	0	0	0	1	1	0.039V
0	0	0	0	0	0	0	1	0.019V
0	0	0	0	0	0	0	0	0.009V

Tabla 1. Valores de Ponderación

Como se ve en la figura 1.2. Como la etapa sumadora es inversora, se obtendrá un voltaje negativo, que se puede transformar positiva con una segunda etapa el amplificador inversor como se muestra.

Los voltajes de los dígitos de un número binario no ofrecen adecuada precisión (salidas de los circuitos digitales). Para evitar perdida de voltaje y aumentar la precisión del conversor hacemos uso de los transistores para mejorar el error, siendo un voltaje de referencia de alta precisión que se conectan a las resistencias sumadoras.

Cada interruptor se conecta hacia la entrada del amplificador cuando el valor del digito correspondiente es 1, en otro caso, se conecta directamente a 0 V.

Convertidor Analógico Digital (ADC)

La utilización del DAC considerado anteriormente permite realizar la conversión inversa, basado en la comparación entre la señal a digitalizar y la proporcionada por el DAC; un circuito secuencial deberá generar los números binarios cuyo voltaje analógico es comparada con el voltaje a convertir, de forma que finaliza en el momento en que ambos voltajes son iguales.

En la figura 1.2 se muestra un convertidor simple que emplea un comparador, un contador, un DAC, un reloj circuito con un contador, reloj, comparador, latch para implementar el ADC.

El comparador proporciona una salida, positivo cuando la señal de entrada de diferencia es positiva, negativa cuando la señal de entrada de diferencia es negativa.

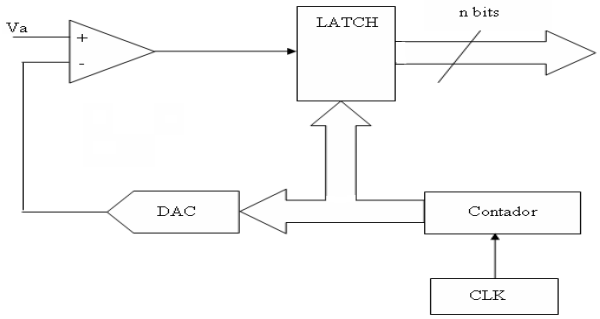


Figura 1.2 Circuito ADC

Cuando se le aplica al contador los pulsos de reloj, pasa por sus estados, se observa que la salida del DAC es una forma de onda de escalera que cambia un volt por cada paso. La salida del DAC alcanza su máximo valor de 255 V, cuando el contador esta en estado 11111111. Cuando el contador vuelve al estado cero, la salida del DAC regresa a 0 V.

Una vez armado esta parte se implementa el comparador y el match de forma que cuando el voltaje analógico y la salida del DAC al compararse sean iguales

se active el latch y deje pasar los n bits.

Desarrollo Experimental

Como observamos tenemos que empezar por diseñar el DAC para poder implementar el Convertidor Analógico Digital (ADC) con el circuito visto anteriormente.

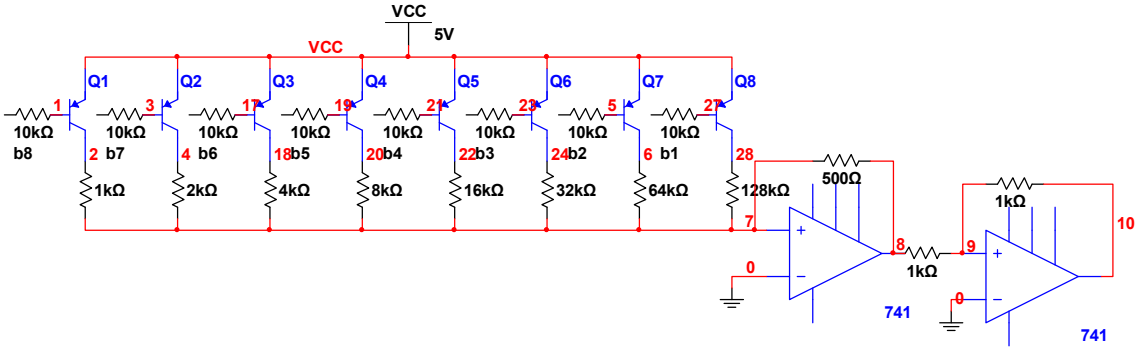
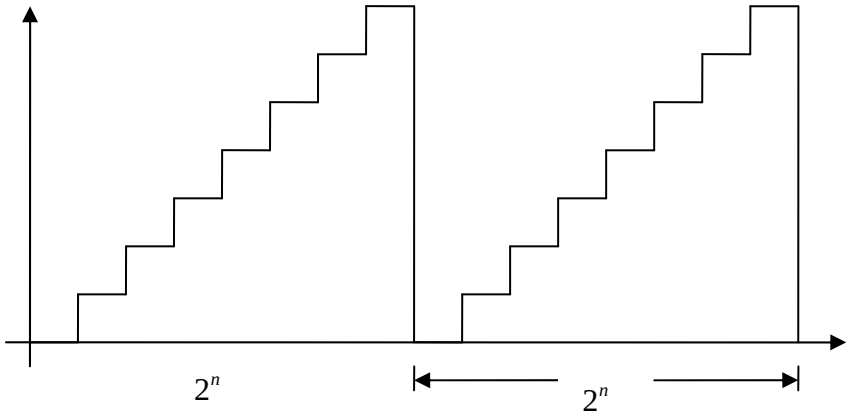


Figura 1.3. DAC implementado.

Para optimizar el rendimiento, se agrega transistores en la entrada digital. La Resistencia limita la corriente que fluye por el dispositivo digital, tomando una mínima corriente suficiente para poner en saturación al transistor.

Con este arreglo, en la salida analógica se tiene variaciones de corriente, por ello es necesario convertir estas variaciones de corriente, a variaciones de voltaje para que las podamos interpretar de una manera mucho más sencilla, utilizando un OPAM, utilizándolo como sumador inversor (Figura 1.3).

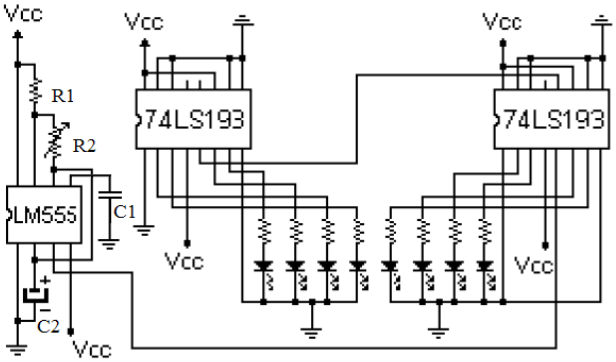


Ahora pasamos al segundo paso armando el diagrama del ADC (Figura 1.2). Si comparamos una señal analógica (este debe estar en el rango del voltaje ajustados a la señal escalera) con la señal escalera, con esto podemos ver cuando nuestras dos señales son iguales.

Para cuando la escalera supere el voltaje de entrada, el comparador cambiara de estado bajo a un alto, indicando que la señal analógica y la escalera son iguales. Para saber el valor digital, colocamos un Latch en las salidas del contador y la salida del comparador para

capturar el dato que tiene nuestro contador en el instante que el comparador cambia de estado.

El contador de 8 bits se implemento con dos contadores de 4 bits y con el reloj como se muestra en la siguiente figura.



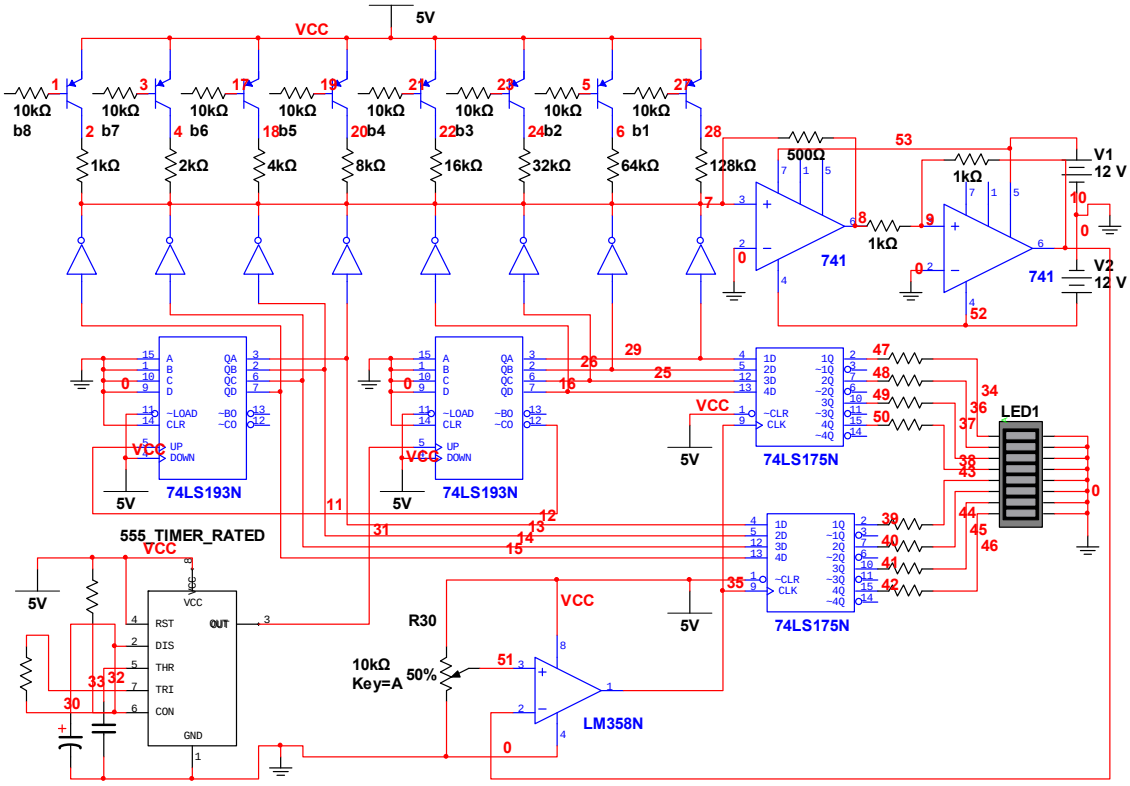


Diagrama eléctrico del Convertidor Analógico Digital.

Resultados

Para el DAC la resolución tuvo pequeñas variaciones, esto debido a que no se utilizaron resistencias de precisión.

Conclusiones

Como se dijo anteriormente que un DAC tiene factores de ponderación el cual varía si las resistencias no son totalmente del valor calculado, es debido a la resolución por lo tanto el valor es lo más cercano al valor exacto.

Referencias

Circuitos Microelectrónicos, Sedra, A., Smith, K., Oxford University Press.

Cerradura por código.

Esta es una sencilla cerradura con un código simple, económico, compacto y confiable.

Como base para este circuito se utilizara el circuito CD4017 que es un contador, así como también 4 compuertas AND.

En primer lugar tenemos la salida 0, en la terminal 3 de nuestro 4017, a nivel alto la cual se conecta al interruptor S1 (NA), y a la resistencia R1 que mantiene a nivel bajo dicha puerta.

A	B	SALIDA
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 1.

De acuerdo a la tabla 1, tenemos la salida de esta puerta a nivel 0, por lo que el diodo D1 queda polarizado inversamente y no conduce. Por esta razón, la terminal 14 (clock) del 4017 no recibe ningún pulso y la salida 0 queda en el nivel alto como en el principio.

En la figura 1 se muestra el circuito eléctrico de la cerradura. Al oprimir el interruptor S1, pasamos a nivel alto uno de las terminales de entrada de la primera puerta del 7408, que anteriormente estaba en nivel bajo. De acuerdo a la tabla 1, la salida de esta puerta cambia a nivel alto polarizando directamente al diodo D1, que conduce enviando un pulso ala entrada clock del CD4017.

Este pulso produce el cambio del nivel alto de la salida 1.

Aquí se produce nuevamente el proceso descrito, y así sucesivamente hasta que el nivel alto llega a la salida 4, terminal 10 del CD4017. Aquí termina el ciclo después de haber presionado cuatro teclas, lasque conforman el código previamente seleccionado. Esta ultima terminal esta conectada por medio de una resistencia a la base de un transistor BC548 que acciona un relé.

El relé puede activar una especie de electroimán para así abrir la puerta o tal vez una alarma.

Lista de materiales

CL1 - 78L05

CL2 - 7408

CL3 – CD4017

D1 a D4 – 1N4148 diodos de uso general.

D5 – 1N4007 diodo rectificador

Q1 –BC548 transistor NPN.

S1 a S6 pulsadores normalmente abiertos.

R1 a R4 2.2K Ω resistors a 1/8W

R5 5.6K Ω 1/8W

C1 10 µF capacitor electrolítico.

K relé de 9V a 12V de bobina.

Autor: Luisor Alexitimico.

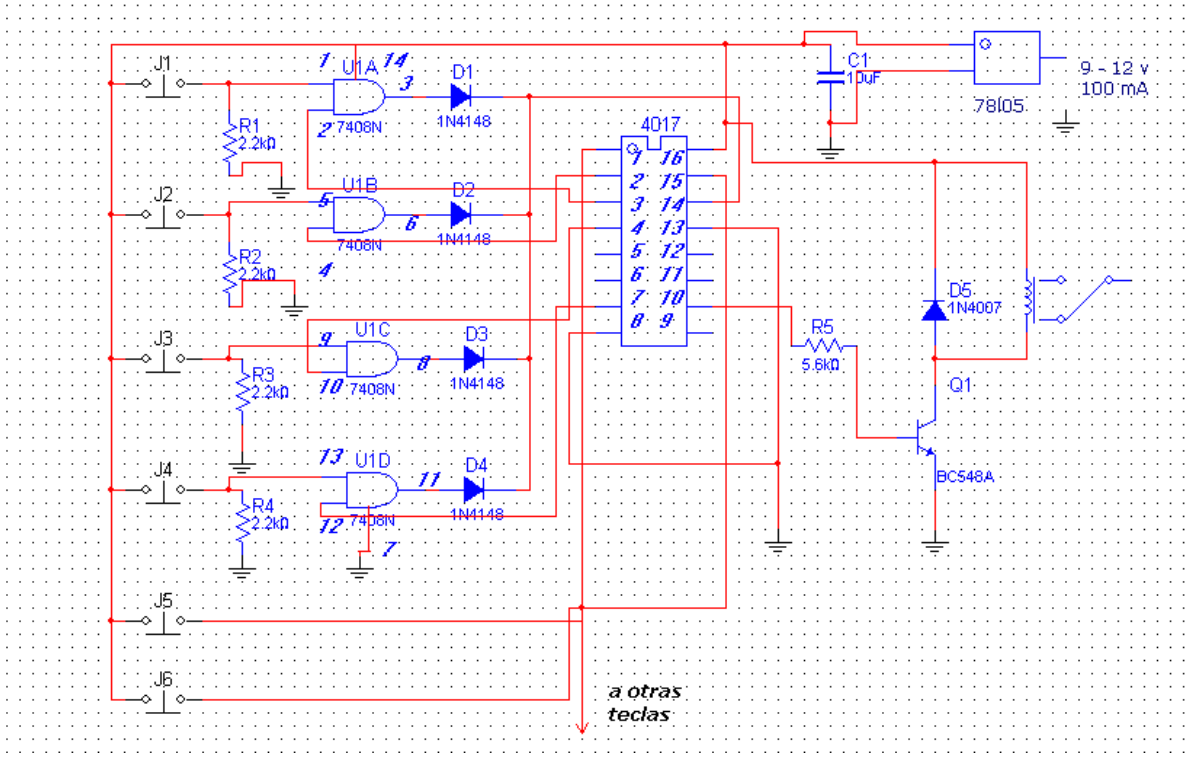
La primera terminal del 4017, salida 5 esta conectada a la terminal 15 que es nuestro reset del mismo integrado, para así poder la orden de que se termino el tecleo del código.

La utilización de un contador obliga a tener que presionar una tecla a la vez y en el orden correcto, dándonos una única manera de teclear el código.

Las demás teclas, ubicadas todas en paralelo están conectadas a la entrada reset del 4017. Es decir que al oprimir una tecla equivocada accidentalmente, el circuito regresa a 0, obligándonos a comenzar de nuevo.

Queda por aclarar que al terminar de introducir el código y polarizar el transistor que acciona el relé, este queda activado debido a que no hay más pulsos en la entrada del clock del 4017 y la salida correspondiente no cambia su nivel. Para desactivar el relé solo bastara presionar alguna tecla que no corresponda al código, para resetear el 4017, regresándolo nuevamente a 0.

Figura 1



Programación: Windows VS LINUX

Historia de C

El lenguaje de programación C fue creado por Dennis Ritchie entre 1969 y 1973 cuando trabajaba en Bell Laboratories de AT&T junto con Ken Thompson en el diseño del sistema operativo UNIX. C fue creado para poder escribir dicho sistema operativo en un lenguaje de alto nivel, independiente del hardware donde se ejecutara.

C se basa en el lenguaje de programación B, escrito por Ken Thompson en cuya mejora colaboró Dennis Ritchie, y B se basa en el lenguaje de programación BCPL escrito por Martin Richards.

Evolución de C

A mediados de los años 60s, Martin Richards diseña el lenguaje BCPL con la finalidad de usarlo para escribir software de sistemas operativos y compiladores.

En 1969, Ken Thompson escribe el Lenguaje B, en Bell Laboratories, con el objetivo de recodificar UNIX (escrito hasta ese momento en lenguaje ensamblador) usando un lenguaje de alto nivel más portable y flexible.

En 1972, Dennis Ritchie modifica el lenguaje B, creando el lenguaje C y reescribiendo el sistema UNIX en dicho lenguaje; añade características nuevas: diseño de tipos y estructuras de datos.

En 1983 ANSI (American National Standards Institute) estandariza C.

En 1983/84, "C con Clases", lenguaje C++. C++ queda disponible en 1985, Creado por Bjarne Stroustrup (en Bell Laboratories).

Fundamentos de programación

Un programa es un conjunto de instrucciones que le dicen a la máquina cómo hacer algo. Sólo hay una manera de hacerlo: escribir una lista de instrucciones una detrás de otra, que se ejecutarán por orden. Algunas de esas instrucciones pueden hacer que la máquina pase a una instrucción que no sea la siguiente, tal vez porque se cumpla una condición que hayamos establecido. A esto se le llama "Programación Imperativa".

Pero este modelo no es lo más cercano a la forma de pensar de un humano, en los últimos años ha tomado fuerza otro

paradigma de computación, llamada "Programación Orientada a Objetos - Object Oriented Programming (en inglés)", en la cual se intentan modelar los sistemas creados como extensiones de la realidad mediante la definición de “objetos” que modelan entidades de la vida real y que interactúan entre sí mediante acciones llamadas métodos o procedimientos.

C es un lenguaje imperativo, no orientado a objetos.

Por otro lado, los lenguajes de programación se clasifican en niveles. Un lenguaje es de más bajo nivel cuanto más cercano esté al código máquina, y un lenguaje de alto nivel es cuanto más lejano esté de la máquina y más cercano al lenguaje humano.

Hay una forma más de clasificar a los lenguajes de programación que es según la forma en que se ejecutan sus órdenes. Existen los lenguajes que son interpretados, cuyas órdenes pasan a través de un intérprete que se encarga de ejecutarlas (a partir del código fuente) en ese momento en que están siendo ejecutadas. Algunos de los lenguajes interpretados son Python, Perl y Tcl. La contraparte de los lenguajes interpretados son los lenguajes compilados (como el mismo C) que se diferencian en que las órdenes son transformadas a lenguaje de máquina y en una etapa final son enlazadas. Los lenguajes compilados tienen la ventaja de la velocidad y la eficiencia, pero los interpretados tienen la ventaja de que, generalmente, son muy portables y de más alto nivel. También existe un lenguaje hecho por Sun, conocido como Java. Los programas escritos en Java pueden o bien compilarse, o bien interpretarse.

C es un lenguaje de alto nivel aunque tiene muchas características de lenguaje de bajo nivel (como el uso que permite hacer de la memoria). Estas características hacen que C sea un lenguaje muy potente, ya que permite optimizar al máximo los recursos de la máquina. Por ende, esto también hace que la dificultad y que los errores que se puedan cometer programando aumenten. Así que a C se le considera de nivel medio.

Turbo C

Turbo C es un entorno de desarrollo integrado y compilador desarrollado por Borland para programar en lenguaje C.

Su primera versión es de 1987, a la que siguieron las versiones 1.5 y 2.0, de 1989

Fue el compilador más popular para desarrollar en C en entornos MS-DOS. Se le considera el primer IDE para C disponible para dicha plataforma.

Fue sustituido por Turbo C++ en 1990.

Éste lo fue, a su vez, por el Borland C++, disponible también para Windows. Tras el Borland C++ llegó el C++Builder.

Tanto el Turbo C 2.0 como el Turbo C++ 1.0 pueden conseguirse gratuitamente en la web de Borland desde el año 2000.

En septiembre de 2006, Borland lanzó una versión recortada del C++Builder para Windows, con el nombre de Turbo C++ for Windows, recuperando así la clásica denominación. Dicho TurboC++ está disponible en dos ediciones: una gratuita, Explorer, y otra de pago, la Pro.

Ambos productos, junto a los otros IDEs de Borland, pasaron a la nueva filial, CodeGear, al ser creada ésta, en noviembre de 2006.

Filosofía de C

C es un lenguaje de programación relativamente minimalista. Uno de los objetivos de diseño de este lenguaje fue que sólo fueran necesarias unas pocas instrucciones en lenguaje máquina para traducir cada elemento del lenguaje, sin que hiciera falta un soporte intenso en tiempo de ejecución. Es muy posible escribir C a bajo nivel de abstracción; de hecho, C se usó como intermediario entre diferentes lenguajes.

En parte a causa de ser de relativamente bajo nivel y de tener un modesto conjunto de características, se pueden desarrollar compiladores de C fácilmente. En consecuencia, el lenguaje C está disponible en un amplio abanico de plataformas (seguramente más que cualquier otro lenguaje).

Características de C

C tiene las siguientes características de importancia: Un núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemá-ticas y de manejo de ficheros, proporcionadas por bibliotecas.

- Un sistema de tipos que impide operaciones sin sentido.
- Usa un lenguaje de preprocesado, el preprocesador de C, para tareas como definir macros e incluir múltiples ficheros de código fuente.
- Acceso a memoria de bajo nivel mediante el uso de punteros.
- Interrupciones al procesador con uniones.
- Un conjunto reducido de palabras clave.

- Punteros a funciones y variables estáticas, que permiten una forma rudimentaria de encapsulado y polimorfismo.
- Tipos de datos agregados (struct) que permiten que datos relacionados se combinen y se manipulen como un todo (en una única variable).

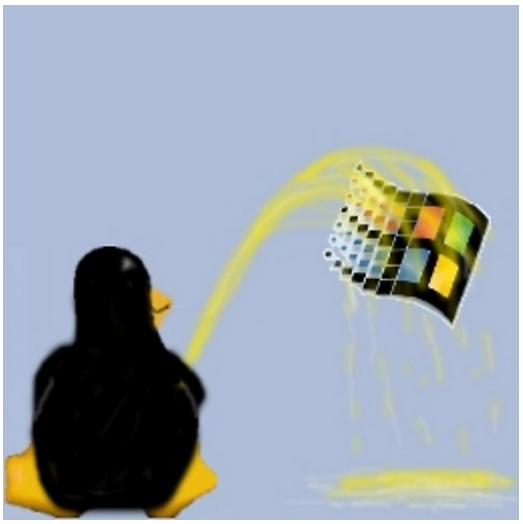
Algunas características de las que C carece que se encuentran en otros lenguajes:

- Recolección de basura.
- Soporte para programación orientada a objetos, aunque la implementación original de C++ fue un preprocesador que traducía código fuente de C++ a C.
- Encapsulación.
- Funciones anidadas, aunque GCC tiene esta característica como extensión.
- Polimorfismo en tiempo de código en forma de sobrecarga, sobrecarga de operadores y sólo dispone de un soporte rudimentario para la programación genérica.
- Soporte nativo para programación multihilo y redes de computadores.

Aunque la lista de las características útiles de las que carece C es larga, este factor ha sido importante para su aceptación, porque escribir rápidamente nuevos compiladores para nuevas plataformas, mantiene lo que realmente hace el programa bajo el control directo del programador, y permite imple-mentar la solución más natural para cada plataforma. Ésta es la causa de que a menudo C sea más eficiente que otros lenguajes. Típicamente, sólo la programación cuidadosa en lenguaje ensamblador produce un código más rápido, pues da control total sobre la máquina.

Programación en C de Windows VS Programación en C de UNIX.

El mito de una programación de difícil acceso se ha difundido por generaciones enteras, pero el hecho de adentrarte en ambas plataformas, desmiente esos rumores. Una de las principales dificultades de adaptarse a los sistemas UNIX, radica en el hecho de suponer que todo es totalmente distinto y tendremos que aprender algo desde su base, pero en realidad hemos sido presas de un temor mal infundado, por que el hecho es claro y la historia no nos dejará mentir, las versiones de C fueron creadas bajo la plataforma UNIX, lo único que nosotros hemos hecho, es usar las versiones adaptadas para la plataforma de Microsoft, es decir las distintas aplicaciones de este programa, usado en varias generaciones del famosísimo Windows y, claro, adaptarnos a las modificaciones y actualizaciones que llegan a sufrir en algún momento determinado.



Uno de los principales objetivos de este artículo es desmentir aquellos rumores, que han hecho que nosotros tengamos temor aun del simple hecho de instalarlo, obviamente el usarlo es un tabú aún en nuestros días, pero quienes ya hemos dado el primer paso, con certeza te podemos decir que es una muy buena opción.

Aunque la primera vez no sepamos nada sobre la consola de comandos de UNIX, veremos que programar es prácticamente igual que como lo venimos haciendo en Windows. Es necesario señalar que si encontraremos diferencias, en el uso de librerías, en su aplicación, en el mismo nombre de instrucciones, en ocasiones varían, pero hacen lo mismo. Pero estas diferencias no deben ser obstáculo para nosotros para no poder, o dejar de utilizar una herramienta tan poderosa como lo es el C.

A continuación dejo unos pequeños ejemplos para que los puedan hacer, uno es para Windows, el otro es para UNIX; aunque muy sencillo, pero fácil de entender.

Reloj analógico. Este puede ser compilado en el Turbo C, o sea que necesariamente deberá estar instalado en su computadora. Posteriormente pueden crear su ejecutable.

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
#include <graphics.h>
#include <dos.h>
#define PI 3.1416
#define VUELT 180

int x2, y2, i, control, modo, minuto, hora, segundo, minutero, segundero, horas;
int xIZQ=0, ySUP=0, xDER, yINF;
float xh, yh, x, xm, ym, xs, ys, xi, yi, xa, yb, alarma, alarma1, alarma2;
char a;
void alar (void);
void reloj (void);
void relojd (void);
void puntos (void);
void instrucciones (void);
struct reloj
{
int hora;
int minuto;
int segundo;
}
```

```
tiempo, *apu_tiempo;
main ()
{
clrscr ();
int control, modo, x1=0, y1=0, x2, y2, indice, x;
control = DETECT;
modo = DETECT;
void modgraf (void);
initgraph (&control, &modo, “C:\\TC\\BGI”);
x2 = getmaxx ();
y2 = getmaxy ();
alarma2 = 15;
cleardevice ();
setbkcolor (BLACK);
apu_tiempo = &tiempo;
instrucciones ();
do
{
alar ();
if (alarma==tiempo.hora&&alarma1==tiempo.minuto&alarma2>tiempo.segundo)
delay (0);
else
delay (1000);
tiempo.segundo++;
if (tiempo.segundo==60)
{
tiempo.minuto++;
tiempo.segundo=0;
}
if (tiempo.minuto==60)
{
tiempo.hora++;
tiempo.minuto=0;
if (tiempo.hora==24)
{
tiempo.hora=0;
}
}
reloj ();
puntos ();
textcolor (14);
relojd();
while (tiempo.minuto<58);
getch ();
return 0;
}
void instrucciones (void)
{
do
{
cout<<”Ajuste su reloj \n\n\t Recuerde que su reloj es de 24 horas.”;
cout<<”\n\n\t Escriba la hora de inicio del reloj : “;
cin>>apu_tiempo->hora;
cout<<”\n\n\t Escriba los minutos de inicio del reloj : “;
cin>>apu_tiempo->minuto;
cout<<”\n\n\t Escriba la hora de sonar la alarma : “;
cin>>alarma;
cout<<”\n\n\t Escriba los minutos de la alarma : ”;
cin>>alarma1;
if
(tiempo.hora>24||tiempo.hora<0||tiempo.minuto<0||tiempo.minuto>60||alarma<0||alarma>24||
alarma1>60||alarma1<0)
{
cout<<”Cometió un error”;
clrscr ();
}
cleardevice ();
}
while(tiempo.hora>24||tiempo.hora<0||tiempo.minuto<0||tiempo.minuto>59||alarma<0||alarma
>24||alarma1>60||alarma1<0);
}
void alar (void)
{
if (alarma==tiempo.hora&&alarma1==tiempo.minuto&&alarma2>tiempo.segundo)
{
sound (100);
delay (1000);
```

```
nosound ();
}
}
void reloj (void)
{
x2 = getmaxx ();
y2 = getmaxy ();
cleardevice ();
horas = ((30*apu_tiempo->hora+apu_tiempo->minuto/2)+90);
minutero = ((6*apu_tiempo->minuto)+90);
segundero = ((6*apu_tiempo->segundo)+90);
setcolor (14);
circle (x2/2, y2/2, 200);
circle (x2/2, y2/2, 180);
xh = .6*VUELT*cos(horas*PI/VUELT);
yh = .6*VUELT*sen(horas*PI/VUELT);
setcolor (10);
line (x2/2, y2/2, x2/2-xh, y2/2-yh);
xm = .85*VUELT*cos(minutero*PI/VUELT);
ym = .85*VUELT*sin(minutero*PI/VUELT);
setcolor (10);
line (x2/2, y2/2, x2/2-xm, y2/2-ym);
xs = .95*VUELT*cos(segundero*PI/VUELT);
ys = .95*VUELT*sin(segundero*PI/VUELT);
setcolor (10);
line (x2/2, y2/2, x2/2-xs, y2/2-ys);
}
void puntos (void)
{
x2 = getmaxx ();
y2 = getmaxy ();
for (x = 0; x < 60; x += 5)
{
xa = .9*180*cos(((6*x)+90*PI/VUELT);
yb = .9*180*sin(((6*x)+90*PI/VUELT);
setcolor (14);
outtextxy ((x2/2-xa), (y2/2-yb), "{ÛÛ");
}
line (x2/2-35, y2/2+55, x2/2+50, y2/2+55);
line (x2/2-35, y2/2+90, x2/2+50, y2/2+90);
}
void reloj (void)
{
gotoxy (38, 20);
printf ("%2.2d:%2.2d:%2.2d", apu_tiempo->hora, apu_tiempo->minuto, apu_tiempo->segundo);
}
```

Calculadora básica. Para ser ejecutada en la terminal.

```
// Para todos aquellos que comienzan en esto.
// Primero abren una terminal
// Después abren el editor vi y lo guardan con terminacion (.sh)
// Para abrirla pones el comando siguiente en la terminal
// (sh nombre_puesto.sh) sin parentesis.
// Se aceptan opiniones para mejorarla


// Código:

trap './menu' 2
while :
do
echo "****************************************"
echo " R a u L i n u x "
echo "****************************************"
echo
echo
echo
echo " menu "
echo "********************************************"
echo "* *"
echo "* *"
echo "* a) Suma *"
echo "* b) Resta *"
```

```
echo "* c) Multiplicación *"
echo "* d) División *"
echo "* e) Hacer operaciones simultáneas (+, -, *, /) *"
echo "* f) Salir *"
echo "* *"
echo "*****************************************"
echo "selecciona una opción"
read ELECCION
case $ELECCION in

a|A)
echo " Introduce tu primer valor"
read valor1
echo " Introduce tu segundo valor"
read valor2
echo " Introduce tu tercer valor"
read valor3
resultado=`expr $valor1 "+" $valor2 "+" $valor3`
echo El resultado de la suma de $valor1 + $valor2 + $valor3 es = $resultado;;
b|B)
echo " Introduce tu primer valor"
read valor1
echo " Introduce tu segundo valor"
read valor2
echo " Introduce tu tercer valor"
read valor3
resultado =`expr $valor1 "-" $valor2 "-" $valor3`
echo El resultado de la suma de $valor1 - $valor2 - $valor3 es = $resultado;;
c|C)
echo " Introduce tu primer valor"
read valor1
echo " Introduce tu segundo valor"
read valor2
echo " Introduce tu tercer valor"
read valor3
resultado=`expr $valor1 "*" $valor2 "*" $valor3`
echo El resultado de la suma de $valor1 x $valor2 x $valor3 es = $resultado;;
d|D)
echo " Introduce tu primer valor"
read valor1
echo " Introduce tu segundo valor"
read valor2
resultado=`expr $valor1 "/" $valor2`
echo El resultado de $valor1 / $valor2 es = $resultado;;
e|E)
echo "Introduce tu primer valor:"
read valor1
echo "Introduce el operador, (+,-,*,/)"
read ope
echo " Introduce tu segundo valor:"
read valor2
echo "Introduce el operador, (+,-,*,/)"
read ope
echo " Introduce tu tercer valor"
read valor3
echo " Introduce el operador, (+,-,*,/)"
read ope
echo " Introduce tu cuarto valor"
read valor4
resultado=`expr $valor1 "$ope" $valor2 "$ope" $valor3 "$ope" $valor4`
echo El resultado es = $resultado;;
f|F) exit;;
*)echo "Opción no válida, SIGUE INTENTANDO";;
esac
done
```

Conclusiones

Podremos encontrar una gran diversidad de comandos y librerías si nos ponemos a comparar ambas plataformas, pues vamos a notar que difieren más de lo que pensamos en un principio, pero al final vamos a notar que tienen una gran similitud en cuanto a su estructura, por que

eso es un rasgo que simplemente no los puede hacer tan distintos uno del otro.

Los ejemplos de este artículo difieren bastante, pero si tenemos nociones básica de programación, no se nos hizo difícil encontrar la lógica del programa hecho en UNIX; en general, en C todo lleva una estructura definida, eso es lo que lo hace una herramienta tan poderosa.

Es importante tener bien definida la diferencia entre plataformas, para saber aplicar la extensa teoría que cada una abarca, aunque el artículo no está enfocado en eso, espera en posteriores publicaciones, para que todos conozcamos el grandioso mundo UNIX.

Raúl García Chetla.

Trivia.

1.-Imagina que compras un globo en el zócalo lleno de gas Helio, de esos que se elevan solos, y necesitas pesarlo. ¿Cómo pesarías el globo lleno de gas?

2.- ¿Qué le pasa a un televisor si lo conectas a 220 VAC?

3.- ¿Por qué el cielo se ve azul?

4.- ¿Por qué el océano pacífico profundo se ve azul?

5.-Imagine que tiene un electrón libre. ¿Qué le pasaría si lo conectáramos a tierra?

By R.Doradus.

Premios:

1 CD con libros de muy alto interés.

1 dsPIC30F3011.

Revistas de electrónica.

Envía tus respuestas a blastedrobotics@yahoo.com.mx